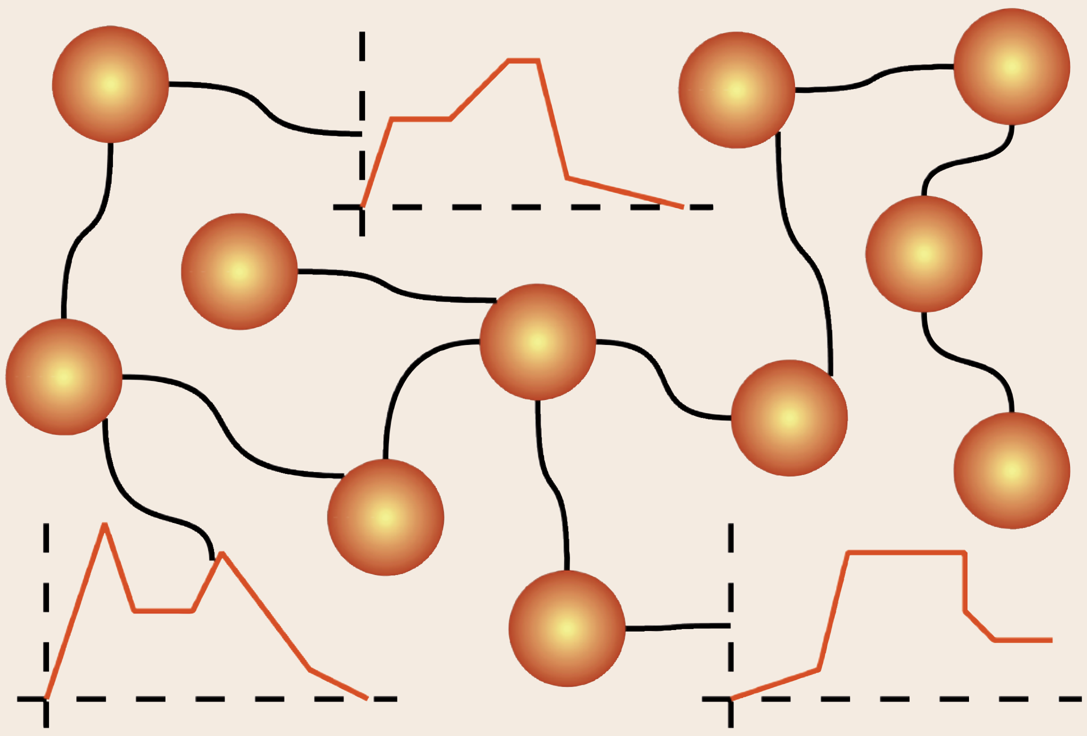
Gábor Nagypál

Possibly imperfect ontologies for effective information retrieval



universitätsverlag karlsruhe

Gábor Nagypál

Possibly imperfect ontologies for effective information retrieval

Possibly imperfect ontologies for effective information retrieval

by
Gábor Nagypál

universitätsverlag karlsruhe

Dissertation, Universität Karlsruhe (TH)
Fakultät für Informatik, 2007
Referenten: Prof. em. Dr. Dr. h. c. Peter C. Lockemann, Prof. Dr. Rudi Studer
Tag der mündlichen Prüfung: 16.05.2007

Impressum

Universitätsverlag Karlsruhe
c/o Universitätsbibliothek
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

Dieses Werk ist unter folgender Creative Commons-Lizenz
lizenziert: http://creativecommons.org/licenses/by-nc-nd/2.0/de/

Universitätsverlag Karlsruhe 2007
Print on Demand

ISBN: 978-3-86644-190-3

To Bea and Bendi.

Acknowledgments

I wrote this work during my time at the FZI Research Center for Information Technologies. I would like to thank to all of my colleagues at FZI and especially to Simone Braun, Stephan Grimm, Heiko Paoli, Andreas Schmidt, Peter Tomczyk, and Andreas Walter for the great working atmosphere that made my years at FZI enjoyable.

My very warm thanks go to Dr. Wassilios Kazakos. He supported my work during his time in FZI and also thereafter. He read and commented many versions of my thesis from the earliest versions to the latest ones. We also share a common interest in history and Wassili's historical comments definitely improved the historical quality of my examples in the thesis. Finally, Wassili also helped me to start my life after my PhD work (yes, it seems that such a thing really exists).

I am deeply indebted to Dr. Boris Motik who helped me to polish my ideas on fuzzy time intervals that finally lead to this thesis. Boris also showed me how to pursue a PhD research and how to successfully finish it, just by doing it perfectly right.

I am much obliged to Prof. em. Dr. Dr. h. c. Peter C. Lockemann, my thesis supervisor. He helped start my research carrier in Germany by inviting me to FZI as a Marie Curie fellow. Later he supported my application as a PhD student to Karlsruhe. He read and corrected all of my written ideas and thesis versions amazingly fast but thoroughly during the last years. It is clear that without him this thesis would not exist. I would also like to thank to Prof. Dr. Rudi Studer who accepted my request to be my second supervisor despite of his numerous duties. In general, the scientific works of Prof. Studer's group were always a great inspiration to me.

My warm thanks go to dr. Sándor Gajdos, my thesis supervisor in Hungary, for his support in the early times and also that he encouraged me to make this huge step and continue my research in Germany.

I would like to thank to my students. Without their support, work and ideas it would have been impossible to implement and evaluate such a complex system. József Vadkerti implemented the first version of the semantic search engine on Lucene basis. Zhigao Yu worked a lot on the user interface of the IRCON prototype. Andreas Hintermeier implemented the tool to construct fuzzy time intervals. Finally, Ümit Yoldas elaborated on and implemented my first vague ideas about metadata generation. His work served as the basis for the metadata generation part of this thesis.

Last but not least my biggest thank goes to my family. Without the whole-life support of my parents I would have not even dared to start my PhD research. My wife, Bea and my son, Bendi, to whom this thesis is dedicated, tolerated the many evenings, nights and weekends I had to steal from them to finish this work. Bea persuaded me to stay in Germany and finish my thesis here. Her continuous support was invaluable for successfully finishing this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The sheer amount of information that is available for users on the World Wide Web, in an enterprise portal, or even on their own desktop is unmanageable without using effective and efficient search engines. E.g., it would be nearly impossible to use the Web of today without an efficient search engine, such as Google[1].

Although the quality of search engine results has improved significantly in the last few years, the results are still not adequate in many cases. The main reason is that current commercial search engines are still based on the full-text search paradigm, i.e., they are still based on the assumption that relevant documents must be somewhat similar to a user query on the syntactic, textual level. Searching for relevant documents using a classical, syntax-based search engine is therefore similar to a "prediction game" [vB98]. The users try to predict which terms are included in relevant documents, and document providers try to predict which terms interested users would use.

There are lots of situations when it is easy to "win" the prediction game, i.e., to use the proper terms in our query. In this case existing search technology works well. This is usually the case when we search for concrete, characteristic things, such as a specific person, product or website, which has a distinguishing name. In many other cases, however, the syntactic (dis)similarity does not reflect semantic (dis)similarity, and classical systems fail. This is especially the case when the searcher has a more abstract, research type of information need, which is formulated using vague, higher level concepts[2]. Many relevant documents will not contain those vague concepts directly but will be related only at a semantic level. Moreover, some special queries, such as spatial or temporal queries, do not really fit into the term-based paradigm. It is very hard to find documents relevant for the "in the XX. century" information need by using simple keyword matching.

This problem is well-known in the information retrieval community. A possible solution to solve it is to annotate resources with *semantic metadata*, which describes the meaning of the document. Using this metadata it is possible to find also such relevant documents that are syntactically not similar to the query terms.

Semantic metadata is usually specified using a controlled vocabulary, i.e., a finite set of carefully chosen terms. The motivation for constraining the set of vocabulary elements is to avoid the ambiguity of unconstrained natural language. In the field of classical information retrieval

[1]`http://www.google.com`

[2]Such as: "Important *political events* during the *Second World War*"

(IR), so called *thesauri* are used to specify the controlled vocabulary. Terms in a thesaurus are normally structured using linguistic relations, such as narrower terms, broader terms, and synonyms.

After Tim Berners-Lee coined the idea of the Semantic Web [BLHL01], ontologies – shared, explicit specifications of a conceptualization describing a specific domain [Gru93] – became popular. Ontologies together with semantic metadata form the basic infrastructure of the Semantic Web vision. Ontologies define a controlled vocabulary in the first place, like thesauri. Ontologies define vocabulary elements, however, on an abstract, semantic level. That means, ontology elements are not terms any more but abstract notions, identified by language independent, globally unique identifiers (URIs). Moreover, relations among ontology elements are not necessary linguistic relations any more, instead arbitrary semantic relations are allowed. To summarize, ontologies can describe a domain more precisely than a thesaurus, in a language independent way.

In general, it seems compelling and intuitive that semantic metadata help improve the *effectiveness*[3] of search engines. After all, having high-quality semantic annotations, it seems to be a trivial task to retrieve all semantically relevant documents to a query. Unfortunately, this is not true in practice.

It is a very expensive, and time consuming task to develop a thesaurus. Therefore, the coverage of a thesaurus is almost never complete, i.e., there will always be important terms that are not included in the thesaurus, and therefore cannot be used in semantic annotations. On the other hand, if we try to include all kinds of terms into the thesaurus, it will be soon too complicated and large for practical use, and therefore it will fail one of its main goals, namely to reduce the ambiguity of natural language.

If a thesaurus is already available, it is still an expensive and time consuming task to annotate existing documents with the terms it contains. It will therefore often happen that important annotations will be missing, and some of the annotations will even be wrong. These arguments also hold for ontologies that provide an even more powerful, and thus more complicated, model for knowledge representation than thesauri. Therefore, it is important to see that in the real world, ontologies and semantic annotations are almost always imperfect in some way.

Considering these issues, it is not really surprising that in general, it cannot be shown that thesauri improve the effectiveness of IR systems. On the contrary, systems operating solely on thesaurus-based annotations are usually less effective than systems directly using document texts [Cle91, Sal86]. The reasons are inadequate coverage and errors in annotations. Recent research also shows, however, that the combination of thesaurus-based search algorithms with full-text search algorithms can improve effectiveness in some domains[SRN04, Cro00].

In most of the state-of-the art ontology based systems it is silently assumed that ontologies, when used in an IR system, automatically improve effectiveness. Based on the experiences with thesauri, however, it is clear that this assumption cannot be accepted without further validation. There is a hope, of course, that the assumption that ontologies help IR is not false. After all, ontologies provide much more sophisticated means to precisely represent domain knowledge. I.e., there is a possibility that the positive effects of using a precise, semantic

[3]It is important to distinguish between *efficiency* and *effectiveness* of an IR system. *Efficiency* means that the searcher gets the results fast, *effectiveness* means that the quality of results is good.

domain model in IR will outweigh the inherent negative effects of imperfection that cannot be fully avoided.

Therefore, *the main goal of this thesis is to validate the assumption that ontologies help improve information retrieval effectiveness.* Special attention is given to the issue of imperfection in ontologies and semantic metadata. It is important to address this issue because, as classical IR research on thesauri (i.e., on very simple ontologies) shows, imperfection has the potential to invalidate the intuitively compelling assumption about the usefulness of ontologies for IR.

Another main concern of this thesis is the *efficiency* of ontology-based information systems. Because of the increased representation power of ontologies, reasoning with such ontologies has very high complexity. Moreover, ontology reasoners are yet immature, their performance is not comparable with other well-established technologies, such as relational databases and full-text search engines. It is therefore a very challenging task to keep the good efficiency of current IR systems, while at the same time improving their effectiveness using ontologies — which is the ultimate goal of this research.

The main issues toward these goals are the following:

Place of ontologies in the IR process: It has to be analyzed where ontological knowledge can be exploited during the IR process.

Methodology for using ontologies in an IR system: This work aims to provide a holistic solution for the application of ontologies in IR systems.

Representing imperfect domains: Not only ontologies and semantic metadata can be imperfect but also our mental model (conceptualization) can be vague, uncertain and subjective. This latter kind of imperfection, which I term *domain imperfection*, has to be considered as well. This work gives solutions for representing domain imperfection in the temporal dimension.

Scalability issues: Reasoning in ontologies using expressive formalisms, such as OWL-DL[4], is highly complex. In classical IT terms ontology reasoning is non-tractable[5]. On the other hand, scalability (in terms of the size of the documents) is very important, especially in Web information systems, where millions or even billions of documents have to be managed. Because of the theoretical worst-case complexity of ontology reasoning it is very unlikely that solutions using solely ontology reasoning for information retrieval will scale so well that they can be used in large web information systems. This work therefore examines the possibility of reusing existing, scalable IR technology as part of an ontology-based information system.

Evaluation of IR effectiveness: As mentioned, the statement that ontologies help improve IR effectiveness cannot be accepted without further evaluation because of the issue of imperfection. Therefore, it has to be experimentally evaluated whether in general it is possible to improve IR effectiveness using ontologies. In this work, evaluation is conducted using the classical and well-accepted evaluation measures of IR — precision and recall. It is also of interest whether higher quality, more precise ontologies provide better

[4]OWL-DL is the ontology standard promoted by the W3C at the time of writing of this thesis. See [Mv04]

[5]Reasoning in OWL-DL is NEXPTIME-complete, while already NP-complete problems are usually considered as non-tractable

results than simpler and thus less precise ontologies. A positive result would motivate using more complex, more powerful ontology formalisms, such as OWL-DL, instead of simpler, light-weight ontologies, or thesauri.

The main idea of this thesis is to solve these issues by the combination of classical IR technologies with ontology-based reasoning. In particular, I propose to use ontologies only in those IR process steps where scalability is not an issue. This is the case during the indexing phase where semantic metadata of documents are automatically generated. For the query execution itself, I propose to use classical full-text search to retrieve documents. This technology has been proved to scale well even to the size of the whole Web[6].

The basic assumption underlying this approach is that it is possible to create such document and query representations so that the classical, syntax-based ranking algorithms yield semantically meaningful results.

The thesis is organized as follows:

In Chapter 2, I introduce the fundamentals of the areas information retrieval and ontologies, which are needed to understand the discussions in this thesis. In Chapter 3, I analyze why full-text search engines fail to deliver high-quality results for some types of queries. Based on this analysis, I identify the major requirements that an ontology-based information system should fulfill. Based on these requirements, I analyze the state of the art in ontology-based information systems in Chapter 4. In Chapter 5, I provide a high-level overview of my approach that meets the identified requirements. In the later chapters, I address specific issues of my solution in more detail.

Chapter 6 introduces a fuzzy temporal model to represent imperfect temporal information, and also discusses how to create such intervals. Chapter 7 identifies the requirements of an ontology formalism that can be used in my system, and describes the ontology formalism that I used during my work. Chapter 8 discusses how to use the background knowledge stored in the ontology to create high quality semantic annotations. Chapter 9 shows how exploit those semantic metadata to improve the effectiveness of search: how to expand the user query; and how to combine the results of various queries to diminish the negative effect of ontology imperfection.

For an ontology-based information system, an ontology is always needed. Chapter 10 introduces a methodology to create an ontology. This methodology is based on the experiences during the VICODI EU project, and it is customized for the needs of ontology-based information retrieval.

The ideas presented in this thesis were implemented in a research prototype. Chapter 11 discusses some implementation issues and also introduces the prototype system from the end-user's prespective. Chapter 12 evaluates the ideas presented in the previous chapters by discussing the experiments made using the prototype implementation of the information system. Finally, Chapter 13 concludes this thesis and provides some outlook.

[6]Consider e.g., Google.

Chapter 2

Fundamentals

The two main areas of this thesis are information retrieval (IR) and ontologies. This chapter introduces the fundamentals of these two areas, which are needed to understand the later discussions.

2.1 The IR process

To effectively support information retrieval, one must first understand how users search, i.e., the IR process must be analyzed.

Figure 2.1 shows a typical IR process. At the beginning, the user would like to satisfy some abstract *information need*. She or he has to *formulate a query* representing this information need (or part of it), which can be executed by the system. This query is submitted to the system. The system itself is free to automatically *transform* (usually expand) *the query*, if needed.

The IR system does not usually operate on the physical documents themselves but rather on a representation of the documents in a specific information model. This representation is created during a *document indexing* step. Finally, the document representations are *matched* against the query, and the matching representations are *ranked* according to an algorithm, and returned to the user. The result list normally contains a user friendly summarization of the document, which is generated from the document representation stored in the system. Based on this summary, the user can easily proceed to the physical document (if it is available in the system).

There are many different ways to represent documents and queries in an IR system, and it is not necessary that document representations and the query representation use the same information model. E.g., in classical Boolean retrieval [BR99, pp. 25–27], the document is represented as a flat list of terms, and the query is a logical formula over the query terms. There are many other models, however, which represent the query and the documents in the same information model. E.g., the vector space model (VSM) [BR99, pp. 27–30] represents both documents and queries as weighted lists of terms. If the query and the documents are represented in the same model, the matching and ranking step is usually implemented by defining a suitable similarity measure on the information model. The result is the list of documents that are most similar to the query according to this similarity measure. The list is ranked by the similarity score of the documents.

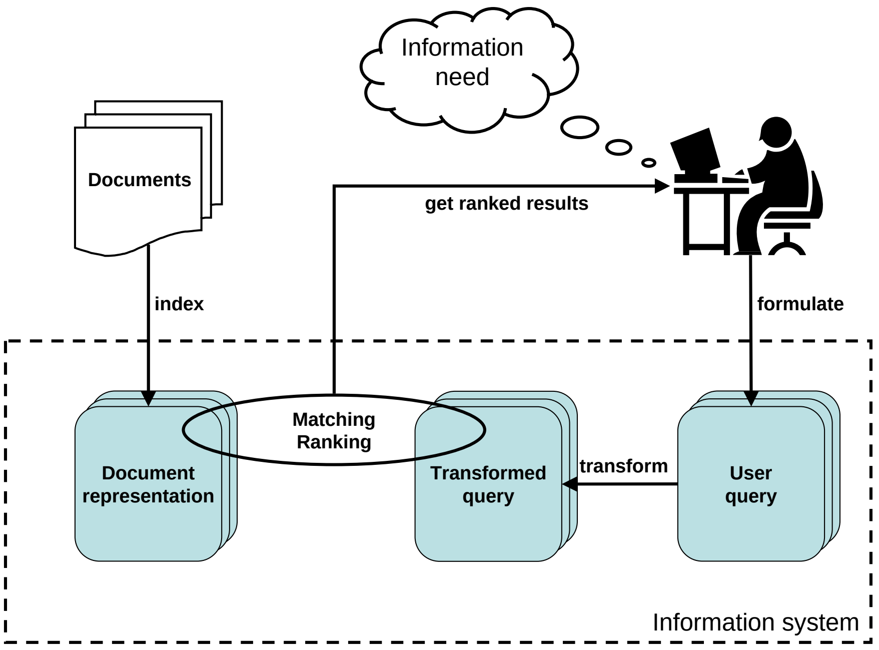


Figure 2.1: The information retrieval process

While the indexing step usually happens only once, and the document representations are cached in the IR system, the “formulate query–execute–examine results” cycle normally happens many times. Typically, users do not know their information need exactly. Instead, the information need evolves during the search process, based on the information the user sees during her or his search. Bates coined the term “berry picking” in her seminal works [Bat90, Bat89] to describe the behavior of a typical user during the search process. The metaphor refers to the process where users formulate many simple, possibly semantically independent queries, which describe only parts of their complex information need. Moreover, users also jump between different information sources. In other words, they collect small pieces of information (the berries) in the forest of information systems. Additionally, Bates also noted that a typical search process continually evolves, i.e., the information need of the users changes based on the documents they viewed during the process. E.g., a user can start with the information need “Events of the Russian Revolution” but she or he can get fascinated by one of the key figures of this event, and end up with searching for details about this person.

2.2 Data retrieval vs. information retrieval

It is important to see the difference between classical data retrieval (which happens, e.g., in database management systems) and information retrieval.

In information retrieval the usual assumption, which is validated by everyday experience, is that most users cannot formulate their information need precisely in form of a query. There are many reasons for it. First, as was described above, users are often not sure about their information need. Second, it is very common that users do not know the query formalism that well. This problem can be solved by educating the users. Further, in many cases users are simply not ready to invest a bigger amount of time to formulate a complex, precise query but specify only a rough approximation of their real information need. They hope that the query representation is “close enough” to their information need, so that the system will provide meaningful results, and thus they save time by avoiding the tiring mental process of formulating a complex query. Some recent studies analyzing the logs of major search engines show that the average query submitted by users contains only two or three query terms [JP01, SJ04]. Finally, it is also possible that the query formalism itself is not powerful enough to faithfully represent the information need.

Based on this discussion, it is easy to see that in most cases one IR query is only an approximation of the real information need. It is therefore even theoretically not possible to provide the “perfect” answer(s) to a query but rather a set of possibly good answers is needed. On the one hand, we need to consider partial matches, or similarity between the query and the candidate resources in the repository. On the other hand, the answers should be *ranked* according to the probability that they are relevant to the original, ill-defined information need.

This is a fundamental difference between information retrieval and data retrieval in databases. In the latter case, it is assumed that the query perfectly describes the information need of the user (or a program), and therefore the perfect set of answers can be returned. Here, ranking of the answers is not needed, as each and every answer that matches the query is completely relevant to the original information need. There is a recent movement to integrate the merits of IR and database systems [AYCR$^+$05]. Researchers strive to keep the possibility of asking structural queries but provide efficient ranking of results at the same time. It is questionable today, however, whether it is possible to meet these goals in one system.

2.3 Full-text IR and the vector space model

The state of the art in IR is still full-text search today. Many of the popular systems use some kind of term based information model. In these models documents are represented as some structure of *terms*. A term is usually a word in the document but it can also be a phrase, or any other character sequence, such as a URI.

One of the most popular term-based information models for document and query representation in IR systems is the vector space model (VSM) [BR99]. This model represents both documents and queries as a weighted set of terms. The basic assumption underlying the model is that the terms are independent of each other, i.e., if there are t terms in the whole document repository, one term represents one dimension in the abstract t-dimensional term space. This means, each document representation is a t-dimensional vector $\vec{d_j}$, where the i^{th} element of the vector represents the weight $w_{i,j}$ of the term t_i in document d_j. The query vector $\vec{q}$ is interpreted similarly.

Using this metaphor of a t-dimensional space, the similarity between two representations is calculated in this model using the cosine metric in Eucledian spaces:

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{\left|\vec{d_j}\right| \cdot |\vec{q}|} \tag{2.1}$$

$$= \frac{\sum_{i=1}^{t} w_{i,j} \cdot w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \tag{2.2}$$

There are many different ways to calculate the $w_{i,j}$ weights of the terms but practically all of the state-of-the-art systems using the VSM use some variation of the TF/IDF heuristic. TF stands for term frequency, IDF stands for inverse document frequency. Informally, the term frequency counts how many times a term appears in a document. The document frequency of a term counts in how many documents a term appears in the document collection. The inverse document frequency is the inverse of this number. The TF/IDF heuristic states that the term weight is related to these two frequencies, i.e.,

$$w_{i,j} = TF \cdot IDF \tag{2.3}$$

As term frequency value it is usual to use the $f_{i,j}$ relative frequency of a term, i.e.,

$$f_{i,j} = \frac{F_{i,j}}{max_l F_{l,j}} \tag{2.4}$$

where $F_{l,j}$ denotes the absolute frequency of the term t_l in document d_j. The inverse document frequency idf_i of term t_i can be calculated as

$$idf_i = log\frac{N}{n_i} \tag{2.5}$$

where N denotes the total number of documents in the system, and n_i denotes the number of documents where term t_i appears.

These formulas for calculating the term weights represent just one possible way. In practice, almost every IR system applies its own heuristic for calculating the term weights. These heuristics usually always include the relative frequency and the inverse document frequency of terms in some way but sometimes they also consider other metrics. E.g., the OKAPI TF score [RWB99], which provides the best results in IR experiments, also considers the length of the actual document, and the length of the average document in the repository.

Although the term independence assumption, which underlies the VSM, clearly does not hold in the real world[1], the model still produces surprisingly good results. Although there are lots of other, more complicated, term-based IR models available, the results of VSM is comparable with the results of more complicated models. On the other hand, the VSM is very simple, easy to comprehend for users, and it is easy to compute the similarity scores. That explains its popularity.

[1]E.g., the words“computer” and “network” are more likely to appear together in a document, than the words “computer” and “lunch”.

2.4 Efficiency and Effectiveness

It is important to clarify some IR terminology, which is frequently used in the literature, and which will also be used in this work. Generally, there are two concerns of an IR system. First, it should provide good results, which fulfill the information need of the user. If a system fulfills this goal, we say it is *effective*, or it performs well. Therefore in the area of IR, the word *effectiveness* always refers to the quality of results. In many cases, the word *performance* is also used to refer to the quality of results. To avoid confusion with the word performance referring to the execution speed of a system, I will use the phrase *retrieval performance* instead. This terminology is also common in the IR literature.

The other concern of an IR system is that it should provide high-quality answers fast. Nowadays, users are not ready to wait for a long time to get the results of the system. If the user abandons an IR system because it is too slow, it does not matter how good the results are the system would otherwise provide. If a system provides its answers fast, we say it is *efficient*. During this thesis, I will also use the word *performance* to denote the speed of a system, i.e., its efficiency. This is usual in most areas of computer science.

Clearly, the ultimate goal of IR research is to provide information systems that are both effective and efficient.

2.5 Representing background knowledge

2.5.1 Thesauri

Besides various term-frequency heuristics, another way to improve search effectiveness is to incorporate background knowledge into the search process. The IR community concentrated so far on using background knowledge expressed in the form of thesauri [BR99, pp. 170–173]. Thesauri define a set of standard terms that can be used to index and search a document collection (controlled vocabulary) and a set of linguistic relations between those terms. Typical linguistic relations are the “narrow term” or “hyponym”, and the “broader term” or “hypernym” relations. Examples of thesauri are the HASSET thesaurus[2] (Humanities and Social Science Electronic Thesaurus), and the GEMET thesaurus[3] (GEneral Multilingual Environmental Thesaurus).

Probably the most famous thesaurus is the Wordnet[4], which defines linguistic relations for the whole English language. Wordnet, however, is also called an “ontology” sometimes because it defines the relations between so-called “synsets” — defining a specific meaning of a word — and not between simple terms. This also shows — as we will see soon — that the notions “thesaurus” and “ontology” are not clearly defined, and the borders between the two notions are fuzzy.

[2]`http://www.data-archive.ac.uk/search/hassetSearch.asp`
[3]`http://www.eionet.eu.int/gemet`
[4]`http://wordnet.princeton.edu`

2.5.2 Ontologies

A more powerful way to represent background knowledge about a domain than using thesauri is to use ontologies. Ontologies form the basic infrastructure of the Semantic Web [BLHL01].

Originally, ontology was a philosophical science defined as the "study of existence, of all the kinds of entities — abstract and concrete — that make up the world." [Sow00]. Intuitively, the science of ontology defines the domain of discourse we can make statements about, i.e., it defines the predicates of a logical formalism. In other words, ontology connects the abstract formulas of a logical formalism with the real world.

The fundamental question of ontology is: "What is there?". The simplest answer "Everything" is useless, and therefore different ontological theories of various philosophers provide different categorizations of "things" that exist in the world. These categorizations normally specify a tree, starting with the most abstract category of THING or BEING, and defining a hierarchy of categories. In this hierarchy each member of a subcategory is also a member of its supercategory (normally referred to as an "is-a" relation in computer science).

In the area of knowledge representation, researchers use the definition of Gruber [Gru93], which defines an ontology as a "specification of a conceptualization". Note that here an ontology is a concrete artifact and not an abstract science any more. Therefore in the context of knowledge representation it makes sense to speak about "an ontology" or "ontologies".

As everyone can feel, this definition of Gruber is quite general, which caused a lot of confusion about what the word ontology really means. As the idea of the Semantic Web became more and more popular among researchers, the word "ontology" became more and more popular, too. As a result of that, today almost every knowledge representation formalism is called "an ontology", which actually renders this word empty.

Interestingly, this observation, i.e., that "many knowledge representation formalisms can be called as an ontology" is explicitly stated in [SW01], where also a summary of existing formalisms is given, which were considered as an ontology in one or more scientific publications. I cite this summary in Figure 2.2. As can be seen, formalisms that are compatible with Gruber's definition range from catalogs (simple list of things) to logical formalisms that support automatic reasoning. A very similar categorization is given in [McG03].

Because the original definition of an ontology is so vague, I felt it useful to provide a slightly more restrictive, and thus more precise, definition of what an ontology is.

Definition (Ontology)**.** *I consider in this thesis as an* ontology *any formalism with a* well-defined mathematical interpretation*, which is capable at least to represent a* subconcept taxonomy*,* concept instances *and* user-defined relations *between concepts.*

2.5.3 Ontology modeling constructs

In a typical ontology meeting my definition there are the following modeling constructs:

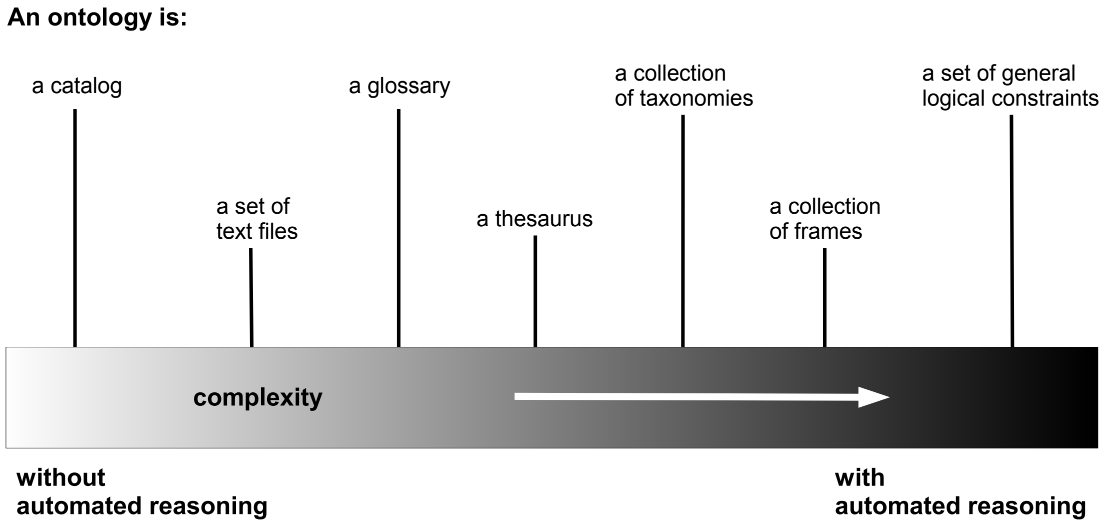


Figure 2.2: Different types of ontologies

- Elements of the knowledge domain are called *instances*. Examples are NAPOLEON BONAPARTE or GEORGE W. BUSH.

- Sets of instances are called *concepts*. Examples are PERSON or COUNTRY. Instances of a concept are connected to a concept via the "is a" or "instance of" relation. In contrast to object oriented programming, most ontology languages do not require that an instance belongs to exactly one concept. E.g., NAPOLEON BONAPARTE can be the instance of EMPEROR and GENERAL at the same time.

- Concepts can be organized into a "concept hierarchy" or "concept taxonomy" via the "subconcept of" relation. E.g., EMPEROR can be modeled as a subconcept of PERSON. A subconcept of relation is only valid formally if all instances of the subconcept are also instances of the superconcept[5]. In our case, all emperors are persons, therefore it is a correct subconcept relation. An example for a formally invalid subconcept relation is the Politics $\rightarrow$ Elections relation[6] where an election is not a "politics" but rather belongs to the politics topic. The requirement of the formality of the subconcept relations is one of the major differences between ontologies as I understand them and other less formal structures such as thesauri.

- Concepts can be connected via *relations*. E.g., two PERSON concepts can be connected via the MARRIEDWITH relation. Many ontology languages support only *binary relations*, i.e., relations that connect exactly two instances.

[5]This definition represents an *extensional* interpretation where the definition of the subconcept relation is based on the instances, i.e., on the extensions of the participating concepts. An *intensional* definition of the subconcept relation is also possible that is based on the logical definition of the concepts. In this case, the definition of the subconcept should logically include (imply) the definition of the superconcept. These two kinds of definitions define the same subconcept relation.

[6]taken from `www.yahoo.com`

- *Attributes* containing data values can be defined on concepts. E.g., the HEIGHTINCM attribute can be defined on the concept PERSON
- Sometimes attributes and relations together are termed as *properties*.
- Instances are connected via instances of relations of their concepts, i.e., via *relation instances*. E.g., the instances NAPOLEON and JOSPEHINE can be connected via an instance of the MARRIEDWITH relation.
- *Attribute values* can be specified on instances. E.g., the value of the HEIGHTINCM attribute on the instance NAPOLEON would be 168.
- Properties (relations and attributes) can sometimes have *subproperties*. The validity criteria for the subproperty relation is similar to the subconcept relation: an instance of the subproperty should be always a valid instance of the superproperty. E.g., we can define the LIVESWITH relation as a superproperty of the MARRIEDWITH relation.
- Many ontology formalisms separate the concepts, instances and properties from each other. I.e., if something was modeled as a concept, it cannot be a property or an instance in the same ontology. Sometimes it is useful, however, if some entities can be instances, concepts or properties at the same time. E.g., an APE is an instance of the SPECIES concept but at the same time the concept of the AMY, THE APE[7] instance. If an entity may play different roles in an ontology formalism, we say that the formalism supports *metamodeling*.
- Ontology formalisms work with abstract entities, which are identified by unique, language independent identifiers (in many case URIs). It is therefore a common practice to define a separate *lexical layer* in the ontology, which provides language-dependent labels and their synonyms for the specific abstract ontology entities (see Figure 2.3). E.g., the instance URN:ONTOLOGY:NAPOLEON_I can have the label "Napoleon Bonaparte" and the synonyms "Napoleon I of France", "Napoleon I" etc. Using this technique it is also possible to define different lexical layers for different languages, while reusing the same abstract ontology structure. Usually, also some informal description of the ontology entities (documentation) is stored in the lexical layer.

  The lexical layer has two very important roles. First, it makes it easier for humans to browse the ontology. Second, it is crucial for automatic mapping of natural language texts to ontology elements.

2.5.4 Semantic annotations

The usual way to use ontologies and thesauri in IR is to annotate documents with the elements of the ontology (or thesaurus). This annotation is usually called *semantic annotation* or *(semantic) metadata*. I will also use these terms interchangeably in this thesis.

[7] Amy, the ape who could communicate using sign language, is one of the main characters in Michael Crichton's famous novel, Congo.

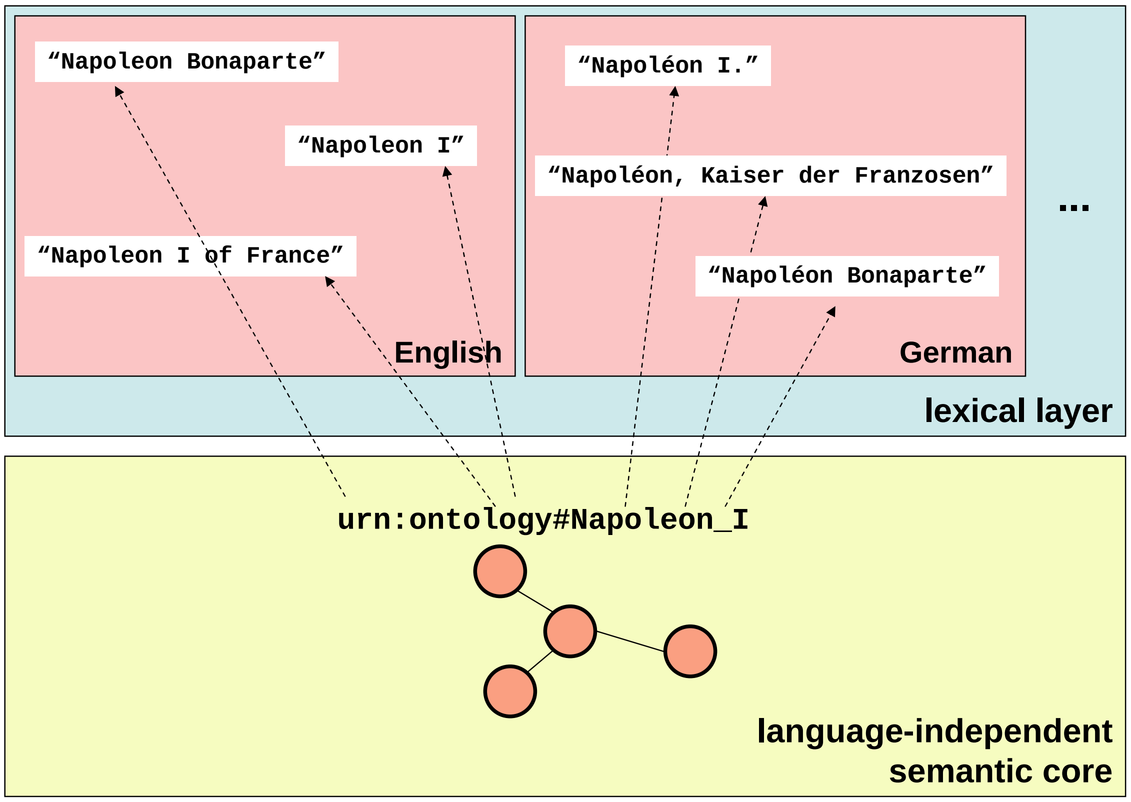


Figure 2.3: Semantic core and lexical layer in ontologies

2.6 Summary

In this chapter I introduced the fundamentals of information retrieval and ontologies that are necessary to follow the discussion in later chapters. First, I analyzed the information retrieval process and discussed the difference between information and data retrieval. Later, I introduced the vector space model and the popular TF-IDF heuristic to calculate similarity between document representations in this model. I also clarified the notions of efficiency and effectiveness.

In the remaining part of the chapter, I introduced ontologies. First, I gave a more precise definition of an ontology than the most cited definition of Gruber. Later, I analyzed the usual structure of an ontology fulfilling this definition.

Chapter 3

Problem analysis

In this chapter, the causes are analyzed why full-text search fails on specific kinds of searches and how ontologies could help to solve these problems. I will show how the possible imperfection of ontologies makes the claim that ontologies help improve the efficiency of information retrieval unacceptable without thorough evaluation. I also identify other crucial features, such as scalability and user friendliness, that an ontology-based information retrieval system, which strives to solve the deficiencies of full-text search, should fulfill. I formulate the identified features of this chapter as formal requirements toward the new ontology-based system. These requirements will serve as a basis for the discussion in later chapters.

3.1 Motivating scenario

To guide the problem analysis about the impact of ontologies on information retrieval and also to motivate the use of ontologies in the information retrieval process, I will use the VICODI system that was developed during the EU IST VICODI project with my active participation [NDO05]. The VICODI system is a typical ontology-based information system and thus shows most of the opportunities and problems of this kind of systems.

The goal of the VICODI project was to demonstrate the utility of the so-called *visual contextualization* by building a web portal for European history[1]. The main idea of visual contextualization is to visualize (a part of) the document context to make the document content more comprehensible for the user[2].

Although there were already some attempts to give a thorough definition for what a context is (e.g., [Dey01]), these definitions are vague and not generally accepted. In the following discussion I use the word “context” in its usual, informal meaning in the English language[3].

In the case of the VICODI project, the spatial and temporal context of a document was visualized. The context visualization was based on the semantic metadata that was semi-automatically generated for the documents. Semantic metadata of the documents used an ontology of European history. In the `eurohistory.net` portal, which was developed during the project, we

[1] Available under `http://eurohistory.net`

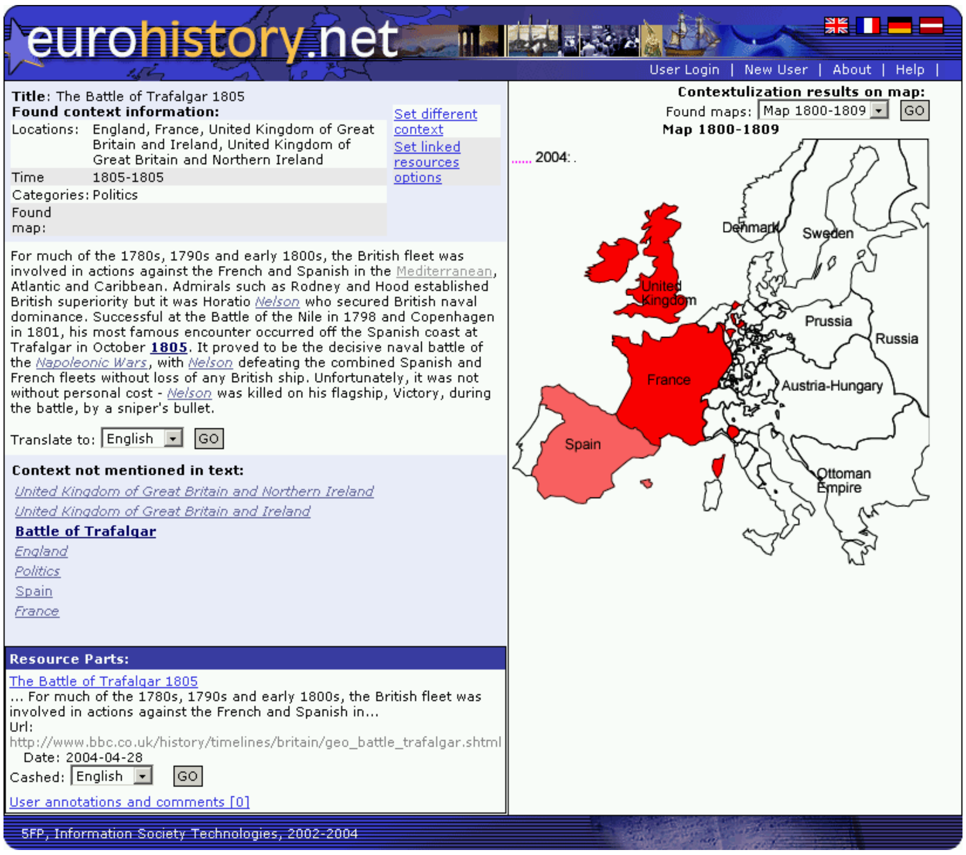
[2] Using a well-known metaphor from knowledge management, knowledge is “information in context” [Rum01].

[3] According the Merriam-Webster dictionary, context is “the interrelated conditions in which something exists or occurs”.

displayed historical maps together with documents. The map reflected the actual historical period of the document where important geographical and political regions were highlighted.

In addition, also navigational elements were generated, which allowed the user to initiate new semantic queries simply by clicking on these elements. Terms in the text that were part of the semantic metadata of the document were highlighted. These terms and also the elements of the map were clickable and initiated so called *context-sensitive queries*. These queries operated on the semantic metadata of documents in the repository. In addition to the element that was clicked on, the whole semantic metadata of the actual document was considered. I.e., if the user clicked on "Russia" on the map and the actual document was about "Napoleon", she got documents where both Russia and Napoleon appeared. In other words, the user got information about Russia in the context of Napoleon. That is where the term "context-sensitive query" came from.

For example, Figure 3.1 shows a document about the Battle of Trafalgar, which was visually contextualized using the aforementioned techniques.



Title: The Battle of Trafalgar 1805
Found context information:

Locations:	England, France, United Kingdom of Great Britain and Ireland, United Kingdom of Great Britain and Northern Ireland
Time	1805-1805
Categories:	Politics
Found map:	

For much of the 1780s, 1790s and early 1800s, the British fleet was involved in actions against the French and Spanish in the Mediterranean, Atlantic and Caribbean. Admirals such as Rodney and Hood established British superiority but it was Horatio *Nelson* who secured British naval dominance. Successful at the Battle of the Nile in 1798 and Copenhagen in 1801, his most famous encounter occurred off the Spanish coast at Trafalgar in October **1805**. It proved to be the decisive naval battle of the *Napoleonic Wars*, with *Nelson* defeating the combined Spanish and French fleets without loss of any British ship. Unfortunately, it was not without personal cost - *Nelson* was killed on his flagship, Victory, during the battle, by a sniper's bullet.

Translate to: English

Context not mentioned in text:
United Kingdom of Great Britain and Northern Ireland
United Kingdom of Great Britain and Ireland
Battle of Trafalgar
England
Politics
Spain
France

Contextulization results on map:
Found maps: Map 1800-1809
Map 1800-1809

Resource Parts:
The Battle of Trafalgar 1805
... For much of the 1780s, 1790s and early 1800s, the British fleet was involved in actions against the French and Spanish in...
Url:
http://www.bbc.co.uk/history/timelines/britain/geo_battle_trafalgar.shtml
Date: 2004-04-28
Cashed: English
User annotations and comments [0]

5FP, Information Society Technologies, 2002-2004

Figure 3.1: Visually contextualized document about the Battle of Trafalgar

To summarize, the `eurohistory.net` portal is an ontology-based information system, where a complex, large ontology is used in various ways to improve the services provided for the user. It is therefore a good case study to identify typical problems and merits of these kinds of systems. I will use therefore the `eurohistory.net` portal and its domain — European history, and history and news in general — in my examples throughout this thesis.

3.2 The prediction game

Van Bakel coined the term "prediction game" as a methaphor for the classical, text-based search process [vB98]. A syntax-based search is successful if the user can successfully guess the right terms that are contained in relevant documents, and only in relevant documents. Providers, on the other hand, have to guess which terms will be used by their potential users and use those in their documents or web sites[4].

For some specific kinds of information needs, it is easy to "win" the prediction game, i.e., to find the proper search terms. Guha and his colleagues categorize searches as *navigational* and *research searches*[5]. In navigational searches the users are interested to find a very specific document (or web site) by providing phrases they expect to find in the document. In many cases the user already knows the document, she or he only wants to find it again. In these kind of searches it is normally easy to choose the right terms and those terms describe the relevant document quite characteristically. E.g., when we search the web site of a company or a celebrity, or we search for a scientific paper and we know the authors and the title. In such navigational searches we can greatly profit from the impressive performance and simplicity of full-text searching.

In research searches, on the other hand, the user would like to find any (mostly unknown) resource about a specific object or topic. In this case, it is usually not trivial to guess the exact terms that will show up in relevant documents. It is especially hard to find the right terms if the user is unfamiliar with the topic, although she or he needs more information exactly in this case. Therefore, we can say that for research searches the existing syntax-based technology is less suitable and there is a big potential for improvements using semantic technologies. In this thesis, I concentrate on these types of questions.

3.3 Full-text search

3.3.1 Problems with full-text search

Pure text-based search fails when a search term is not found literally in relevant documents. The major cases when this happens are the following:

Vagueness of natural language: Synonyms, homographs and inflection of words can all fool algorithms that see search terms only as a sequence of characters.

Indirectly relevant concepts: There are many cases, where specific concepts[6] are not mentioned directly in the document text but they are still relevant for the document semantically. Clearly, current search engines cannot find those documents. High-level, vaguely

[4]There are various utilities and services to optimize keywords in web sites, such as `http://www.submitexpress.com/` or `http://www.wordtracker.com/`

[5]There are also other possibilities to categorize user search goals. E.g., Rose and Levinson [RL04] identify *navigational*, *informational* and *resource* types of searches, whereas their navigational category is more restricted than that of Guha et al. I use the categorization of Guha et al. because of its simplicity.

[6]Here I use the word concept in its usual, informal meaning and not as an ontology modeling construct that was introduced in Chapter 2.

defined abstract concepts like the "Kosovo Conflict", "Industrial Revolution" or the "Iraq War" are typical examples of this phenomenon. Other example of indirect concepts is when users search for the "European Union" and they do not find relevant documents containing only the words "Berlin" or "Germany". Similarly, if users search for "cars" they will not find documents mentioning only "BMW" or "VW". These indirect concepts can be explored only by exploiting semantic relations between concepts, such as the "part Of" or the "is a" relations.

Numeric dimensions: In the case of numeric dimensions we are usually not interested in exact matches but we expect that relevant documents contain a feature which is in a specific interval. In this case, there is usually no syntactic matching among terms in relevant documents. Good examples for such dimensions are time and space. If we search documents about the "20^{th} century" using exactly that phrase, we expect to find also relevant resources containing the character sequences like "1945" or "1956" because 1945 and 1956 as numbers are in the time interval from 1901 to 2000. Clearly, those relevant documents will not be found by simple keyword matching. Another example is the query "near Karlsruhe". In this case we are also interested in documents about Ettlingen, Wissembourg etc. because the spatial coordinates are in a specific area around Karlsruhe. Keyword matching fails in that case, too.

Most of the current systems can successfully handle various inflection forms of words using stemming algorithms in most cases. However, it seems that the lots of heuristics and ranking formulas using text-based statistics that were developed during classical IR research in the last decades [BR99], cannot master the other mentioned issues.

3.3.2 Requirements inferred by full-text search problems

Ideally, an information system should provide a solution for all of the mentioned deficiencies of full-text search engines. This also includes the deficiencies in the area of numeric dimension. In this work, I will address all the mentioned problems, however, in the numeric dimension area I will concentrate only on the temporal dimension and I ignore the issues with the spatial dimension. Therefore, based on the discussion above, the following requirements can be formulated.

Requirement (NL vagueness)**.** *An information system should handle the vagueness of natural language.*

Requirement (Semantic relations)**.** *An information system should exploit semantic relations to find also relevant concepts that are not mentioned explicitly in the document text.*

Requirement (Time dimension)**.** *An information system should support user queries concerning the time dimension.*

3.4 Ontologies for information retrieval

Apart from the usual statistical approaches, semantic annotation provides another way to improve IR effectiveness. Ontology formalisms that meet the ontology definition in Section 2.5.2

allow for a much more sophisticated representation of background knowledge than classical thesauri. They represent knowledge on the semantic level, i.e., they contain semantic entities — concepts, relations and instances — instead of simple words. Moreover, they allow for specifying custom semantic relations between entities and also for storing well-known facts and axioms about a knowledge domain (including temporal information). This additional expressive power (compared to thesauri) allows for the identification of the validity context of specific relations. E.g., while in the context of the “Napoleon invades Russia” event the “Napoleon – Russia” relation is valid, it does not hold in general[7].

Based on this discussion, it seems that ontologies theoretically solve all of the mentioned problems of full-text search.

3.4.1 Potential uses of ontologies in the IR process

I see the potential to exploit ontologies in the IR process (see Section 2.1) at the following places:

Query formulation: The ontology as a kind of controlled vocabulary can support the user in formulating the query. First, the user gets an idea of what is available in the document repository by checking the ontology. Second, she or he can also see the semantical relations among the ontology entities. This helps her or him to navigate to potentially interesting ontology entities and add them to the query.

Query expansion: Based on the semantic relations in the ontology, it is possible to find ontology entities which could also be interesting for the actual query. Using the temporal and spatial information in the ontology, it is even possible to consider the actual context of the query and use only relations that are relevant for the actual context. For example, if the query contains Napoleon and St. Helena, we should not use the relation connecting Napoleon with Russia[8]. The query expansion can happen automatically but it is also possible to generate only suggestions for the user for possible expansions of the query. In this case, the technique can be considered as a kind of query formulation support.

Similarity measure: It is easy to see the potential of an ontology-based similarity measure. Compared to syntax-based similarity measures, ontology-based measures could exploit the ontology structure and thus measure semantic similarity. E.g., the term list (“Lenin”, “Winter Palace”, “1917”) has no syntactic similarity to the term list (“Vladimir Ilyich Ulyanov”, “Russian Revolution”) but the two lists are semantically similar[9].

[7]For example, Napoleon had nothing to do with Russia in the early years of his career.

[8]Napoleon was exiled to the island St. Helena in 1815 and he died there in 1821. He invaded Russia years before, in 1812.

[9]The attack on the Czar’s Winter Palace in St. Petersburg in 1917 was one of the most important events of the Russian Revolution. Vladimir Ilyich Ulyanov is the birth name of Lenin, the leader of the revolution.

Indexing: If we have an ontology, it is useful to create semantic annotations as (part of) document representations. This allows us to match documents with queries semantically, instead of syntactically. E.g., while the term “Napoleon” is ambiguous, the ontology URI `#Napoleon_I` clearly identifies Napoleon Bonaparte and avoids returning documents for example of his son, Napoleon III.

In addition to using ontology elements as semantic annotations, we can use the knowledge stored in the ontology to identify potentially interesting elements to include into the semantic annotation, which are not explicitly mentioned in the text. E.g., if many events, locations and politicians related to the Kosovo Conflict appear in a document and also the time context is that of the Kosovo Conflict, we can safely assume that also the Kosovo Conflict itself is relevant for the document. Thus, it should be added to the semantic annotation, even if the phrase “Kosovo Conflict” does not appear in the document.

Basically it is the same idea that we described at query expansion: we use the knowledge stored in the ontology to find relevant elements in the ontology to a specific set of already known ontology elements, *in context*. In contrast to classical thesauri, ontologies allow us to represent relations at the semantic level and to represent spatial and temporal context information of specific relations. In many application domains, including history, spatial and temporal context is very important and they are good indications whether a potential relation between ontology entities is valid for a specific document, or for a specific query.

Visualization: If we have semantic annotations associated with the documents, we can exploit them to display additional relevant information to a document. This makes it easier for users to comprehend the content of the document. The already introduced `eurohistory.net` portal is a good example for this approach. As was discussed, in the `eurohistory.net` portal the spatial and temporal context of the document was visualized by displaying a historical map relevant for the temporal context of the document. The most important countries in the spatial context were colored on the map[10] (see also Figure 3.1).

It is important to see that semantic metadata alone is usually not enough for visualization purposes but the ontology itself has to be used, too. In our application example, because visualization happens at the country level but semantic metadata can be also at the settlement or region levels, additional inferencing is needed to determine the country (or countries) of specific settlements and regions. E.g, if BERLIN is part of the semantic annotation and the temporal context is the early 19th century, it will be inferred that PRUSSIA should be colored on the historical map. In the early 21th century the relevant country would be GERMANY, of course.

Visualization is a huge research field (see e.g., Chapter 10 of [BR99]) and a thorough examination of this field is out of scope of this thesis. Therefore, it is mentioned here only for the sake of completeness.

Navigation: As we discussed, a typical information retrieval process is an iterative, evolving process. In other words, users usually submit many queries to collect all of the information pieces that together satisfy their (possibly changing) information needs. In addition to context

[10]Such maps are called choropleth maps

visualization, also navigational elements can be generated automatically. They support users in reformulating their queries, based on the semantic metadata and information stored in the ontology. E.g., in the `eurohistory.net` portal visualized elements were interactive. The user could click on any of the visualized elements (such as on parts of the colorized maps), which initiated a query that returned all resources about that element in the context of the actual document[11]. E.g., if the actual document was about Napoleon and the temporal context of the document was the early 19th century, the user was able to navigate to resources about Russia in the context of Napoleon and the 19th century simply by clicking on Russia on the historical map.

3.5 Imperfection in background knowledge

As even thesauri solve many of the problems of full-text based systems, one would intuitively expect that using thesauri significantly improves search effectiveness. As ontologies are even more powerful, this assumption also applies to them naturally. Therefore, ontology-based information systems are automatically accepted as superior to classical, syntax-based systems by many researchers. Unfortunately, experience with thesauri shows that this assumption is usually not true because of the imperfection of thesauri [Sal86].

Because our intuition that using background knowledge will automatically increase retrieval effectiveness does not seem to be correct, it is a natural requirement that systems that claim to achieve a gain in retrieval performance, should be evaluated.

Requirement (Evaluation)**.** *The claim that an information system using background knowledge increases IR effectiveness cannot be accepted intuitively but must be evaluated.*

3.5.1 Imperfect ontologies

One possible major cause for this failure in the case of thesauri is the "noise" of thesaurus relations between thesaurus terms. As was mentioned before, linguistic relations, such as synonyms, are normally valid only between specific meanings of two words. Thesauri, however, represent those relations as generally valid between the syntactic form of words. E.g., while "baby" and "infant" are synonymous in one specific context, the word "baby" has many other meanings where it is not synonymous with "infant"[12]. Therefore, representing these words as synonyms in a thesaurus would not be correct in all situations. This deficiency of thesauri usually results in false positives in the search result.

Another, and perhaps the bigger problem is that the manual creation of thesauri and the annotation of documents with thesaurus terms is very expensive. Moreover, automatic creation of high-quality thesauri fails because term co-occurrence, which is used by most statistical methods to measure the strength of the semantic relation between words, is not valid from a linguistic-semantical point of view [Kur05]. I.e., if two words appear together frequently, it

[11]It was assumed that the context of the document can be estimated by considering the semantic metadata of the document.

[12]Such as when it is used to denote someone's girlfriend.

does not necessary mean that there is a strong relation between them. On the contrary, if two words do not appear together frequently, it does not mean that there is not semantic relation between them. E.g., [Kur05] found based on the analysis of the German and English Wikipedia that members of word groups, such as “New York”, “Albert Einstein” or “hard drive”, appear the most frequently together. In this case, however, there is no semantic relation between the words of the word group. I.e., one cannot say that the word “new” is semantically related to the word “york”, or the word “hard” is semantically related to the word “drive” and therefore there should be some relation between them in a thesaurus. On the other hand, in the case of words that did have semantic relations, such as synonymy[13], hyponymy[14] or meronymy[15], the co-occurence measure was not significantly higher than in the case of completely unrelated words.

As a result, thesauri and annotations using them are often incomplete or erroneous, resulting in decreased search performance.

As was discussed (see Section 3.4), ontologies have many advantages in comparison to classical thesauri and thus they have a bigger potential to improve IR results than thesauri. However, ontologies (and semantic annotations using them) also suffer from the high costs of manual creation, similarly to thesauri. Moreover, automatic creation of ontologies and semantic metadata is even a more challenging task than the automatic creation of thesauri. Because of this, ontologies and semantic annotations are hardly ever perfect. Indeed, currently good quality ontologies and semantic annotations are a very scarce resource. This claim is based on both personal experiences during the VICODI project [NDO05] and on our analysis of available ontologies and metadata on the present Web[16].

During the VICODI project, a comprehensive ontology of European history was developed. Although the `eurohistory.net` portal, which was the result of VICODI, showed some of the potentials of an ontology-based information system, the quality of the results were plagued by the lack of proper ontological information, due to the prohibitive cost of developing an ontology fully covering such a wide domain.

The main lesson learned from the project is that it is very hard to switch from present full-text based information systems to semantic based ones in one big step because the costs of a high-quality, comprehensive ontology that would cover[17] the whole content of the system are prohibitive. Rather a gradual approach is needed, which combines the merits of statistical and ontological approaches and thus provides a smooth transition between the two worlds. In such an approach it would not be required to build a perfect ontology covering the whole domain in one step.

In addition to the costs of ontology creation, another cause for ontology imperfection is the limited expressive power of ontology formalisms. Although they are much more powerful than thesauri, there are still many important aspects that cannot be modeled in present-day

[13]words with (almost) same meaning, such as “doctor” and “physician”

[14]words having an “is a” relationship, such as “car” and “BMW”

[15]words having a “part of” relationship, such as “car” and “wheel”

[16]E.g., `http://www.daml.org/ontologies/`, `http://ontolingua.nici.kun.nl:5915/` and `http://protege.cim3.net/cgi-bin/wiki.pl?ProtegeOntologiesLibrary`

[17]Under “ontology coverage” I mean the situation when an ontology contains the necessary elements to describe the relevant entities of a document. In this case I say that the ontology “covers” the topic that is described in the document, or simply the ontology “covers” the document.

ontology languages[18]. Therefore, imperfection in ontologies and metadata should probably be considered even in the long run, as the expressive power of ontologies cannot be significantly raised without losing decidability of ontology reasoning.

Requirement (Ontology imperfection)**.** *An information system should tolerate the imperfection of ontologies and semantic metadata.*

3.5.2 Domain imperfection

Even if there were enough resources in a project to build the perfect ontology and to semantically annotate the resources without errors, in some cases our knowledge of the domain is simply imperfect. I term this type of imperfection as *domain imperfection*. If we want to build an ontology which faithfully reflects our domain knowledge, domain imperfection should also be represented. If we encoded our imperfect knowledge as if it was perfect, we would clearly distort our domain knowledge. This could potentially result in reduced retrieval performance, at the end.

Requirement (Domain imperfection)**.** *An ontology should explicitly represent imperfection of domain knowledge.*

According to Smets [Sme96], the two main types of imperfection are imprecision and uncertainty. *Imprecision* is related to the content of a statement: more than one world is compatible with the information. *Uncertainty*, on the other hand, results from a lack of information about the world, so that we cannot decide whether a statement is true or false. In other words, imprecision is a property of the information itself, while uncertainty is a property of the relation between the information and our knowledge about the world.

In our application scenario of history, domain imperfection appears mainly in the time dimension. Therefore, various types of temporal imperfection are reviewed below to motivate the need for representing imperfect information in ontologies and also to identify the requirements for a temporal model.

It is important to note that time modeling is a crucial feature in many other application domains, not only in history. Examples are medicine, criminal and financial information systems. Moreover, every system that deals with news is inherently time dependent. After all, what is news today, is history tomorrow. The importance of time modeling is shown by the numerous works in the area of temporal databases [JDB$^+$98, EJS98] and temporal reasoning [Vil94].

3.6 Time modeling in history

It is quite obvious that time modeling is a fundamental issue for modeling historical information, since almost every historical statement is time dependent. Additionally, we identified several specific features during the VICODI project that make capturing this information a challenge. For one, time information in history is often uncertain or ill-defined. It is usually

[18]Such as belief, uncertainty, gradual truth values. In many cases also the form of logical axioms are limited.

extracted from historical documents written in an imprecise and inherently *vague style*. Even worse, important documents are often missing or contain contradictory information, so temporal information about historical events is *uncertain*. Apart from the uncertainty of temporal information, historical events are often abstract, so their definition is inherently *subjective*. For example, we have found it impossible to precisely define the time span of the "Middle Ages".

Imperfection of historical temporal specifications is also shown by specifications such as "? - 1640" (meaning that the beginning time of the event is unknown) or "ca. 1801" (meaning circa, i.e., around this year).

I review now the unique features of historical temporal specifications in more detail.

3.6.1 Uncertainty

Sometimes information about a historical event can only be deduced from documents reporting about related events. Often several documents state contradictory facts about some event. As an example consider Stalin's birth date. Officially for the USSR it is 1879-12-12[19] but according to church records his birth was registered on 1878-12-06. Historians are in disagreement over which time specification is the right one.

In such a case we say that the temporal specification of the event is *uncertain*. Temporal uncertainty belongs to Smets' uncertainty category of imperfection.

3.6.2 Subjectivity

Many historical events are not exactly defined but are *subjective*. For example, "Early Renaissance", "Russian Revolution" or "Industrial Revolution" do not have a clear definition, so it is impossible to clearly state exactly when these events occurred. The temporal extent of these abstract events sometimes also depends on the cultural background of the domain expert. E.g., the "Second World War" or the "Middle Ages" mean something different for experts from Germany and Japan.

In this case it is intuitive for historians to talk about "beginning or end phases", "process, development and core periods" or "transition periods", which clearly indicates that the traditional model of having temporal intervals with definite start and end points does not meet the reality in this case.

Temporal subjectivity belongs to Smets' imprecision category of imperfection.

[19]For the sake of consistency and simplicity, I use the language independent ISO 8601 date format in this thesis. This format has the YYYY-MM-DD pattern, where YYYY denotes the year, MM denotes the month as number, and DD denotes the day. See e.g., [WW97] for more information.

3.6.3 Vagueness

Historical time specifications are given at different granularity (years, months, days) and are often defined *vaguely* (early morning, spring etc.). Hence, the temporal specification is not known precisely but is vague.

The reader may also note that any temporal specification made in a natural language will become vague if we refine the granularity of temporal axis sufficiently.

Temporal vagueness belongs to Smets' imprecision category of imperfection, like temporal subjectivity.

3.6.4 Combined aspects

There are also events exhibiting a combination of the aforementioned aspects of imperfection, so the temporal model should be capable of representing all of them in a unified manner. E.g., in the statement "The Cold War ended in the late eighties" both subjectivity (Cold War) and vagueness (late eighties) are present.

Requirement (Unified representation of temporal imperfection)**.** *A temporal model should represent the uncertainty, vagueness and subjectivity aspects of imperfection in a unified framework.*

3.7 User Friendliness

An ontology-based system usually provides the possibility to formulate highly complex, ontology-based queries which are very similar to full-fledged database queries, expressed in SQL. These kinds of queries are seen as superior by many researchers to simple, full-text queries. Intuitively, this claim seems plausible. While full-text query engines usually process queries only as a bag of words, it is possible to express very complex relationships between the query concepts with ontological queries. E.g., it is possible to express the following query: "All battles of the Second World War in France where Charles de Gaulle participated". A full-text search engine would see only the words "battle", "Second World War" and "Charles de Gaulle" and it is impossible to represent the relations between the query concepts.

However, as mentioned in Section 2.2, it is a high cognitive load for users to formulate their inherently imprecise and uncertain information needs in some explicit form. This cognitive load is further increased if a system forces users to formulate a database-like structured, complex query. Most casual users are not able to formulate SQL queries and similarly they cannot formulate ontology queries, either. Although the process of query formulation can be supported by various tools and visualization, it is still much slower and more complicated than simply typing some keywords in a text area. This was also our experience during the VICODI project: our users wanted to search like they got used to search in Google — by simply typing some of the keywords.

Requirement (Natural language query). *An information system should provide the possibility for its users to start with a simple, natural language query.*

After the user got the first results from the information system, new and innovative ontology-based user interface techniques can be used to support the further browsing in the repository, or query reformulation [Sto05]. The starting point, however, should be always easy and natural for the users, otherwise they will refuse using the system. Of course, for power users, such as librarians or researchers, an advanced, ontology-based query interface can be provided additionally, if needed.

3.8 Scalability and Interactivity

3.8.1 The importance of efficiency

The main concern of information retrieval in general, and in this thesis in particular, is the *effectiveness* of IR, i.e., the quality of results. However, *efficiency*, i.e., the speed of the system retrieving the results, should not be neglected, either. E.g., in the VICODI portal a very important lesson learned was that if users are unsatisfied with the speed of the system, they do not care about the quality of the results but leave the system and search for another, faster one.

In the case of the `eurohistory.net` portal, users were especially unhappy about the email notification system we used in the case of time intensive tasks[20]. This all shows that nowadays most users expect information systems that they can use interactively. Therefore, during this thesis I consider only interactive information systems and assume that for an interactive application the maximum response time users are ready to tolerate is *10 seconds*.

It is important to stress, however, that there are some processes of an information system which need not to be interactive but can be processed offline. Such examples include in most cases the automatic creation of metadata and the indexing of documents for full-text search.

An ontology-based information system should be scalable both in terms of ontology size and the size of the document repository. For scalability, the latter is critical. Even in the Intranet of a bigger company there are potentially millions of documents, which should be handled by an information system. If we consider the whole Web, scalability is an even bigger challenge. E.g., the index of the Google search engine contains over 8 billion documents[21]. In this thesis I do not address the issue of a web search engine but the solution should be usable in a typical company Intranet.

Requirement (Scalability). *An information system should execute user queries interactively, i.e., the response time should not be higher than 10 seconds. The system should be interactive even with millions of documents in its repository.*

[20]The system did not provide the result interactively but sent an email notification to the user when her or his request was processed and the user could return to a specific web page to view the results.

[21]Source: `http://de.wikipedia.org/wiki/Google`

3.8.2 Efficiency problems of ontology reasoning

Many of the existing ontology-based information systems foresee a central ontology management framework and most of the tasks in the information system are accomplished using ontology reasoning or ontology navigation services provided by this framework [MSSV02, HMSS01, VFC05]. These systems represent everything in the ontology: the background domain knowledge, the documents (resources), the document annotations, and the lexical information of the ontology elements[22].

This was also the case at the beginning of the development at the `eurohistory.net` portal. We planned to use the KAON ontology management system [MMV02] as the basic infrastructure for the portal, which provides many features for ontology navigation, management, multilingual labels, and light-weight reasoning. It turned out, however, that for some important tasks the KAON system did not provide support, or its performance was not adequate.

In an ontology-based system it is a challenge to reach the required level of scalability because ontology-reasoning is theoretically highly complex and it is usually not tractable[23]. Moreover, as ontologies are still a relatively young technology, existing reasoners and ontology-management systems are still immature and do not use all of the caching and indexing technologies that are available for more established technologies, such as relational databases [24].

The fundamental problem using ontology reasoning for information retrieval is, however, the lack of ranking support in ontology reasoning. Ontology queries yield a set of new facts, without any ranking — similarly to results of database queries. Although it is possible to calculate some ranking score after the results were retrieved, this approach does not scale, if there are lots of intermediate results.

3.8.3 Experimental demonstration of the efficiency problems

To demonstrate the problems of using ontology reasoning for IR, I did some experiments by running the typical task of full-text search on documents represented in the well-known vector space model (see Section 2.3 and [BR99, pp. 27–30]). I used the PostgreSQL database[25] on one hand, with a highly optimized, simple schema to represent terms of documents according to the vector space model. The schema was motivated by the relational implementation of the Topic-based Vector Space Model (TVSM) [Kur04] and is shown in Appendix A.

My motivation to use a relational database (RDBMS) as one of the test systems instead of an ontology reasoner was that most of the current ontology management frameworks use relational databases as their underlying infrastructure[26]. I.e., some ontology reasoning steps are

[22]Such as labels in the natural languages supported by the system, where there can be optionally many labels for a language, representing all the synonymous descriptions of the abstract ontology entity.

[23]Sometimes not even decidable.

[24]Although caching of results is already used in some systems, such as Ontobroker (`http://www.ontoprise.de`) or Pellet (`http://www.mindswap.org/2003/pellet/`)

[25]`http://www.postgresql.org/`

[26]Some other reasoners support only in-memory ontologies. However, such reasoners cannot handle large ontologies that do not fit in the main memory of the computer and therefore I do not consider those as a viable alternative.

translated to SQL queries on databases; others that cannot be expressed as database operations are calculated in memory on the data returned by the SQL queries. Therefore, using a bare RDBMS actually simulates a perfect ontology reasoner that executes all of the ontology reasoning tasks that are not expressible as SQL queries in zero time. In other words, during the test I eliminated the possible negative effect of the yet immature implementations of ontology reasoners. I could do this simplification because already the bare database technology exhibits the main problem of ontology reasoning for IR, i.e., that ranking must be executed after retrieving intermediate search results. To simulate full-text search in a relational database, first documents that contain at least one search term are returned and later the ranking scores are calculated on-the-fly by the SQL query. The first step can be executed very efficiently because database indexes on term and document ids can be used. If there are too many documents, however, which satisfy this first condition, it is extremely expensive to keep all of the results in the main memory and calculate the VSM scores afterward. To summarize the discussion, with the following test I actually show that this deficiency alone is enough to render using RDBMS and thus also ontology reasoning infeasible for IR on big document collections.

The other system participating in the test was my own full-text search engine based on the Lucene framework[27]. Lucene in an open-source Java library which makes it possible to implement simple full-text search engines very easily. Lucene does not use any relational database for storing the full-text index but it defines its own, highly optimized, file based index structure. Its main feature in this context is that the calculation of ranking scores happens during query execution.

During the test I run a full-text search on documents and both systems returned a ranked list of relevant results. I was mostly interested in how the systems scale in terms of document size (number of terms in a document), query size, and number of documents in the repository. I executed the tests using a local PostgreSQL database installation and a local Lucene-based Java search engine on the same machine and measured the execution time. The system I used was a Pentium IV 2.8 GHz PC with 1 GB RAM. As I used a multi-tasking operating system[28] for the tests, the exact execution time values are most probably not precise but the tendency and the order of magnitude of the values should be significant. The results of the experiments are shown from Table 3.1 to Table 3.4. The response time values that violate the SCALABILITY requirement are marked.

As one can expect the response times are increased when the query size (Table 3.1), the repository size (Table 3.2), and the average document size (Table 3.4) grows. Concerning Table 3.3, it is important to see that if the number of used terms in the whole repository increases, the chance that a specific term appears in many documents decreases. This means, for the same repository, and for the same query and document sizes, the more terms are used in the repository, the fewer results will be retrieved. Clearly, less results can be retrieved faster. Therefore, in this case the response times decrease as the number of terms increases[29].

As can be seen from the tables, PostgreSQL response times violate the SCALABILITY requirement in many cases, even for moderate size repositories. On the other hand, Lucene had no problems even when dealing with very big repositories.

[27]`http://lucene.apache.org`

[28]Windows XP Professional

[29]with a small exception on the Lucene side, where the response times are so small that they can be actually considered as constant

Table 3.1: Scalability in terms of query size

Average response time (ms)		
No. of query terms	*PostgreSQL*	*Lucene*
5	1693	94
10	3047	125
50	285517	458
100	267274	620
500	331264	3354
1000	401251	5355

Constants	
No. of docs	100000
No. of terms	10000
No. of terms/docs (max.)	1000

Table 3.2: Scalability in terms of repository size

Average response time (ms)		
No. of docs	*PostgreSQL*	*Lucene*
100	73	172
1000	198	177
10000	1511	183
100000	17371	240
1000000	187244	865

Constants	
No. of query terms	100
No. of terms	10000
No. of terms/docs (max.)	100

Table 3.3: Scalability in terms of term frequency

Average response time (ms)		
No. of terms	*PostgreSQL*	*Lucene*
1000	2177	213
10000	1526	187
100000	99	177
1000000	94	203

Constants	
No. of query terms	100
No. of docs	10000
No. of terms/docs (max.)	100

Table 3.4: Scalability in terms of document size

Average response time (ms)		
Max. no. of terms/doc	*PostgreSQL*	*Lucene*
10	266	156
100	1521	193
1000	21742	213
10000	133115	479

Constants	
No. of query terms	100
No. of terms	10000
No. of documents	10000

This small experiment demonstrates that using the wrong technology for a specific task can have catastrophic consequences when the repository size grows. Some of the ontology-based systems that use ontology reasoning for retrieval do not experience these problems because they work with a relatively small ontology, and with a small to moderate size of documents. However, during the VICODI project, where we had to work with a big ontology, and a real-size document repository, we experienced serious performance problems.

3.8.4 Combining ontology reasoning with other technologies

Based on the experimental results presented here, and our experiences in the VICODI project, I can state that a system using exclusively ontology reasoning for all of its task including information retrieval will not scale well in terms of the repository size.

A possible solution for this problem is if an ontology-based information system does not use ontology reasoning for retrieval exclusively but combines various technologies (including ontology reasoning) to achieve good-quality results and to meet the scalability constraints. This allows the system to use mature, highly optimized techniques to deliver specific standard information system services and use ontology reasoning only at places when it is suitable.

3.9 Feasibility

Ontology-based information systems always need an ontology and semantic annotations using this ontology to be able to operate. As was discussed, it is very expensive to develop an ontology manually. Still, it is possible to achieve because the interesting concepts and their relations are limited in a specific domain. Moreover, there are some tools and systems in the areas of ontology learning, such as Text2Onto [CV05], KIM [KPT$^+$05], or OntoGen [FMG05], which support users populating an ontology with new instances, or finding interesting concepts and relations in a domain. These tools usually work by examining syntactic patterns in texts of a document collection but they do not "understand" the text. Therefore, they can only support manual ontology creation but cannot replace humans completely in the ontology engineering process.

It is even more expensive than building ontologies, however, to manually annotate millions of documents based on the ontology. While some very big companies can afford to pay full-time employees in some limited, mission-critical domains to manually annotate documents, this is not an option for most of the small and medium sized companies. Even the biggest companies cannot afford to manually annotate *all* of the documents on their Intranet.

One of the biggest success factors of the classical full-text search was the fact that existing document collections can be indexed completely automatically. Every approach that is feasible in real life should also operate completely automatically, with the option, that the automatically generated metadata can be reviewed and corrected manually, if needed. It is important, however, that the system outperforms full-text search engines even in the case when it uses only completely automatically generated (i.e., non-validated) metadata, otherwise it is very hard to show the utility of ontology-based approaches.

Requirement (Metadata generation)**.** *An ontology-based information system should generate the metadata needed for its operation completely automatically. Although manual correction of metadata is allowed, the system should be evaluated using the automatically generated metadata.*

3.10 Summary

In this chapter I analyzed the causes why traditional full-text based search engines fail to meet user expectations for research types of queries. I used the `eurohistory.net` web portal as a motivating scenario for the discussion, together with its application domain, i.e., history. We have seen that theoretically thesauri and ontologies can solve most of the identified problems and therefore it could be intuitively expected that using these artifacts should improve IR results. Still, existing IR experience with thesauri shows that this is not necessarily the case. I identified errors and missing information in thesauri and ontologies as the main possible cause for this phenomenon on the one hand; and the missing capabilities of ontology formalisms to precisely represent imperfect domain knowledge, on the other hand.

Later, I reviewed historical temporal specifications as a typical example for domain imperfection. I identified three aspects, namely uncertainty, vagueness and subjectivity, which should be handled by a formalism suitable for representing imperfect temporal information.

Finally, I showed the importance of scalability and interactivity in ontology-based information systems and identified automatic metadata generation as an important criterion for the feasibility of any ontology-based information system.

For easier reference, a full list of identified requirements is shown in Figure 3.2.

NL vagueness: An information system should handle the vagueness of natural language.

Semantic relations: An information system should exploit semantic relations to find also relevant concepts that are not mentioned explicitly in the document text.

Time dimension: An information system should support user queries concerning the time dimension.

Evaluation: The claim that an information system using background knowledge increases IR effectiveness cannot be accepted intuitively, but must be evaluated.

Ontology imperfection: An information system should tolerate the imperfection of ontologies and semantic metadata.

Domain imperfection: An ontology should explicitly represent imperfection of domain knowledge.

Unified representation of temporal imperfection: A temporal model should represent the uncertainty, vagueness and subjectivity aspects of imperfection in a unified framework.

Natural language query: An information system should provide the possibility for its users to start with a simple, natural language query.

Scalability: An information system should execute user queries interactively, i.e. the response time should not be higher than 10 seconds. The system should be interactive even with millions of documents in its repository.

Metadata generation: AAn ontology-based information system should generate the metadata needed for its operation completely automatically. Although manual correction of metadata is allowed, the system should be evaluated using the automatically generated metadata.

Figure 3.2: Identified requirements

Chapter 4

State of the Art

In this chapter, I review the state-of-the art of ontology-based information systems based on the requirements identified in the previous chapter. Currently ontology-based information systems can be categorized into two groups.

Systems that focus on retrieving instances of a given ontology belong to the first group. These systems are mainly useful in knowledge management where the users would like to access the knowledge stored in the form of an ontology. Documents can also be used in such systems but in this case the “one document – one ontology instance” assumption is used. Of course, this assumption is valid only in some special domains, like in the case of web pages of a university department, where one page describes a staff member of a university, or a research topic. Although in our application domain this assumption does not hold, some of the problems are also present in our domain, and some solutions can be reused.

The other group contains information systems that focus on document retrieval. In these systems, documents are annotated with many ontology instances, i.e., they do not have the “one document – one ontology instance” assumption any more. This is also the case in our application domain.

It is important to note that in the first group it does not make sense to consider ontology imperfection because users browse the ontology itself. Therefore, if an ontology instance is missing (or erroneous), there is no chance to notice or correct it. Moreover, semantic metadata generation does not make sense, either, as semantic metadata is not needed in this case.

4.1 Systems focusing on ontology instance retrieval

4.1.1 QuizRDF

QuizRDF [DW04] exploits RDFS [BG04] annotations attached to documents describing exactly one instance from an ontology. It is also assumed that one ontology instance has exactly one direct class in the RDFS schema. The indexer of the system indexes both RDF [MM04] triples (i.e., semantic metadata) and the full-text of the document. The information model for document representation consists of tuples in the form:

```
<text, class, property, doc_URL>
```

An example for a tuple of an RDF statement is:

```
<"Smith", Employee, hasLastName, http://company.org/doc1>
```

representing the fact that the textual value of the hasLastName property of the document with the URL `http://company.org/doc1` is “Smith”. Only such RDF statements are indexed where an RDF literal (i.e., a string) is involved in the statement.

It does not make sense to talk about properties in the full-text part, therefore the full-text part is indexed using tuples with empty properties, such as:

```
<"John Smith", Employee, ∅, http://company.org/doc1>
```

The user can search the text part (first element) of the tuples using traditional full-text search. The results are the documents which are denoted by the URLs in the tuples. Ranking is based on the usual TF-IDF scheme [SB88].

The results can be filtered based on the class information. E.g., the user can search for “Smith” and after that she or he can filter the results to show only employees.

It is also possible to browse the ontology by clicking on the class name, or by following the properties of the class which connects it with other classes. E.g., it is possible to navigate from employees to their skills or projects.

In contrast to simple full-text search, after selecting a class, it is also possible to start a structured query, by filling the textual properties of the class by the desired textual phrases. E.g., it is possible to search for employees with the last name “Smith”.

Discussion

QuizRDF shows a simple approach to combine full-text search with semantic metadata. It demonstrates that ontological information can be encoded so that it is compatible with full-text search. This solution is clearly scalable.

This proposal, however, meets only few of the requirements from Chapter 3. It does not make use of ontology relations during the search, it only provides the possibility to browse the related instances after the search results returned. It also ignores the issues of domain imperfection, and does not make use of temporal information. Moreover, in [DW04] no evaluation of the system is presented.

4.1.2 SEAL Portals

The SEAL I and II portal systems [HMSS01, MSSV02] use a central ontology to provide semantic search and browsing functions. They build on the Ontobroker reasoning engine[1], and implement instance retrieval as ontology query answering. The system was deployed in real-world applications, among others the university portal of the authors was driven by the system. In this portal, users can search for all employees who do research in a specific research topic, or they can list all publications that a specific employee has written.

The biggest problem with using classical ontology query answering for information retrieval is that the results are not ranked (see also Section 3.8.2). Therefore, a novel approach was needed in these portals to rank the results which were provided by the inference engine. A very sophisticated approach was proposed in [SSS03]. Here both the structure of the ontology, and the reasoning structure are considered to find instances that "better" match the initial ontological query than others. The ontology structure is exploited by examining the specificity of the relations between the instances[2]. The more specific a relation, the higher the ranking score given to it. To exploit the reasoning structure, the reasoning tree is extracted from the reasoning engine, and results that are inferred in fewer steps get a higher ranking score.

The ranking algorithm was evaluated in a small-scale user study [Sto05], and was found superior to unranked ontology reasoning results.

Discussion

The SEAL system shows that it is possible to implement information retrieval based on pure ontology query answering. It is questionable, however, how the solution would scale for extensive ontologies, e.g., when there are many thousands or tens of thousands of instances. First, all of the results must be returned which match the query and after that a complex ranking algorithm is applied on these results. If the results that the reasoner provides are numerous, it is expected that the performance of the system will not be adequate.

On the other hand, in many domains the size of the ontology is moderate. E.g., in a typical university department ontology there are not so many instances. Therefore, in these application domains this approach provides a feasible solution that can potentially return very high-quality results.

The ranking algorithm can be problematic in real-world situations, too. While the Ontobroker system is able to provide a reasoning tree, in general it is a very rare feature for a reasoner. Therefore, this technique cannot be used in most ontology-based systems.

This system ignores domain imperfection, and does not exploit temporal information. It does not support natural language queries either.

[1] Ontobroker is a product of the Ontoprise GmbH (`http://www.ontoprise.de`).

[2] i.e., the number of instances that are connected to a specific instance through the same relation

4.1.3 The system of Zhang et al.

An interesting approach is proposed by Zhang et al. [ZYZ$^+$05]. They propose to use reasoning in a fuzzy description logic (DL) ontology to implement semantic information retrieval. In fuzzy DL, statements take the form $i : C \geq \alpha$ denoting the fact that the instance i belongs to concept C with at least the truth value α. Based on such an ontology, a fuzzy DL reasoner can return a list of instances for a query $j : D \geq \beta$, which belong to the concept D with at least the value β.

The main idea in the proposed system is to represent search results of classical full-text search engines as fuzzy DL statements in the ontology. For each document in the repository, an instance[3] is inserted into the ontology. For a full-text query Q, statements in the form $i : D_q \geq \sigma$ are inserted for document instances i, where σ is the score returned by the full-text search engine. D_q is a proxy concept representing all the documents which are relevant to the query Q. The $i : D_q \geq \sigma$ statement expresses the fact that the truth value of “document i is relevant to query Q” is at least σ.

After integrating the results of the full-text query into the ontology, arbitrary fuzzy DL queries can be used to retrieve information from the ontology. E.g., we could ask for people, whose name contain “Smith” and gave a presentation, where at least one of the slides was about the topic “ontology”. In this case, the two full-text queries “Smith” and “ontology” are executed using a traditional full-text engine, and the results are integrated into the ontology using the proxy concepts D_{Smith} and $D_{ontology}$. After that, the query can be answered using fuzzy DL reasoning.

Discussion

The presented approach provides a powerful way to combine results of ontology reasoning and traditional full-text search. Zhang et al. did some experiments with a small knowledge base, and could show that their approach provides better results than running simple queries containing only purely fuzzy DL and purely IR concepts[4]. Moreover, using fuzzy DL has the potential to represent domain imperfection in the ontology, although this option was not discussed in the paper.

Although this system has the potential to answer very complex queries, it does not scale well for big document collections. First of all, the first, the full-text search phase is executed separately, potentially yielding lots of results. These results have to be added to the ontology (yielding potentially a very big ontology), and fuzzy DL reasoning has to be used to retrieve the results. Fuzzy DL reasoning is theoretically highly complex, therefore it does not scale up to ontologies with many elements.

This approach does not exploit temporal information.

It is important to note that although this system retrieves ontology instances, it partially also belongs to the second group of systems because some of the ontology instances represent

[3]Instances are termed as *individuals* in DL terminology but we use the word instance to remain consistent with other parts of the thesis.

[4]Such queries could be executed by a separate search engine and a fuzzy DL reasoner, respectively.

documents. Therefore, it can also be used to retrieve documents, like documents containing the word “ontology”, and having an author with the first name “John”.

4.1.4 The system of Rocha et al.

Rocha and his colleagues propose a system for retrieving documents, describing a specific instance in the ontology [RSA04]. They assume that ontological queries are too complicated for most of the users, and therefore plain text queries have to be supported to get relevant ontology instances[5]. Therefore, the users start with a plain text query, and the system returns a list of relevant ontology instances to the query, better to say the documents describing those instances.

To achieve this goal, they use full-text search on a textual representation of ontology instances. These initial results are refined later by a spread activation procedure which exploits the ontology to find other relevant ontology instances, which are syntactically not similar to the original, textual query.

Generally, a spread activation algorithm propagates weights in a graph between nodes. In this case, the ontology is viewed as a graph, instances being the nodes, and relations between instances being the edges. The initial weights of the nodes are the weights provided by the full-text search. These weights are propagated later between a source instance i and a target instance j using the following formula:

$$w_j = w_i * w_{ij} * f_{ij} * (1 - \alpha) \tag{4.1}$$

where w_j denotes the weight of the target instance after propagation, w_i denotes the weight of the source instance, w_{ij} denotes a dynamically calculated weight of the relation between the instances, and f_{ij} and α are constant parameters of the algorithm. f_{ij} defines the importance of a specific type of relation f. Such a way, one can for example express that the LEADS property is more important than the MEMBEROF property. The α parameter specifies a weakening factor.

The w_{ij} weight of a relation is calculated using the same basic ideas already introduced at the SEAL portal approach. In this case, a so-called cluster measure and a specificity measure are combined to get the w_{ij} weight. The former checks the percentage of common neighbors of the two instances, building on the assumption that instances that have many common neighbors are similar. The latter defines a measure whose value decreases for relations which connect many instances in the ontology. The termination of the propagation algorithm is ensured by simply executing it only once.

The proposed approach was implemented in two systems, and user evaluations were conducted. For two example queries the users were asked to validate whether the proposed instances are really relevant to the query. The results were very convincing, the rate of relevant instances were above 90% for all of the tests. Moreover, sometimes surprising, relevant instances were proposed that could not have been found using full-text search.

[5]This assumption is formulated in our NATURAL LANGUAGE QUERY requirement, as well.

Discussion

Rocha et al.'s approach clearly shows the value of using background knowledge stored in an ontology during information retrieval. Although the complexity of the spread activation algorithm would be prohibitive for document retrieval, it provides acceptable performance for ontology instance retrieval. Their user evaluation shows that ontology-based systems can provide higher quality results than traditional full-text search.

This system does not exploit temporal information, and ignores the issues of domain imperfection.

4.2 Systems focusing on document retrieval

4.2.1 KIM

The KIM (Knowledge and Information Management) platform [KPT$^+$05, PKO$^+$04] provides an integrated framework, where all aspects of ontology-based information systems are supported. These include (semi-)automatic ontology population, (semi-)automatic creation of metadata, and ontological querying.

The system uses the GATE natural language processing (NLP) framework[6] to automatically extract new ontology instances and metadata from documents. Only the labels of existing ontology instances together with the concept taxonomy of the ontology are used in various heuristics during the metadata extraction phase. The semantic annotation, which is the result of the metadata extraction process, connects a specific part of the document (a word or a phrase) with an ontology instance.

Before the indexing phase, KIM extracts the ontology instances from the semantic annotations, and adds their unique identifiers to the document text. Later, during indexing, these ontology instance identifiers are also stored in the full-text index. This makes it possible to search for documents later using ontology instance identifiers, instead of just using the natural language label of the ontology instance. This practically eliminates the vagueness of natural language because instance identifiers are unique in the ontology, in contrast to natural language labels that can be connected to many ontology instances.

KIM requires the user to formulate a classical ontological query. It first filters the ontology instances using ontology inference, and then uses the full-text index to retrieve documents which refer to those ontology instances.

Discussion

KIM provides a holistic ontology-based information system that supports automatic metadata extraction, and even automatic ontology population – an issue that is out of scope of this thesis. The system scales well because simple full-text search is used to access the documents, and

[6]`http://gate.ac.uk/`

ontology reasoning is used only to identify relevant ontology instances based on the user query, i.e., to transform the ontology-based query to a full-text query. The example of KIM shows again that with the proper encoding of semantic metadata as text it is possible to use full-text search engines to index and retrieve semantic information.

Ontology imperfection is addressed by allowing users to combine traditional full-text search results with ontology-based search results. How this combination exactly happens, is not detailed in the literature.

The major weakness of the KIM system is that it does not exploit the ontology in the metadata extraction phase. No semantic relations and no time information are used to increase the precision of the generated metadata. Domain imperfection is not addressed by the system, either.

4.2.2 The system of Vallet et al.

The system proposed by Vallet and his colleagues [VFC05] also provides a complete solution, similarly to KIM, which involves automatic metadata generation and semantic querying.

During annotation, mainly the lexical part of the ontology is used, similarly to the KIM system. In addition, they also exploit a so-called classification taxonomy to help disambiguate ontology instances. The idea is to classify concepts in the ontology into broad categories, such as *Culture*, *Politics*[7] etc. During annotation time, they first classify the document, and based on this classification they choose ontology instances from the candidates, whose category matches the document category. E.g., the word “Irises” is linked to Van Gogh’s painting, if the document is categorized under *Arts*, rather than linking it to the flower.

Documents are represented in the ontology as instances, and semantic annotations are also stored in the ontology. Annotations are connected with documents, and weighted using the standard TF-IDF algorithm.

They require the user to submit an RDQL[8] ontology query, where the query variables can be weighted, showing their importance. This query is first executed against the ontology, and yields a list of ontology instances. Further, all of the documents are retrieved, which refer to those instances. Finally, the results are ranked using the document annotations on the one hand, and the query variable weights on the other hand.

Vallet et al. also address the issue of ontology imperfection by combining the result of the semantic search with the result of a full-text search engine. To create a full-text query, they simply extract all textual information from the original RDQL query. The approach they take is very simple: the rank of a document is simply the weighted sum of the original full-text and the ontology-based search ranks.

Vallet et al.’s work is the only work to the best of my knowledge that evaluates the effect of ontology imperfection on IR results. They executed some queries on a relatively small repository of documents, where some of the queries were about areas that were not covered by the ontology. They found that in those cases the system simply falls back to the level

[7]The taxonomy follows the IPTC Subject Reference Standard, see `http://www.iptc.org/NewsCodes`
[8]`http://www.w3.org/Submission/RDQL/`

of a traditional search engine but does not provide worse results. However, they also found that wrong semantic annotations can seriously spoil the results because the semantic search algorithm trusts those annotation too much.

Discussion

Vallet et al.'s work is very inspiring because they also seriously consider the effect of ontology imperfection on retrieval results. Their work show that one must be very careful during the automatic generation of semantic metadata because wrong semantic metadata has a strong negative impact on retrieval performance.

A major weakness of their system is that they use ontology reasoning for retrieval, which does not scale for big repositories. Moreover, they do not address temporal issues and ignore domain imperfection.

4.2.3 SCORE

SCORE (Semantic Content Organization and Retrieval Engine) is a comprehensive solution [SBA$^+$02] which is reported to be deployed commercially[9]. It provides a high-performance, industry-ready solution. It supports automatic metadata extraction, and also semantic querying.

For metadata extraction SCORE uses a heuristic similar to that of Vallet et al.: it classifies documents into categories, and uses these categories to disambiguate text-to-instance-label mappings. SCORE defines a separate ontology, called the "world model" that defines a hierarchy of document categories together with their attributes that should be automatically filled during semantic metadata generation. I.e., the metadata generation task of SCORE is basically a classical information extraction task where a pre-defined scheme of attributes should be automatically filled. This is a simpler task than generating arbitrary metadata.

SCORE can automatically determine the category of a document using a classifier committee, i.e., by combining results of many classifiers to achieve better results. Knowing the exact category of the document in advance provides good possibilities for disambiguation of entities. E.g., "Golf" can refer to the product of Volkswagen in the category "automobile" and to the sport discipline in the category "sports".

SCORE also uses an ontology that codifies background knowledge about the world, called the "knowledge base". SCORE exploits this ontology by considering semantic relations between instances for disambiguation. I.e., if the document contains many neighbors of one candidate, and none of the other, it chooses the candidate with more neighbors. SCORE can also automatically infer the values of some document attributes using the relations among instances.

[9]However, we could not access the `http://www.voquette.com` URL of the system, which is given in the paper, to verify that fact. It seems that Voquette was acquired by another company, named Fortent (`http://www.fortent.com/`), and SCORE may still provide the basic infrastructure for the various knowledge management products of the company.

Using the generated metadata, various applications can be developed that exploit this metadata, including semantic search, personalization of content etc.

Discussion

SCORE shows a high-performance, pragmatic solution, although the high performance is achieved by storing the whole index in main memory. This may not be feasible for very extensive ontologies and/or document collections. Unfortunately, it is not possible to find out how SCORE works in detail, as in the literature only a high-level description of the commercial system is available.

What definitely can be determined is that the system does not exploit temporal information, and does not address domain and ontology imperfection. An interesting feature of the system is, however, that it exploits semantic relations in the ontology both for disambiguation and for the expansion of the semantic metadata.

4.2.4 OWLIR

The OWLIR (Ontology Web Language and Information Retrieval) system [FMJ$^+$05, SFJ$^+$02] focuses on integrating classical IR technology with semantic search. The system is deployed for event filtering where the goal is to provide university students with relevant event descriptions based on their profile.

The approach of OWLIR is the following. The input of the system are full-text descriptions of events. First, semantic metadata is generated using information extraction techniques. They use the Aerotext™ commercial system, therefore this step is not described in detail. The OWLIR system uses inferencing to enrich semantic metadata which was created by the information extraction system. On the one hand, they exploit the concept hierarchy, e.g., to infer that a basketball match is a sport event. On the other hand, they use the knowledge base to add additional information about specific instances. E.g., they use the Internet Movie Database (IMDB)[10] to add genre information for movies based on their title.

Semantic metadata is stored in DAML+OIL [vPH01][11] but in principle any formalisms based on RDF may be used, including OWL [DS04]. Metadata is embedded into the original document, i.e., a document consists of the original free-text and of the embedded semantic metadata in DAML+OIL format.

The OWLIR system provides a mechanism, termed *swangling*[12], which can convert an arbitrary RDF triple into a string representation. For each RDF triple, seven new RDF triples are generated. These new triples include the original triple and the other triples are formed by replacing one or two positions of the triple with a wildcard entity[13]. E.g., from the triple

`<#movie1, HASGENRE, #romantic>`

[10]`http://imdb.com/`

[11]a predecessor of the OWL W3C standard

[12]Semantic Web mangling

[13]`rdf:Resource`

the following triples are generated:

```
<#movie1, hasGenre, #romantic>
   <*, hasGenre, #romantic>
   <#movie1, *, #romantic>
    <#movie1, hasGenre,*>
      <*, *, #romantic>
       <*, hasGenre,*>
       <#movie1, *, *>
```

Each resulting triple is then hashed and converted to a base-32 number (i.e., to a artificial word) which can be directly indexed by any traditional full-text search engine. To summarize, the swangling process produces seven, full-text-indexer-friendly artificial words for each RDF triple. The swangled terms are then also included into the original document and the result is indexed using a traditional full-text engine.

A user query looks almost identical to a document: it contains free text parts but can also contain DAML+OIL markup. The only difference is that in the DAML+OIL markup of the query, variables may also be used to denote the information needed by the user. After enriching the query using inferencing and applying the described swangling procedure, a simple full-text search provides the results.

The OWLIR system was deployed and evaluated in the domain of university event descriptions, and was shown to provide results that are superior to simple full-text search. The results of the small-scale evaluation showed that semantic search (implemented with swangling) alone is already superior to full-text search, and that the best results are provided when user queries contained both free-text and semantic parts.

Discussion

The OWLIR system demonstrates that ontology-based preprocessing of queries and document metadata can improve the results of IR systems, even if syntactical similarity measures are used during query execution. These results are very motivating for me because they show a possible way to incorporate ontology reasoning to an information system without sacrificing scalability.

Although it is a very flexible solution, one problem with the swangling approach (and with the metadata model itself) is that it is not possible to encode the relative importance of ontology entities (or RDF triples) to the document. By contrast, in the KIM system the identifiers of the recognized ontology entities are inserted each time they appear in the text. I.e., entities that appear more frequently, automatically get a higher score during full-text search[14].

OWLIR does not exploit temporal information, does not address domain imperfection. Moreover, it seems that natural language queries are not supported, that means, in the case of a purely textual query no semantic search happens, only a traditional full-text search is executed. Their automatic metadata generation solution is not used to automatically generate a

[14]Using the usual TF-IDF score.

semantic part for pure full-text queries. Finally, it is somewhat redundant to perform ontology-based inferencing both on the metadata and on the query — a fact that the authors of OWLIR also admit.

4.3 Summary

In the previous sections I analyzed the state of the art based on the requirements that were discussed in Chapter 3. A summary of this analysis is shown in Table 4.1.

Table 4.1: Systems and requirements

	NL vagueness	Semantic relations	Time dimension	Ontology imperfection	Domain imperfection	Scalability	Metadata generation	Evaluation	Natural language query
QuizRDF	X[1]	X[2]	-	N	-	X	N	-	X
SEAL Portals	X	X	-	N	-	-	N	X	-
Zhang et al.	X[3]	X	-	N	X[4]	-	N	X	X[5]
Rocha et al.	X	X	-	N	-	-	N	X	X
KIM	X	X[6]	-	X[7]	-	X[8]	X	-	X[9]
Vallet et al.	X	X	-	X	-	-	X	X	-
SCORE	X	X	-	-	-	X	X	-	?
OWLIR	X	X	-	X	-	X	X	X	-

Legend: **X** – fulfills the requirement, **-** – does not fulfill the requirement, **N** – not applicable, **?** – not known, **X[1]** – only in the case of classes, **X[2]** – only browsing is supported, **X[3]** – not solved in the full-text part, **X[4]** – there is potential in the formalism, **X[5]** – in the full-text part of the query, **X[6]** – only concept taxonomy, **X[7]** – no details are known how full-text search integrated, **X[8]** – starts with ontology search that may not scale, **X[9]** – only in the ontology search part

Based on this analysis the following general statements can be made. All of the systems address the issue of the vagueness of natural language, which is not surprising, as it is one of the main motivations to use an ontology in the first place to avoid the problems that this vagueness causes. Most of the systems combine various techniques such as natural language processing, full-text search and ontology reasoning. It is also apparent that systems that scale well, achieve this scalability by using traditional full-text indexing and querying. They encode semantic information in a way that it can be indexed by traditional search engines. In other words, they also accept the assumption that it is possible to create such document and query representations that the classical, syntax-based ranking algorithms yield semantically meaningful results.

It is interesting to see that although some of the systems are evaluated casually, almost no evaluation was conducted in a real-world situation, where the ontology was incomplete. Vallet et al. were the first to admit that ontology imperfection can have negative effects on the performance of an ontology-based information system, and they were the first to evaluate some of the consequences of imperfect ontologies.

In general, it can be stated that systems focusing on document retrieval address the issue of ontology imperfection by combining full-text search with semantic search. They either do it implicitly by encoding the textual representation of the semantic metadata into the document text (KIM, OWLIR) or they explicitly combine the results of pure full-text and semantic searches (Vallet et al.).

The issue of domain imperfection is not considered by any of the systems. Temporal information is not exploited by any of the analyzed approaches, either. Finally, during the mapping between document texts and ontology entities, mostly only syntactic heuristics are used, or at most the concept taxonomy is exploited. OWLIR and SCORE advance this state of the art by expanding semantic metadata using semantic relations in the ontology.

Chapter 5

Overview of my approach

As was shown in the previous chapter, currently there is no approach mentioned in the literature that fulfills all of the identified requirements in Chapter 3. Therefore, a new approach is needed that fulfills these requirements. This chapter gives a high-level overview of such an approach. My aim is to give a basic, intuitive understanding of the whole solution. In the subsequent chapters, I will introduce specific parts of the solution in more detail.

Based on the analysis in Chapter 3, the problem statement of this work is intuitively the following: The goal is to exploit background knowledge stored in ontologies to increase IR effectiveness so that

- the performance (speed) of the resulting system remains adequate even for big document collections, and
- the possible negative effects of ontology and domain imperfection are tolerated.

5.1 Lessons learned from related work

Based on the analysis of related work in the previous chapter, the following lessons have been learned.

Scalability through full-text search: All of the systems that achieved good performance on big repositories used full-text search engines for document indexing and querying. It seems that today there is no real alternative to full-text search for efficient information retrieval.

Semantic metadata in full-text indexes: In spite of the fact that full-text search engines are used, it is still possible to annotate documents with semantic metadata and harvest the advantages of semantic technology. The solution is to encode semantic annotations so that the result is compatible with the information model used by the chosen full-text search engine[1]. The swangling approach introduced by the OWLIR system [FMJ$^+$05, SFJ$^+$02] shows that in principle, it is possible to encode arbitrary RDF statements in full-text indexes. However, syntactic similarity measures applied by the full-text search engine still must make sense on the encoded version of the semantic metadata to provide meaningful results. It seems questionable

[1] usually the “bag of words” model

whether this assumption holds for such complicated encodings as the introduced swangling technique.

Natural language processing for text-to-ontology mapping: Many systems apply shallow natural language processing (NLP) techniques to improve the results of automatic metadata generation. A major part of metadata generation is the mapping of text snippets to ontology instances. In this task, it is clearly advantageous to use the various syntactic-linguistic heuristics of NLP to find meaningful text snippets that are matched with the lexical layer of the ontology. Moreover, the additional type information that is delivered by some NLP systems[2] is also useful to increase the quality of text-to-ontology mapping.

Combining full-text and semantic search: Some systems combine the results of semantic search[3] with the results of traditional full-text search to deal with ontology imperfection. This is a very simple, natural way to tackle the problem, and has the advantage that the ontology-based system operates exactly as a full-text search system if there is absolutely no ontology available. In other words, the system degrades nicely as the imperfection of the ontology increases.

5.2 Missing features in state-of-the-art solutions

Although state-of-the-art systems solve lots of problems analyzed in Chapter 3, there are still some issues which remain unsolved.

First, **domain imperfection is ignored** by all of the systems. Although in the fuzzy description logic-based information system of Zhang and her colleagues [ZYZ^{+}05] there is a potential in the ontology formalism to represent some aspects of domain imperfection, they do not exploit this opportunity.

Second, the **time dimension is not handled** by any of the systems, i.e., it is not possible to search for temporal information.

Third, **ontology information is not fully exploited** by any of the systems during automatic semantic metadata generation. Some of the reviewed systems use only the lexical layer of the ontology, and the concept hierarchy. In these systems, semantic metadata generation basically means matching ontology entity labels to text snippets in the document content. Of course, this task is already challenging because the mapping between the textual labels and abstract ontological entities is not always trivial and there is often a need for disambiguation.

Some other systems also exploit semantic relations in the ontology to expand the generated metadata and thus to find indirectly relevant instances. The evaluation results of OWLIR show the great potential in doing such expansion. However, there is a chance in the case of an extensive ontology (or in the case of an ontology with errors) that such an expansion adds wrong information to the semantic metadata, especially when applied to the uncertain results

[2]such as the information whether a text snippet denotes a person, or a location

[3]I consider also full-text search operating on encoded semantic metadata as semantic search.

of the disambiguation step. Therefore, all information should be considered to limit the chance of wrong inferences. My hypothesis is that considering the temporal context of the actual document together with the time information stored in the ontology can effectively scope the set of possibly related instances and therefore reduce the chance of wrong inferences.

Finally, the reviewed systems focusing on document retrieval support only traditional ontology-based queries for semantic search, and **do not provide the possibility** for the users **to start a semantic search by specifying a full-text query**.

My solution addresses these weaknesses, and this is the main contribution of this thesis.

5.3 Fundamental modeling decisions

To come up with a solution that fulfills the requirements that were identified in Chapter 3 and advances the state-of-the-art in the areas that were discussed above, some fundamental modeling decisions have to be made. These decisions are discussed in this section.

Same semantic information model for query and documents: I use the same information model to represent semantic metadata of documents and semantic user queries because it has some advantages (see also Section 2.1). First, using the same model, information retrieval can be intuitively understood as finding the most similar document to the query. This makes ranking of results natural. Second, because there is no difference among documents and queries from a conceptual point of view, this approach can be also used for information filtering, when new documents should be matched with user profiles, defined as static queries.

Syntactic similarity measure: The potential of ontology-based similarity measures were identified in Section 3.4.1. However, based on the analysis of the related work, and on my own practical experiences, I believe that such similarity measures simply do not scale well enough to be used in information systems with big document collections. Therefore, I also follow the approach that was already successfully applied by some other systems and use traditional full-text search for querying. In other words, I use the syntactic similarity measure provided by full-text search engines.

Time dimension: To provide support for the time dimension, temporal information must be included into the information model used by my system.

NLP for metadata generation: I follow the trend in state-of-the-art systems and apply shallow NLP on document texts to increase the quality of text-to-ontology-entities mapping. Shallow NLP means that a full (and costly) linguistic analysis of the document text is not conducted, only some limited text analysis techniques are applied, such as tokenization, sentence splitting or named entity recognition.

Fully exploiting ontology: It is extremely expensive to develop ontologies, and there is a lot of useful information stored in them in the optimal case. It is a waste of resources to ignore this information during the metadata generation phase. Using purely syntactic techniques, it is impossible to identify relevant instances that are only implicitly mentioned in the document text. In other words, finding relevant documents that do not explicitly mention some search terms is possible only with semantic search. It is one of the features where semantic search is clearly superior to traditional full-text search. Therefore, this opportunity should be exploited. In contrast to many state-of-the-art systems, I will use semantic relations between instances to find indirectly relevant instances. Moreover, during this process, I will consider the temporal context of the document and the temporal information stored in the ontology to make well-founded decisions about which relations should be really exploited[4].

Ontology-based heuristic rules: Although the ontology itself contains the domain knowledge, it does not specify how to exploit this knowledge. I specify the operational knowledge that describes *how* the ontology should be used during the IR process via ontology-based heuristic rules. The term "heuristic" only emphasizes the fact that it is not fully possible to model by simple rules the complex cognitive process of humans when they decide about the relevance of abstract concepts. Therefore, the rules will just approximate this cognitive process, i.e., they can be viewed as heuristics.

Avoiding redundant use of ontology reasoning: In Section 3.4.1 I discussed the potential advantages of using ontology reasoning during the indexing and querying steps. It is important to see, however, that these two steps should not be analyzed in isolation because they are complementary. Clearly, if resources are perfectly annotated with semantic metadata, i.e., the metadata contains all directly or indirectly relevant ontology entities, there is no need to do any query extension because even a simple syntactic matching of metadata element URIs will find all relevant documents. But even if we do not manage to generate the perfect metadata, it makes no sense to exploit the same ontology during query extension that was used during metadata generation to extend semantic annotations because the query extraction process would suffer from the same problems as the metadata generation process.

Thus the ontology should be used either only during the metadata generation or only during the query expansion step. I choose the metadata generation step because this choice seems to have some advantages:

- The temporal and conceptual context[5] of the document can be estimated better because the estimation is based on a much longer text than at query time. A better estimation of the temporal and conceptual context means better decisions later, when indirectly relevant ontology entities are identified.
- The efficiency of ontology reasoning is less critical because metadata generation is performed offline. This also means that more complicated reasoning axioms can be used to improve semantic metadata than would be possible during query time.

[4]Consider e.g., the example about Napoleon and Russia in Section 3.4.
[5]The conceptual context here denotes the already identified ontology entities in the document text.

So, I use ontology reasoning only during metadata generation, and use fast, syntactic heuristics during query expansion.

Combination of semantic and full-text query results: As was discussed, all state-of-the-art systems that somehow consider ontology imperfection, solve the problem by combining results of semantic search with results of a full-text search engine. This has the advantage that the new system is "backwards compatible" with the classical, full-text search based information systems. I.e., even if there is no ontology at all, the system is still operational. I also use this approach to deal with ontology imperfection.

5.4 The ontology-supported IR process

The decisions described above resulted in the following ontology-supported IR process, shown in Figure 5.1.

The ontology-supported IR process has the same major parts than the classical IR process: indexing and querying.

During the *indexing step* the goal is to create semantic metadata describing the meaning of a document as well as possible. The metadata generation process is split into three parts. First, *shallow natural language processing (NLP)* is used to generate all kinds of useful information that can improve the quality of the text-to-ontology mapping later. This information includes the specification of token and sentence boundaries, and the identification of various types of text snippets: phrases denoting persons, organizations, time specification etc. The information created by NLP is stored in the form of *NLP annotations*. These annotations are used next to create an *initial metadata representation*. During this initial step, text snippets are matched with ontology instances. *Disambiguation* of the ontology instances is also performed in this step. Finally, the *initial set of metadata elements is expanded* using ontology-based heuristic rules. The goal of this expansion step is to find ontology entities that are relevant to the document but which are not explicitly mentioned in the document text.

After the semantic metadata is generated, it should be *indexed by a full-text search engine*. During this step, the metadata is transformed into a representation that is compatible with some chosen full-text engine. After the transformation, a full-text index of the semantic metadata can be created (metadata index in Figure 5.1).

In addition to the semantic metadata creation and indexing, the indexing part of the ontology-supported IR process also includes the *classical full-text indexing of document contents*.

The *querying part* of the process starts with the textual user query[6]. As the first step, this *query has to be parsed*, and a semantic representation of the query has to be provided. As was discussed, the semantic query uses the same information model as the document metadata. Therefore, the same NLP and metadata generation components can be used for this task as for the metadata generation of documents. The only difference is that metadata expansion is not performed during query time, as was discussed in the previous section.

[6]Expert users also have the option to directly specify the semantic query.

As the next step, *two full-text queries are generated* from the semantic query. The first query searches the full-text index representing the semantic metadata of the documents. This query is termed the *metadata query*. The second query searches the full-text index containing the indexed document contents. This query is termed the *content query*. After executing these two queries, the *results are combined*, and the combined result constitutes the final query results that are returned back to the user. I term the whole query process combining the results of the metadata and content queries *semantic querying*.

Various aspects of the ontology-supported IR process are described in Chapters 8 and 9 in more detail.

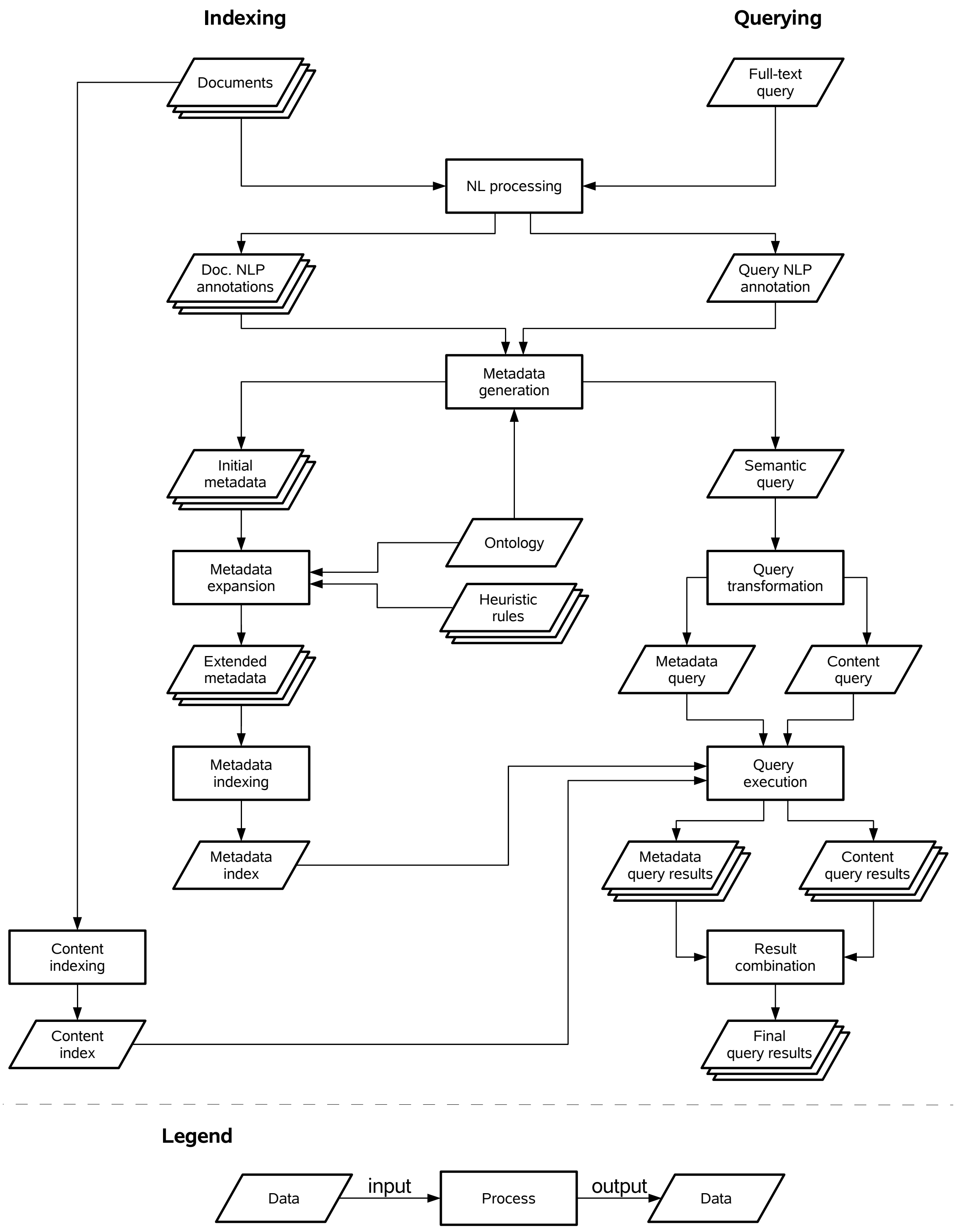


Figure 5.1: Overview of the ontology-supported information retrieval process

5.5 High-level system architecture

To evaluate the ideas presented above, a research prototype named IRCON (**I**nformation **R**etrieval in **C**ontext, using **ON**tologies) was developed, which implements the outlined ontology-supported IR process. The IRCON prototype is a classical three-tier web application, implemented in Java. The high-level architecture of IRCON is shown in Figure 5.2.

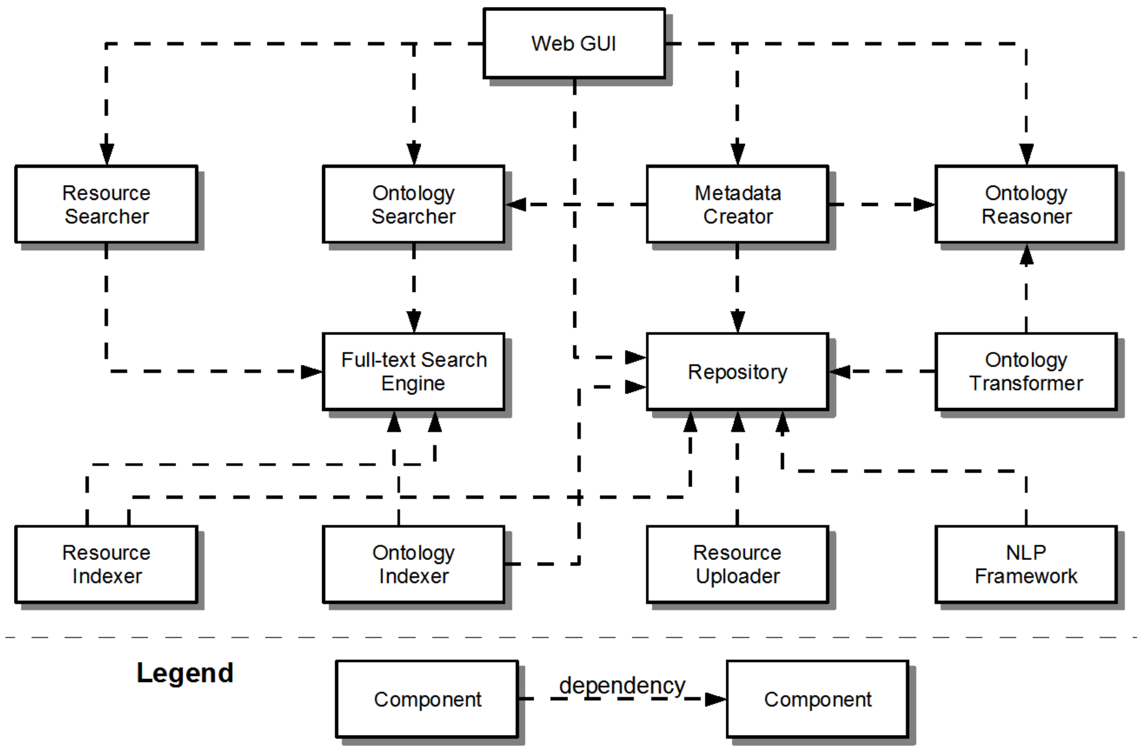


Figure 5.2: The high-level architecture of IRCON

In the architecture, I combine various technologies to achieve scalability and good quality of results at the same time. In detail, the following technological decisions were made:

- For storing *structured information* that does not require ontology reasoning, the mature relational database technology is used. Structured information in this application includes semantic annotations, and also the lexical layer of the ontology. I store the ontology lexical layer in the database because it is just simple syntactical information, and I do not want to reason on it, only to retrieve it for indexing purposes. Further, also the document texts themselves are stored in the relational database, which provides access control on the documents, and allows to keep the repository consistent. In the IRCON prototype, I used the freely available PostgreSQL database management system[7].

- For *full-text search* over ontology labels and in documents, a classical full-text search engine is used. Because the prototype was implemented in Java, I used the Lucene

[7]http://www.postgresql.org/

Java framework[8] to implement the search engine. Lucene is the most popular open-source framework in the Java world for search engines, which scales up to millions of documents. Therefore, it was suitable for my purposes to demonstrate the viability of the solution on real-world size document collections.

- *Ontology navigation and reasoning* is used during ontology-supported indexing of documents, as was discussed above. Ontology management is implemented using the KAON[9] and KAON2[10] frameworks. While KAON provides a comprehensive ontology editing framework [MMV02], KAON2 provides an efficient ontology reasoner implementation [HMS04].
- During metadata generation the GATE[11] framework is used. GATE is an open source, Java-based *NLP framework*. This framework is very popular in the NLP research because it allows an easy combination of various NLP tools, such as part-of-speech tagger, named entity recognizer etc. It forms the basis of some more advanced systems requiring NLP, such as the Text2Onto framework for ontology learning [CV05].

To summarize, I **combine four technological areas** in my architecture to achieve the goals that were identified during the problem analysis: relational databases, full-text search, ontology reasoning, and natural language processing.

The components of the architecture shown in Figure 5.2 are described in the following:

Full-text Search Engine: Maintains its own optimized, file-based full-text indexes (both for metadata and content searches), and executes full-text searches against it. Theoretically, any full-text search engine could be used here, including major web search engines, such as Google. As was discussed, in my prototype I used my own implementation, a search engine based on the Lucene framework.

Repository: Based on a PostgreSQL relational database, the repository stores all kinds of structured information that does not require ontology reasoning, as was discussed above.

NLP Framework: As was discussed, various components of the GATE NLP framework are used. This component takes the text of a document and produces NLP annotations describing various linguistic aspects of the text.

Ontology Reasoner: An important component of the application, it provides access to the ontology. It allows other components to navigate in the ontology structure, and it also allows them to execute ontology-based queries. As was already discussed, the ontology reasoner builds on the KAON and KAON2 frameworks.

Metadata Creator: Creates the semantic annotation of a document. First, it processes the results of the NLP subsystem, which are stored in the repository. Later, it refines the results provided by NLP using ontology-based heuristics. This component is also able to parse a textual query into its semantic representation.

[8] `http://lucene.apache.org`
[9] `http://kaon.semanticweb.org`
[10] `http://kaon2.semanticweb.org`
[11] `http://gate.ac.uk/`

Resource Indexer: Indexes the resources (documents) in the repository. This includes a traditional full-text indexing, and also the indexing of the semantic metadata that was created by the Metadata Creator component, and which is stored in the repository. This component is also responsible for transforming the semantic metadata during indexing to a form that is processable by the full-text search engine.

Resource Searcher: Executes the semantic query, combining the results of the metadata and content queries. It takes a semantic query as an input, and it is responsible for generating the FTS query representations of the semantic query.

Ontology Indexer: Indexes the lexical layer of the ontology. Because the lexical layer contains only textual information, it can be readily indexed by the full-text search engine, no transformation is needed. In addition to the lexical layer information, also information about the parent concepts of an ontology instance is encoded into the index to speed up ontology navigation.

Ontology Searcher: Provides full-text search on the ontology instance labels. Also the required ontology concepts can be specified during the search. E.g., it is possible to search for instances of the PERSON concept that have the label "Bush".

Web GUI: The graphical user interface of the prototype is a usual web interface, implemented using the open-source Tapestry framework[12]. Apart from allowing the user to execute searches in the usual way, the GUI also displays the semantic annotation of a document, and it also allows the user to explore the ontology itself. During the semantic search, the web GUI is responsible for parsing the textual user query into a semantic query representation using the Metadata Creator component. The semantic query is later processed by the Resource Searcher component.

Resouce Uploader: Uploads resources[13] into the resource repository.

Ontology Transformer: The ontology is not necessarily specified by domain experts in a format that can be readily used by the ontology reasoner. This component transforms the ontology from its conceptual format to the format that is supported by the ontology reasoning frameworks used in IRCON.

5.6 Feasibility of the approach

On the one hand, the introduced ontology-based information system has the potential to significantly increase the quality of IR results. On the other hand, it was already discussed that developing ontologies is very expensive, and takes a long time. It is easy to see that a document collection in a company Intranet, or on the World Wide Web, potentially contains documents about arbitrary topics. This means, to fully exploit the power of ontologies an ontology would be needed that models the whole world. Experience shows that building such an ontology is practically impossible. Although there were some attempts to build such mega-ontologies[14],

[12]`http://jakarta.apache.org/tapestry/`

[13]in the current implementation textual documents

[14]E.g., the CYC project (`http://www.opencyc.org/`), whose ontology currently contains 47000 concepts, and 306000 facts about these concepts.

none of these ontologies really managed to model the whole world. Moreover, no really successful application of such huge ontologies is known, therefore the high costs of their development are not justified. In general, it can be observed that the bigger an ontology is, the more difficult to comprehend, and therefore the more difficult and expensive to extend it. This suggests a natural upper-limit on the size of ontologies that can be developed with justifiable investments. This size is definitely much smaller than the size of the "world ontology" would be.

This means that it is practically impossible to cover all documents in the information system by an ontology, and therefore it seems questionable whether it is feasible to develop an ontology-based information system. My proposed system, however, does not require full ontology coverage. Ontology imperfection was accounted for by using NLP and full-text search. This means, the system can operate even with a completely empty ontology. Even in this case, slightly better results are expected than using traditional word-based full-text search engines because of the application of NLP techniques. However, when an ontology of adequate quality is available for some specific domain, the system will hopefully outperform traditional full-text search. In other words, the system does not force its users to fully transform their existing full-text search systems into an ontology-based one but allows a smooth transition between the two worlds. Actually, I believe that the state when every topic in the repository is covered by an ontology will be never reached in practice. For many kinds of searches, the use of ontology-based techniques is simply unnecessary, as was discussed Section 3.2. For such topics, users will probably never have the motivation to invest money and resources in ontology building.

Therefore, I foresee the following usage of the proposed system in a practical situation. First, the system is operated without any ontology. Later, specific domains can be identified, where the effectiveness of the system is inadequate[15]. In these specific areas ontologies are developed, together with heuristics that describe how to exploit the ontological information to improve metadata generation, and consequently to improve IR results. This usage pattern guarantees that

- resources are not wasted on building ontology and developing heuristics on domains where full-text search already provides adequate results,
- the costs of ontology development are directly motivated by user need, and increased user satisfaction caused by better results instantly justifies the costs,
- ontology development will be feasible because small, well-scoped ontologies are built, and no attempt is made to build some huge "world ontology".

5.7 Summary

In this chapter, I gave a general overview of my approach. First, I reviewed ideas and lessons learned from state-of-the-art systems, which are useful in my research context. Later, I analyzed missing features in state-of-the-art systems, and described an ontology-based information retrieval process, which addresses those deficiencies. Next, I introduced the high-level

[15]E.g., by analyzing user complaints, or performing evaluation.

system architecture, which technically realizes that IR process. At the end of the chapter, I discussed the feasibility of my proposed ontology-based IR system in a real-world situation.

Chapter 6

Representing temporal imperfection

As was discussed in Section 3.5.2, domain imperfection should explicitly be represented in the ontology to lower ontology imperfection and consequently to create the potential to increase the quality of search results (DOMAIN IMPERFECTION requirement). This chapter describes a solution for representing domain imperfection for temporal specifications in ontologies. A new, fuzzy temporal model is proposed after analyzing the requirements for such a temporal model. Later, it is shown how to acquire such fuzzy temporal specifications from domain experts. Finally, the generality of the approach is demonstrated by applying it on imperfect spatial specifications.

6.1 Requirements

As was shown in Section 3.6, the three aspects of temporal imperfection, namely uncertainty, vagueness and subjectivity, should be represented in a common modeling framework (UNIFIED REPRESENTATION OF TEMPORAL IMPERFECTION requirement). This is a minimal requirement for a suitable time model.

6.1.1 Temporal Relations

Generally speaking, a temporal model should provide support for the usual temporal relations. A model that does not provide for temporal relations, cannot be considered a proper temporal model. The usual requirement in the field of temporal models is that a model should at least be able to represent the thirteen temporal interval relations defined by Allen in [All83]. Apart from these relations, we have found during the VICODI project that the relation `intersects`, checking whether two time intervals have a common point, is also very useful in the historical context [NM03]. Hence, Table 6.1 summarizes all of the operations required[1]. In this table, for an interval i, i^- denotes the starting point of the interval and i^+ denotes the ending point of the interval.

[1]i.e., the thirteen relations of Allen and the `intersects` relation

Table 6.1: Required temporal relations

Interval Relation	*Definition*
i `before` j	$i^+ < j^-$
i `after` j	j `before` i
i `overlaps` j	$(i^- < j^-) \wedge (i^+ > j^-) \wedge (i^+ < j^+)$
i `overlapped-by` j	j `overlaps` i
i `during` j	$(i^- > j^-) \wedge (i^+ < j^+)$
i `contains` j	j `during` i
i `meets` j	$i^+ = j^-$
i `met-by` j	j `meets` i
i `starts` j	$(i^- = j^-) \wedge (i^+ < j^+)$
i `started-by` j	j `starts` i
i `finishes` j	$(j^- < i^-) \wedge (i^+ = j^+)$
i `finished-by` j	j `finishes` i
i `equals` j	$(i^- = j^-) \wedge (i^+ = j^+)$
i `intersects` j	$(i^+ > j^-) \wedge (i^- < j^+)$

6.1.2 Compatibility with classical time specifications

It is a natural requirement that the temporal model should, if applied to traditional, crisp and definite temporal specifications, yield the same results as in classical temporal models. After all, there are some temporal specifications, which are precise, and we would like to use those together with the imperfect specifications. In other words, the approach should naturally subsume the classical case.

6.1.3 Temporal Specifications vs. General Theory of Time

My requirements differ from those usually found in temporal reasoning literature (e.g. [DP89, Dut88, GV95, DAR02]). My goal is not to develop a general axiomatization of time, which can be used to reason about relatively known temporal events. For example, in temporal logic one can axiomatize that event A occurred before event B and event B occurred before event C. Then one can derive that A occurred before C, even without knowing the exact time when either of the events occurred.

In my application field, I deal mainly with concrete temporal specifications which may be imperfect but still use absolute dates. I.e., instead of knowing the event A occurred before event B, the absolute dates of the A and B events are known (such as 750 B.C or 2006-01-01). In such a setting, axiomatizing the total order of the time dimension is not necessary since it follows naturally from the total order of dates. E.g., it is quite easy to determine using basic arithmetics that the year 2000 is later than the year 1000, there is no need for complicated logical axioms.

I have found that many application domains share this fundamental feature: rather than requiring a general theory of time allowing arbitrary inferences, they deal with numerous concrete temporal facts associated with domain entities[2]. In such a setting, a general theory of time is overkill because it unnecessarily requires inferencing capabilities of significant computational complexity. This logical theory should be replaced with more efficient mechanisms, implemented outside the logical framework.

6.2 Models of imperfection

As we have seen, *three aspects of imperfection* have to be modeled: *uncertainty*, *vagueness* and *subjectivity*. In this section, I review the most popular theories to represent these aspects of imperfection.

6.2.1 Modeling uncertainty

For modeling uncertainty, probability theory is the most popular and the most suitable theory. I assume that the basics of probability theory are well-known to the reader. If not, the basics can be found in many textbooks, e.g., in [Fel70].

6.2.2 Modeling vagueness

Although probability theory could also be used to model vagueness, it is generally debated whether probability distributions are appropriate for representing cases when objective statistics that probability distributions are based on are missing [DP86]. Because of that, most approaches in temporal reasoning for modeling vague temporal knowledge use the *possibility theory* that was proposed by Zadeh [Zad78].

Because possibility theory may not be generally known, I review the basics here. More details are available in [DP86].

Let $\Pi : 2^{\Omega} \rightarrow [0, 1]$ be a possibility measure defined on an event space Ω. Let $\Pi(A)$ denote the degree of possibility that event $A \subseteq \Omega$ occurs (or that proposition A is true). The fundamental axiom of possibility theory is that the possibility of the disjunction of two propositions A and B is the maximum of the possibility of the individual propositions [DP86]:

$$\Pi(A \vee B) = max(\Pi(A), \Pi(B)) \tag{6.1}$$

Related to this possibility measure we can also define a possibility distribution $\pi : \Omega \rightarrow [0, 1]$ as

$$\pi(x) = \Pi(\{x\}) \quad \text{for all } x \in \Omega \tag{6.2}$$

Unlike a probability distribution, a possibility distribution does not necessarily sum up to one.

[2]E.g., the life times of historical persons.

Using (6.1) it is easy to see that

$$\Pi(A) = max_{x \in A} \pi(x) \quad \text{for all } A \in \Omega \tag{6.3}$$

A classical example to show the difference between a possibility measure vs. a probability measure follows [Zad78, Sme96]. Consider the number of eggs that Hans is going to eat tomorrow morning. Let $\pi(u)$ the degree of ease with which Hans can eat u eggs. Let $p(u)$ be the probability that Hans will eat u eggs tomorrow. A possible distribution of $p(u)$ and $\pi(u)$ is shown on Table 6.2.

u	1	2	3	4	5	6	7	8
$\pi(u)$	1.0	1.0	1.0	1.0	0.8	0.6	0.4	0.2
$p(u)$	0.1	0.8	0.1	0.0	0.0	0.0	0.0	0.0

Table 6.2: The possibility and probability distributions for the egg example

We can see that while the possibility that Hans can eat four eggs for breakfast is one, the probability that he may do is zero. While the possibility distribution describes Hans' capability to eat eggs, the probability distribution describes his habit about eating eggs. Although Hans could perhaps eat even 8 eggs if he starts with a completely empty stomach and he is forced to do it (possibility), it is absolutely unlikely that he would eat 8 eggs next day based on the statistics about the eggs he has eaten in the past (probability). I.e., a high degree of possibility does not imply a high degree of probability, nor does a low degree of probability imply a low degree of possibility. However, if an event is impossible, it is also improbable.

6.2.3 Modeling subjectivity

For representing the third type of imperfection, subjectivity, the major theory is the theory of fuzzy sets, also proposed by Zadeh [Zad65].

In this subsection, I briefly recapitulate the fundamental notions about the fuzzy sets and fuzzy logic. Further details can be found in any textbook of fuzzy sets or fuzzy logic (e.g. [DP00, NW97, MMSW93]).

Fuzzy sets generalize the notion of classical, crisp sets[3]. A crisp subset A of the set $\mathcal{U}$ (the universe of discourse) can be specified using the *characteristic function* $A : \mathcal{U} \to \{0, 1\}$. $A(x) = 1$ if $x \in A$ and $A(x) = 0$ if $x \notin A$. Similarly, a *fuzzy subset* $\mathbf{A}$ of $\mathcal{U}$ can be characterized with a *membership function* $\mathbf{A} : \mathcal{U} \to [0, 1]$. For each $x \in \mathcal{U}$ $\mathbf{A}(x)$ represents the membership grade of x in $\mathbf{A}$. Hence, x can be a member of a $\mathbf{A}$ only partially. I call fuzzy subsets (of $\mathcal{U}$) simply *fuzzy sets* from now on and assume that the universe of discourse is understood from the context.

Similarly to the crisp case, the logical connectives *and* ($\wedge$), *or* ($\vee$) and *negation* ($\neg$) may be identified in the fuzzy case with the set operations *fuzzy intersection* ($\cap$), *fuzzy union* ($\cup$) and

[3]The word "crisp" is used as a counterpart of "fuzzy".

fuzzy set complement ($\mathbf{A}^C$). The usual definition of the fuzzy set operations (which we will also use in this thesis) are the following:

$$(\mathbf{A} \cap \mathbf{B})(x) = \min(\mathbf{A}(x), \mathbf{B}(x)) \tag{6.4}$$

$$(\mathbf{A} \cup \mathbf{B})(x) = \max(\mathbf{A}(x), \mathbf{B}(x)) \tag{6.5}$$

$$\mathbf{A}^C(x) = 1 - \mathbf{A}(x) \tag{6.6}$$

The *core* of $\mathbf{A}$ is the crisp set $C_{\mathbf{A}} = \{x \in \mathcal{U} : \mathbf{A}(x) = 1\}$, i.e., the set of elements which completely belong to $\mathbf{A}$. The *support* of $\mathbf{A}$ is the crisp set $S_{\mathbf{A}} = \{x \in \mathcal{U} : \mathbf{A}(x) > 0\}$, i.e., the set of elements which somewhat belong to $\mathbf{A}$.

A fuzzy set $\mathbf{A}$ is called *convex* if the following holds:

$$\forall x \forall x_1 \forall x_2 \;\; x \in [x_1, x_2] \Rightarrow \mathbf{A}(x) \geq \min(\mathbf{A}(x_1), \mathbf{A}(x_2)) \tag{6.7}$$

The *height* of a fuzzy set is the maximum membership grade of any element from the universe in the fuzzy set. I.e.,

$$height(\mathbf{A}) = \sup_{x \in \mathcal{U}} \mathbf{A}(x) \tag{6.8}$$

A fuzzy set is called *normalized* if its height is 1.

6.2.4 Comparison of the theories

If we consider the three introduced theories (probability theory, possibility theory, and fuzzy sets), it is easy to see that from a mathematical point of view all three theories assign a value from the $[0, 1]$ interval to each member of the universe of discourse. The only difference from this mathematical point of view is the set of axioms that must hold on this assignment. The axioms of the probability theory are the most restrictive, whereas in the case of the fuzzy set theory and the possibility theory, there are no restrictions on the assignment. Thus, from a mathematical point of view, all three theories define fuzzy sets. Therefore, it is useful to discuss the semantic difference between the theories from an epistemical point of view. The difference between the possibility and the probability theory was already discussed in Section 6.2.2, therefore I will concentrate here on the difference between the fuzzy set theory and the other two theories.

Fuzzy set vs. possibility distribution

First, let us compare the fuzzy set theory with the possibility theory. Both a fuzzy set and a possibility distribution define an ill-defined set by providing membership grades or possibility values for the members of a universe. Our intuition tells us that it does not really matter whether an ill-defined set is given as a fuzzy set or a possibility distribution, it is the same. Zadeh formalized this intuition in his *possibilistic principle* as follows. Let $\mathbf{A}(x)$ denote the membership of $x \in U$ in $\mathbf{A}$ fuzzy set, describing an ill-defined set. Let $\pi_A(x)$ denote the possibility distribution of the same ill-defined set. In this case the following equality holds:

$$\pi_A(x) = \mathbf{A}(x) \quad \text{for all } x \in \mathcal{U} \tag{6.9}$$

where $\mathcal{U}$ denotes the universe. In other words, a fuzzy set can be also interpreted as a possibility distribution and a possibility distribution can also be interpreted as a fuzzy set.

The only difference between a mathematical fuzzy set interpreted as a fuzzy set, or a possibility distribution, is the usual application scenario where they are used. In the case of a fuzzy set, one normally knows an objectively measurable feature of an individual and the question is how much this individual belongs to the ill-defined set. In the case of the possibility theory, we know for sure that an individual belongs to the ill-defined set and we are interested in the possibility that this individual has some specific measurable feature. E.g., let us consider the fuzzy set of "Tall men", shown in Figure 6.1.

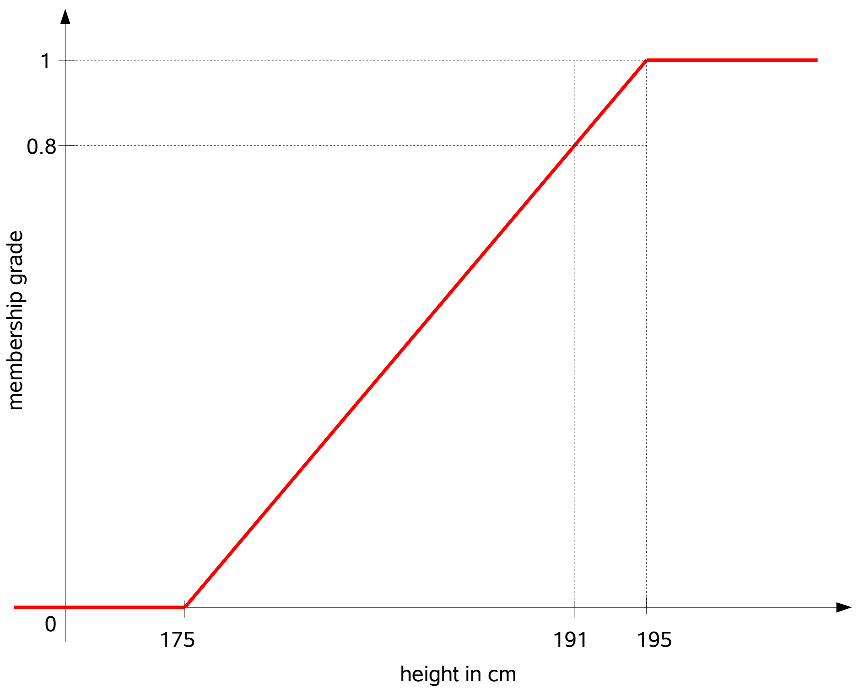


Figure 6.1: Fuzzy set of tall men

If we interpret this definition as a fuzzy set we can ask how much Hans, who is 191 cm tall, belongs to the set of "Tall men". The answer is 0.8, i.e., Hans is pretty tall. If we interpret this mathematical construct as a possibility distribution, we can ask, how big is the possibility that Hans is 191 cm tall, if we know that Hans is tall. The result is again 0.8, this time denoting the information that it is quite possible that Hans' height is exactly 191 cm if we know that he is tall.

Fuzzy set membership value vs. probability

To highlight the difference between the degree of membership, represented by fuzzy sets, and uncertainty, represented by probability theory, let us consider the following example. If we have a bottle of unknown liquid where it is written: "Drinkable with degree 0.7" (degree of membership), it is 100% sure that the liquid will not kill us but it is also 100% sure that we will have some health problems after drinking the liquid. An example for such a liquid could be swamp water. On the other hand, if it is written on the bottle: "Drinkable with probability 0.7" (uncertainty), we have either a totally drinkable liquid (e.g., water) or a totally undrinkable liquid (e.g., poison) in the bottle.

It is also clear that the two dimensions can be mixed, i.e., it is easy to imagine a bottle which is "drinkable with degree 0.7 with a probability of 0.8". In this case, it is very likely, that the liquid is almost drinkable (e.g., swamp water).

6.3 Hierarchy of imperfection aspects in time specifications

Although it was already briefly explained in Section 3.6 why the aspects of uncertainty, vagueness and subjectivity have to be combined in historical time specifications, I would like to further motivate the need for this combination on a detailed example.

Based on my analysis of historical specifications, I noticed that there is a natural hierarchy among the aspects of imperfection in these specifications. To explain these aspects, I take the example of specifying the time interval of the "Dark Ages". The Dark Ages normally denotes the period of the Early Middle Ages and it starts with the fall of the ancient Western Roman Empire, and ends with the coronation of Charlemagne as the Emperor of the Roman Empire. The beginning of the end of the Dark Ages starts with the birth of Charlemagne[4].

On the lowest level there is the **vagueness** of temporal specifications which is inherent in natural language. "Early morning", "in spring", "at the beginning of the eighties" are just some simple examples of typical vague phrases, which usually appear in natural language texts. In our example there are many examples of vagueness. The fall of the Roman Empire happens gradually during the period from 337 to 476. As another example the birth date of Charlemagne can be considered. One possible period for the birth date starts on 747-04-15 and ends on 748-04-01, and no further information is known to make this period more precise. It is interesting to note that even natural language specifications, which seem to be precise, such as "in 476", are vague if we use a higher granularity of time in our model, such as days.

Above vagueness comes **uncertainty**. This aspect of imperfection is probably not very common in most application domains but it is important in history. In history, in some cases there are contradictory accounts about a specific event, containing alternative dates. Sometimes it is not clear, which one of these accounts (and thus dates) is true. It is important to see that the alternative dates themselves can be also vague because the accounts are themselves written in

[4]Of course, because the Dark Ages is a very subjective notion, it is just a possible definition for the sake of example which can be definitely debated historically.

a natural language. Considering our example, the birth of Charlemagne is a typical example of an uncertain date. There are three generally accepted possibilities. Two of the possible dates are known precisely: 742-04-02 (probability 0.3) and 747-04-01 (probability 0.1). The third (and most probable, with probability 0.6) possibility is that Charlemagne was born sometime during the 747-04-15 – 748-04-01 period. Here it is easy to see that this specification is both uncertain and vague.

At the highest level, we have **subjectivity**. In the case of complex events, it makes sense to speak of transition periods, which are somewhat still relevant to the complex event but not completely. The beginning and the end of these periods are usually marked by specific events and the dates of these events can be vague and also uncertain. In our example, the fall of the Western Roman Empire definitely marks the beginning of the Dark Ages. The birth of Charlemagne marks the beginning of the transition period, which leads out of the Dark Ages. This transition period we consider only partially relevant to the “Dark Ages”, with a relevance value of 0.7. Finally, the coronation of Charlemagne as Emperor on 0800-12-25 definitely ends the “Dark Ages”.

The hierarchy of the imperfection aspects is shown in Figure 6.2, using the example of the Dark Ages.

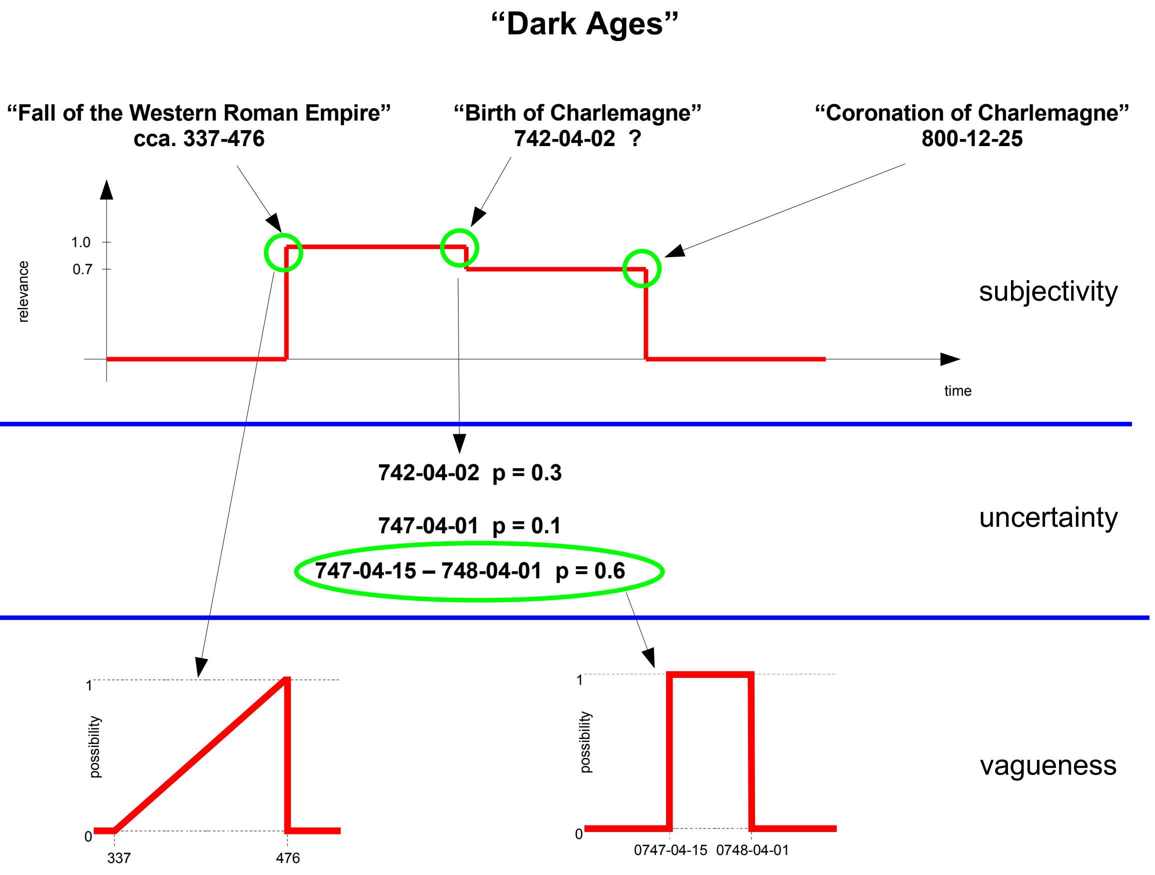


Figure 6.2: The hierarchy of imperfection aspects on the example of Dark Ages

6.4 Related Work

There is a significant amount of approaches for representing temporal information in the areas of temporal reasoning and temporal databases. Most of these, however, consider only classical time intervals (or time points) and do not deal with any form of temporal imperfection.

There are some approaches, however, which provide some solutions for handling imperfect temporal information. Most of them model uncertainty or vagueness with possibility or probability distributions on interval endpoints.

In the area of temporal databases [EJS98], the approaches [DRS01, DS98] define the probability distribution of each endpoint of crisp time intervals. However, most of the temporal specifications are rather vague than uncertain and as was described in Section 6.2.2, possibility theory is considered as more suitable for modeling such specifications.

Because of that, most approaches in temporal reasoning for modeling imperfect temporal knowledge use possibility distributions expressed as fuzzy sets to represent uncertainty. Dubois and Prade [DP89] propose an approach where endpoints of a fuzzy interval are modeled as fuzzy numbers. Further, they use possibility theory [Zad78, DP86] to calculate time points which are possibly or necessarily between the two fuzzy endpoints. They also provide fuzzy extensions of Allen's interval algebra and some basic inference mechanisms.

Kurutach [Kur95] also proposes using fuzzy numbers representing interval endpoints similarly to Dubois and Prade. Moreover, he imposes constraints on the length of intervals. Godo and Vila [GV95] propose using fuzzy sets constraining the length of the time period between intervals. Although this approach is adequate for some problems in the health-care domain, it is inadequate for modeling historical imprecise intervals as it is not possible to specify absolute dates for intervals.

As was shown above, almost all of the approaches for representing uncertain intervals are based on the notion of uncertain or vague interval endpoints. However, as was discussed in Section 3.6, it is not always intuitive to assume that there is a (possibly ill-known) definite starting and ending point for an interval. There are some events, where it makes sense of speaking of "transition periods" in addition of a "core period". E.g., consider the case of the Russian Revolution. Although the core period of this event is only between 1917 and 1919, the periods of 1905–1917 and 1919–1930 are also considered as partially part of this complex event. In this case, the transition periods are much longer than the core period of the event. Using an endpoint-based approach one has to decide which one of the possible endpoints to take, which means losing information.

An interesting approach for representing uncertainty about time-dependent events, which follows a different idea than the works introduced so far is that of Dutta's [Dut88]. He uses the set of known intervals as the universe for fuzzy sets. A fuzzy set representing an event e shows the possibility for each interval i that that event occurs in it. Although this approach is different from the other described approaches in the sense that it is capable of representing fuzzily defined events, it views intervals themselves only as abstract, crisp entities without any further temporal specification. Therefore, it is not applicable to our application scenario.

As a summary, we can tell that although there is a significant body of work in time modeling, the vast majority of related work concentrates on modeling crisp and definite temporal information (for which I use the synonyms “traditional” or “classical” in this chapter). Moreover, existing approaches for modeling imprecise temporal information focus on handling either uncertainty or subjectivity but not both.

Because existing approaches do not fulfill the requirements described in Section 6.1 the development of a new temporal model is needed.

6.5 Temporal Model

6.5.1 The preferred mathematical model of imperfection

Based on the analysis of related work we can state that possibility theory is the prevalent modeling formalism to model temporal imperfection. Possibility distributions alone, however, model only vagueness, while we also have to model uncertainty and subjectivity.

We have seen that mathematically both a probability distribution (representing uncertainty) and a possibility distribution (representing vagueness) is a fuzzy set. Therefore, the theory of fuzzy sets as a mathematical formalism can serve as a unifying theory for all kinds of imperfection. Based on this considerations, I build my time model on the fuzzy set theory.

6.5.2 Intervals or Points?

The most fundamental question in any temporal model is the choice of the basic temporal primitive. The relevant scientific literature mentions two standard primitives [Vil94, Cho94]: *time instants* (or time points, chronons) and *time intervals*. A temporal model can be based on either or on both of them.

There is an ongoing debate in the literature about which primitive is more appropriate, with no clear winner. While time instants are more commonly used in the temporal database research community [EJS98], time intervals or mixed approaches are more popular in the artificial intelligence (AI) community [Vil94]. In [Vil94] it has been argued that the choice of the basic primitive mostly depends on the application requirements. I believe that time intervals are closer to human intuitive perception of time. Instants can always be viewed as time intervals if the granularity of time dimension is sufficiently increased. Further, intervals lend themselves to an intuitive generalization to the fuzzy case.

6.5.3 Continuous vs. Discrete

Although there are some good arguments in the temporal database literature (e.g., [MS91]) for using a discrete time model, most of the approaches in AI use the continuous model, as it fits well to the choice of intervals as the basic primitive.

Under a continuous temporal model, each natural language “time point” translates to an interval in the temporal model. Therefore, I choose the unbounded, continuous time line $\mathcal{T}$ as the basis for defining time intervals which is isomorphic to the set of real numbers. I.e., there exists a natural total ordering among elements of $\mathcal{T}$. Elements of $\mathcal{T}$ are termed as “time points” in this work. To anchor this abstract time line to a real calendar system, I choose the zero time point t_0 to match to the zero point of the Gregorian calendar, measured by the Greenwich Mean Time (GMT).

6.5.4 Time Intervals as Fuzzy Sets

In the following presentation, I assume that we need to represent an abstract interval i when a historical event happens. Let i^- denote its abstract, ill-known starting point and i^+ its abstract, ill-known ending point. As discussed in Section 3.6, intervals of historical events cannot be defined precisely. I call such an interval *imperfect* and model it by means of a fuzzy set **I**, defined by its membership function $\mathbf{I} : \mathcal{T} \to [0, 1]$.

$\mathbf{I}(t)$ represents our confidence level that t is in i. If $\mathbf{I}(t) = 0$, we are completely confident that t is not in i; if $\mathbf{I}(t) = 1$, we are completely confident that t is in i.

I term such a fuzzy set representing an abstract interval as a *fuzzy interval*. $\mathcal{I}$ denotes the set of all fuzzy intervals.

Fuzzy intervals are capable of representing imprecision caused by all of the three special properties of historical knowledge (vagueness, uncertainty or subjectivity) in a unifying way. It is because the possibility to express partial confidence of the membership of some time point t in i allows us the express the imperfection of the accounts about the interval i, regardless of the actual nature of imperfection. E.g., Figure 6.3 shows the fuzzy interval for the “Dark Ages”, which contains all three kinds of imprecision, as was discussed before[5].

I do not pose any constraints on the fuzzy intervals except the requirement that they should be convex. I.e., the abstract interval i, which is represented by the fuzzy interval, should be continuous. Although some events occur at time intervals which are not continuous (e.g., “Poland exists as a country”), they can be represented as a set of subevents denoting continuous parts of the original event (e.g. “Poland exists for the first time”, “Poland is divided among the Russian Empire, the Habsburg Empire and Prussia/Germany” and “Poland exists after the First World War”).

Although it is not a requirement, usually fuzzy intervals will be normalized because in most cases, we are absolutely certain that at least some periods are definitely part of the abstract interval i. Of course, there are some exceptions imaginable. E.g., the life period of King Arthur because we are not even sure whether King Arthur existed at all.

In the case of normalized convex fuzzy sets, their support and core are continuous[6].

Further, I assume that all time points of i must be in the support of **I** (denoted as $S_\mathbf{I}$). I.e., if $t \notin S_\mathbf{I}$, then we are certain that $t \notin i$. I also assume that all of the time points in the core of **I** (denoted as $C_\mathbf{I}$) are really members of i. I.e., if $t \in C_\mathbf{I}$, then we are certain that $t \in i$. This

[5]The exact algorithm for constructing this interval is discussed in the next section.
[6]The core of a non-normalized fuzzy set is empty.

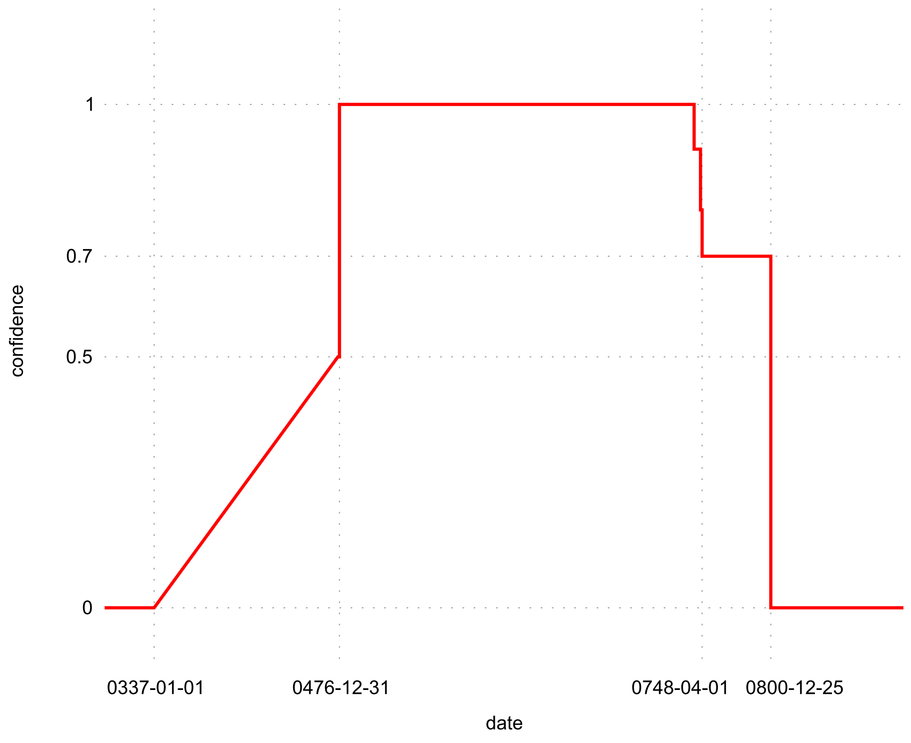


Figure 6.3: The fuzzy temporal interval of “Dark Ages”

can also be written as $C_{\mathbf{I}} \subseteq i \subseteq S_{\mathbf{I}}$. The relation between the core and the support of a fuzzy interval $\mathbf{I}$ and the abstract interval i is shown in Figure 6.4.

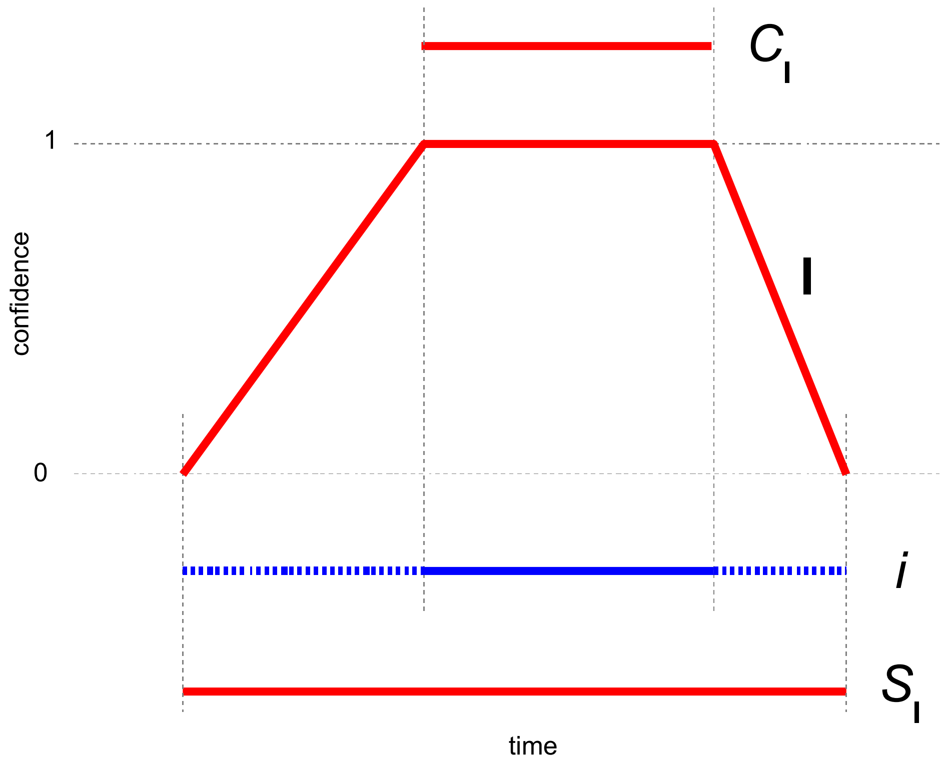


Figure 6.4: Relation between the fuzzy interval and the abstract interval

6.6 Creating fuzzy time intervals

Although we have now a temporal model that can represent all aspects of temporal imperfection, it is unrealistic to expect that domain experts will directly provide us with fuzzy temporal intervals. This is especially true in a domain like history, where the experts do not have a mathematical background. It is therefore very important to discuss how to obtain such fuzzy temporal intervals in a real-world scenario.

6.6.1 Creating intervals using statistical distribution

A common technique to create fuzzy sets is to collect some statistical information about how many percent of domain experts claim that a specific proposition is true or false [DP86]. E.g., if 90% of the experts say that the date 1939-09-01 belongs to the complex event "Second World War" then the membership value of the 1939-09-01 time point in the fuzzy temporal interval representing the Second World War is 0.9.

Using this heuristics, even the World Wide Web can be exploited to automatically generate fuzzy time intervals. Such an approach is described by Schockaert [Sch05]. The main idea of his approach is to crawl the Web for statements in the form "XXX started in YYY", "XXX ended in YYY" etc. for a specific event XXX. He uses the Google search engine, executes such queries and analyzes the results. After the analysis, he is able to extract candidate starting and ending dates for a specific event, and also the frequency distribution of the dates. He applies various heuristics on this frequency distribution to automatically construct the fuzzy time interval.

Although this approach is described for the fuzzy time model proposed by Schockaert and his colleagues [SCK, SCK05], the same procedure could also be applied for my model. However, I prefer a manual approach because I see some problems with the statistics-based approach.

First, because fuzzy intervals are very subjective, there is a danger that an interval generated based on a general statistics will not meet the intuition of domain experts or users. E.g., on the Internet there are sources which claim that the Second World War ended with the Reunification of Germany. Although most of the historians will not agree with this statement, a purely statistical approach, such as the one described here, is sensitive to such outliers. On the other hand, if domain experts are not available, Schockaert's approach provides a simple solution to create fuzzy time intervals.

Second, it is important to see that an interval is not only imperfect because the definition of complex events are subjective but even the mental model of *one* expert is imperfect. On the one hand, uncertainty and vagueness comes from the historical sources and no expert can eliminate that type of imperfection. On the other hand, no true historian would name a specific date as a starting or ending point of a complex event. They rather think in "transition periods", as was described. Therefore, even if we ask one historian, she will not give us a definite time interval. Consequently, it is useful to provide an approach that can capture the mental model of one expert about a specific fuzzy interval. For that purpose, a statistics-based approach is clearly not an option.

My approach is to provide a graphical tool for domain experts where they can specify the different aspects of imperfection separately, and the tool constructs the fuzzy time interval automatically, based on this information. The advantage of this solution in comparison with the statistics-based approach is that in theory it is enough to have just one historian to define the fuzzy intervals. The clear drawback of the approach is the manual effort needed to create the intervals.

6.6.2 The algorithm for the construction of fuzzy intervals

The hierarchy described in Section 6.3 allows the domain experts to build a fuzzy time interval step-by-step. They start by specifying the vagueness level, proceed by specifying uncertainty, and finally build a complex interval by specifying the subjectivity aspect.

Before I explain the algorithm to construct fuzzy intervals, I have to introduce some terminology. A *probability point* represents one point in a probability distribution. A probability point itself can be either crisp (one point on the timeline), or vague. In the latter case one probability point is represented by a possibility distribution, i.e., a fuzzy set. Probability points that form a probability distribution are grouped together in a *subjectivity point*. A subjectivity point is an abstract time point, where the subjective degree of happening of a complex historical event changes. E.g., the "birth of Charlemagne" event is such a subjectivity point, which changes the subjective degree of relevance from 1.0 to 0.7, for the interval of the Dark Ages. A subjectivity point can be also a crisp point, if it consists of only one probability point, which is not vague. E.g., this is the case for the coronation of Charlemagne, where we know the exact date: 0800-12-25. Alternatively, a subjectivity point is itself a complex fuzzy set. The “birth of Charlemagne” or the “fall of the Western Roman Empire” events form such subjectivity points. (See also Figure 6.2.)

As the experts specify only probability and subjectivity points, we need an algorithm which combines the three aspects of imperfection. We have already seen in Section 6.2 that mathematically it is not a problem to combine possibility and probability distributions with fuzzy sets because all of the three theories can be represented mathematically as fuzzy sets. However, epistemologically there are big differences between those theories. Therefore, it is not possible to simply combine a fuzzy set representing a probability distribution with a fuzzy set representing degrees of membership but a sophisticated algorithm is needed which considers the epistemological meaning of the fuzzy sets.

I provide here such an algorithm, which takes a set of subjectivity points and their membership values as an input and creates a fuzzy set representing all three kinds of imperfection together. As was already discussed, this combined fuzzy set represents the combined confidence of the expert that a specific time point belongs to the abstract interval of a complex historical event.

The algorithm consists of two major logical steps. First, the fuzzy set for each subjectivity point is created. Later, the fuzzy sets of the individual subjectivity points are combined to form the final fuzzy set representing the abstract interval. These two steps are described below in detail.

Constructing fuzzy representation of a subjectivity point

1. For all uncertainty points of the subjectivity point do:

   a) Scale underlying fuzzy set with the probability of the uncertainty point. E.g., in Figure 6.5 the scaled fuzzy set of the 0747-04-15 – 0748-04-01 probability point is shown, which is part of the “Birth of Charlemagne” subjectivity point.

   b) Calculate the *positive extension* of the scaled fuzzy set (representing the confidence that a time point is after or equal the abstract point represented by the fuzzy set). If $\mathbf{I}_{i,j}(t)$ denotes the original fuzzy set of the j^{th} uncertainty point of the i^{th} subjectivity point, then $\mathbf{I}^{+}_{i,j}(t)$ denoting the positive extension is calculated as:

   $$\mathbf{I}^{+}_{i,j}(t) = \sup_{s \leq t} I_{i,j}(s) \tag{6.10}$$

   E.g., the positive extension of the 0747-04-15 – 0748-04-01 probability point is shown in Figure 6.6.

   c) Calculate the *negative extension* of the scaled fuzzy set (representing the confidence that a time point is before the abstract point represented by the fuzzy set). If $\mathbf{I}_{i,j}(t)$ denotes the original fuzzy set of the j^{th} uncertainty point of the i^{th} subjectivity point, then $\mathbf{I}^{-}_{i,j}(t)$ denoting the negative extension is calculated as:

   $$\mathbf{I}^{-}_{i,j}(t) = \sup_{s > t} I_{i,j}(s) \tag{6.11}$$

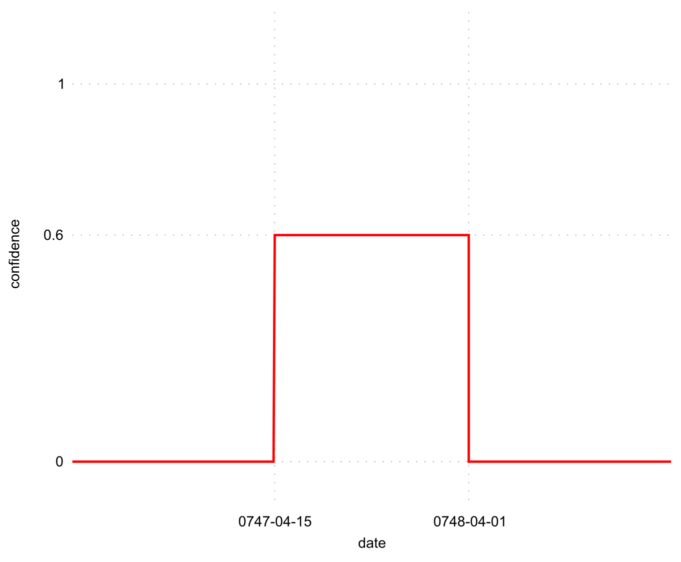


Figure 6.5: The scaled down fuzzy set of 0747-04-15 – 0748-04-01

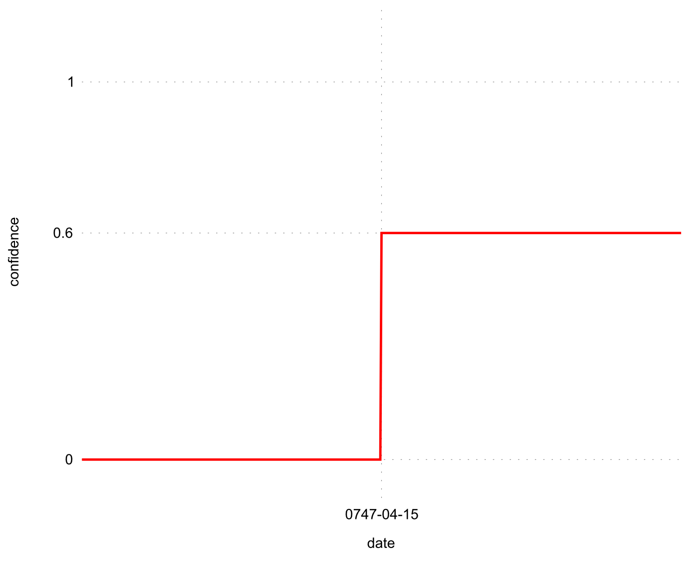


Figure 6.6: The positive extension of 0747-04-15 – 0748-04-01

E.g., the negative extension of the 0747-04-15 – 0748-04-01 probability point is shown in Figure 6.7.

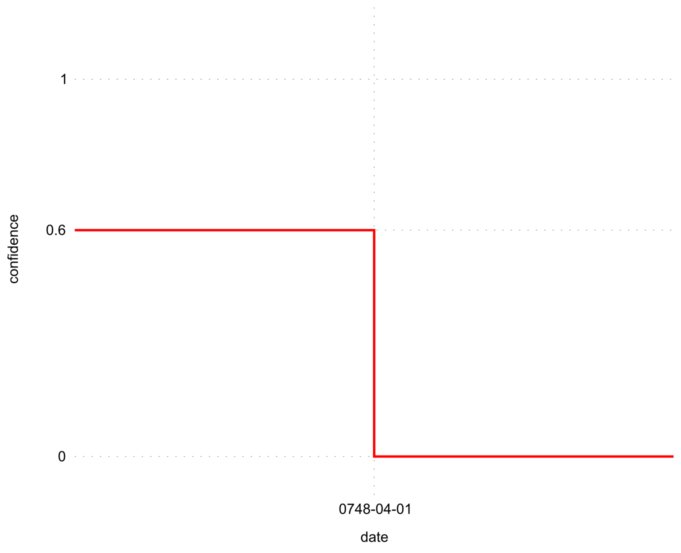


Figure 6.7: The negative extension of 0747-04-15 – 0748-04-01

2. Calculate the positive extension of the whole subjectivity point (denoted by $\mathbf{I}_i^+(t)$ for the i^{th} subjectivity point) by combining the positive extensions of the uncertainty points by summing their values.

$$\mathbf{I}_i^+(t) = \sum_{j=1}^{n} \mathbf{I}_{i,j}^+(t) \tag{6.12}$$

E.g., the positive extension of the whole “Birth of Charlemagne” subjectivity point is shown in Figure 6.10. For reference, also the positive and negative extensions of the 0742-04-02 and 0747-04-01 probability points are displayed in Figure 6.8 and Figure 6.9.

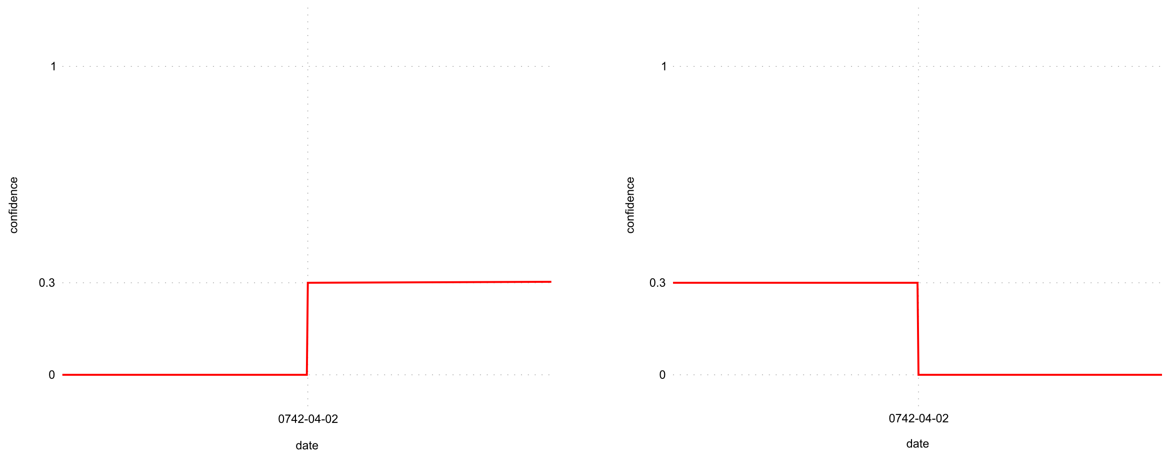


Figure 6.8: The positive and negative extensions of 0742-04-02

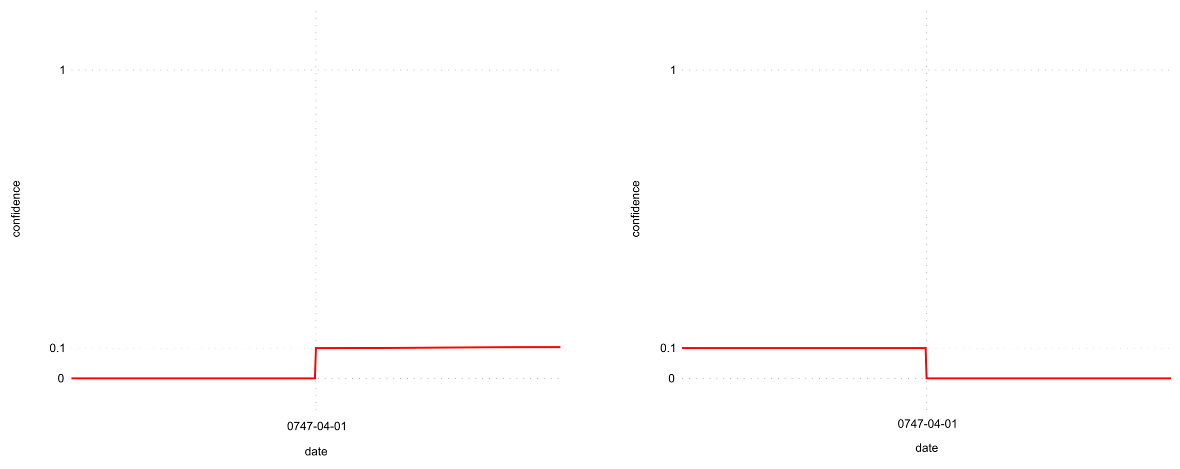


Figure 6.9: The positive and negative extensions of 0747-04-01

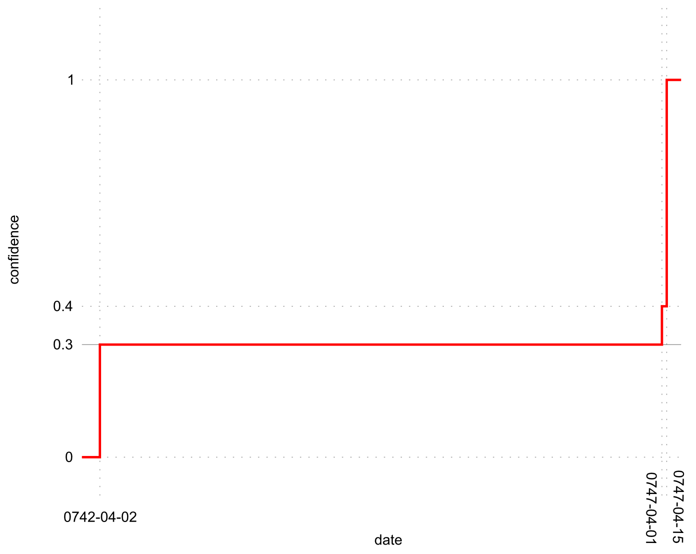


Figure 6.10: The positive extension of “Birth of Charlemagne”

3. Calculate the negative extension of the whole subjectivity point (denoted by $\mathbf{I}_i^-(t)$ for the i^{th} subjectivity point) by combining the negative extensions of the uncertainty points by summing their values.

$$\mathbf{I}_i^-(t) = \sum_{j=1}^{n} \mathbf{I}_{i,j}^-(t) \tag{6.13}$$

   E.g., the negative extension of the whole “Birth of Charlemagne” subjectivity point is shown in Figure 6.11.

4. Calculate the fuzzy set representing our confidence level that a specific time point is after and not before the subjectivity point (denoted by $\mathbf{C}_i(t)$) as:

$$\mathbf{C}_i(t) = \frac{\mathbf{I}_i^+(t) + (1 - \mathbf{I}_i^-(t))}{2}$$

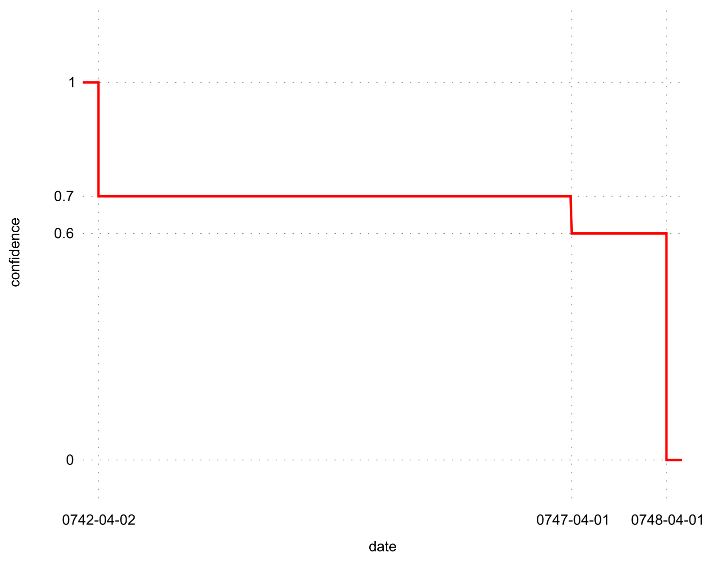


Figure 6.11: The negative extension of “Birth of Charlemagne”

E.g., the “after and not before” fuzzy set for the “Birth of Charlemagne” subjectivity point is shown in Figure 6.12.

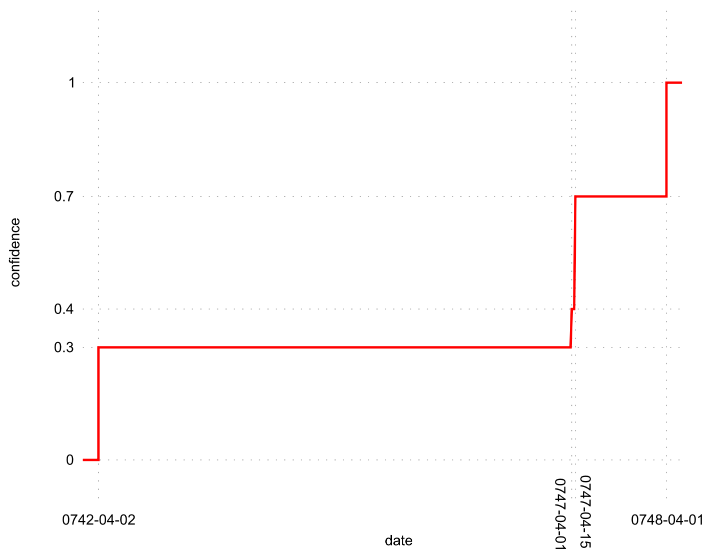


Figure 6.12: The “after and not before” set of “Birth of Charlemagne”

5. Calculate the portion of the final fuzzy set defined by the i^{th} subjectivity point (denoted by $\mathbf{X}_i(t)$) by combining the values before and after the subjectivity point.

$$\mathbf{X}_i(t) = (1 - \mathbf{C}_i(t))B_i + \mathbf{C}_i(t)A_i,$$

where B_i denotes the subjective value before the i^{th} subjectivity point and A_i denotes the value after the i^{th} subjectivity point.

E.g., the portion of the final fuzzy set defined by the “Birth of Charlemagne” subjectivity point is shown in Figure 6.13.

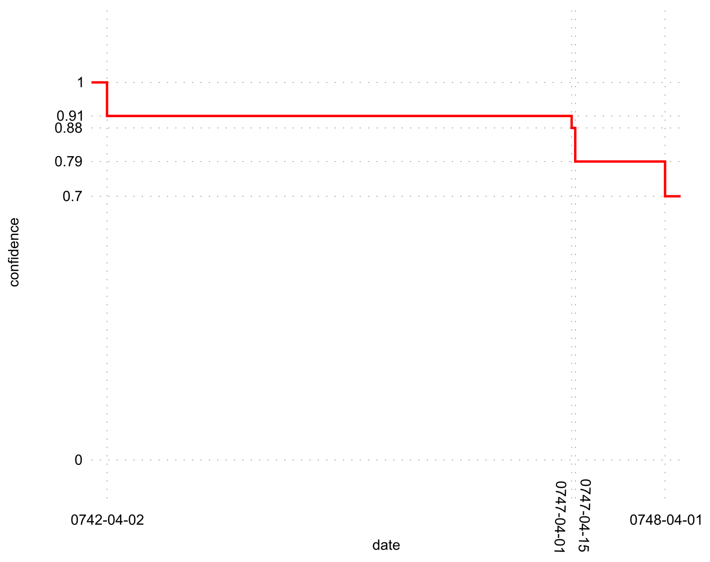


Figure 6.13: The portion defined by “Birth of Charlemagne”

Constructing fuzzy interval from subjectivity points

After we have the fuzzy sets for the individual subjectivity points, it is possible to construct the final fuzzy interval. We construct the fuzzy set representing the final fuzzy interval $\mathbf{X}(t)$ that combines all three aspects of imperfection by gluing the $\mathbf{X}_i(t)$ pieces together. Let c_i^- denote the first point of the support of $\mathbf{C}_i(t)$ (i.e., the first point that has a non-zero membership value). We define pairwise disjoint validity intervals for the $\mathbf{X}_i(t), 1 \geq i \leq n$ pieces, which are shown in Table 6.3.

Table 6.3: Validity interval definitions

Piece	*Validity interval*
$X_1(t)$	$(-\infty, c_2^-)$
$X_2(t)$	$[c_2^-, c_3^-)$
$\vdots$	
$X_i(t), i > 1$ and $i < n$	$[c_i^-, c_{i+1}^-)$
$\vdots$	
$X_n(t)$	$[c_n^-, +\infty)$

If v_i denotes the i^{th} validity interval, $\mathbf{X}(t)$ is calculated as:

$$\mathbf{X}(t) = \mathbf{X}_i(t) \text{ if } t \in v_i \tag{6.14}$$

Finally, the last remaining question is how to define B_i and A_i for the specific $\mathbf{X}_i(t)$ pieces. The A_i values are given by the user using the GUI. The B_i values are defined as:

$$\begin{aligned} B_i &= \mathbf{X}_{i-1}(c_i^-) \text{ if } i \geq 2 \\ B_1 &= 0 \end{aligned} \tag{6.15}$$

E.g., the whole fuzzy set for the Dark Ages is shown in Figure 6.16. For reference, the portions defined by the "Fall of the Western Roman Empire" and the "Coronation of Charlemagne" are shown in Figure 6.14 and in Figure 6.15.

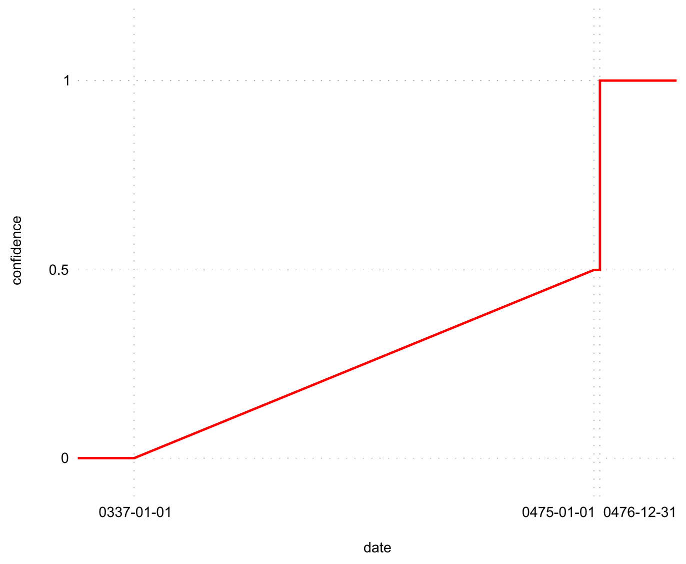


Figure 6.14: The portion defined by "Fall of the Western Roman Empire"

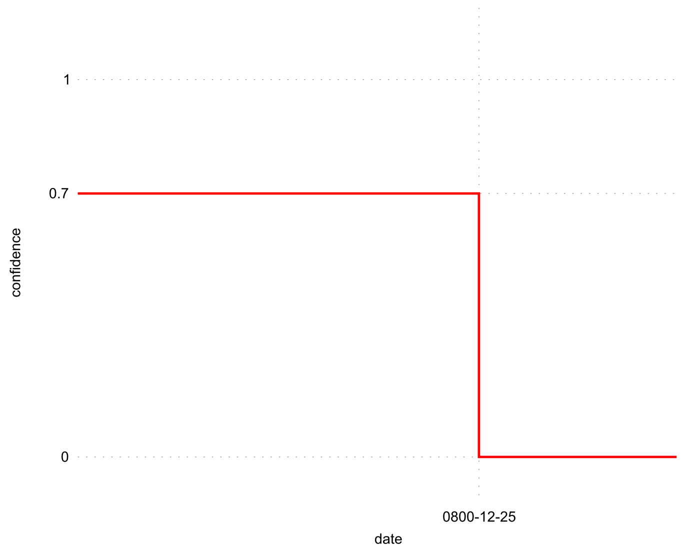


Figure 6.15: The portion defined by "Coronation of Charlemagne"

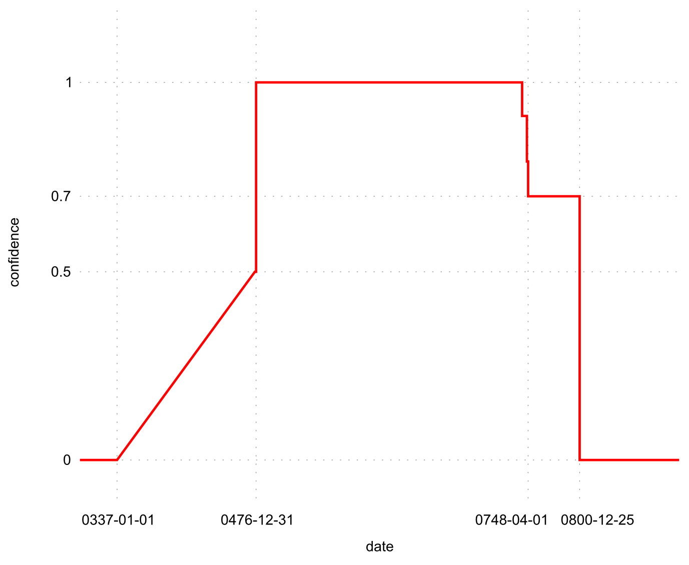


Figure 6.16: The fuzzy temporal interval of “Dark Ages”

Discussion

The introduced algorithm generates a fuzzy interval from an arbitrary combination of vagueness, uncertainty and subjectivity specifications. It is important to note, however, that at specific steps of the algorithm some application-dependent, ad-hoc decisions were made, how to combine the different kinds of theories. This was needed because currently there is no mathematical theory, how probability distributions, possibility distributions and fuzzy sets should be combined. As was already discussed, these constructs are all fuzzy sets mathematically. Still, epistemically they represent very different things. Therefore, it is not trivial, how the different theories should be combined and what is the exact meaning of such combinations. Consequently, other strategies are also possible theoretically, if the results of the algorithm do not seem to be intuitive for the domain experts. In our case, we were happy with the results of the algorithm because it always produced results that seemed to be intuitive.

In the algorithm, examples for such application-specific combinations are the following:

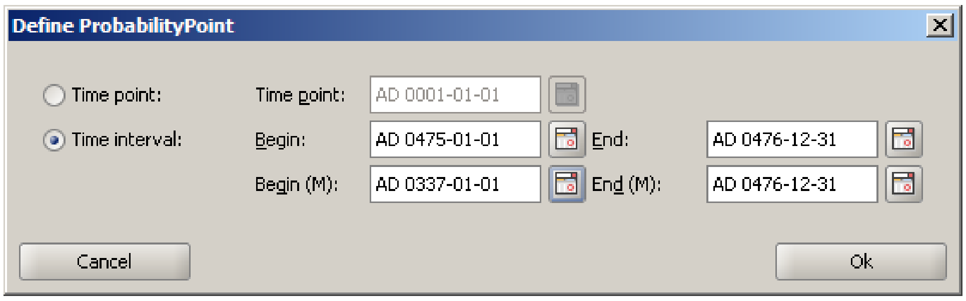
- In step $1.a$, we scaled the values of a possibility distribution with values of a probability distribution. Although mathematically it is a valid operation, epistemically it is problematic to multiply a possibility value with a probability value. The result of such a combination is neither a possibility, nor a probability distribution. Clearly, instead of the simple multiplication, other, more complex, mathematical operations, would be equally valid. E.g., one could multiply with the square of the probability value to emphasize values with a greater probability.
- In step 4, we calculated our confidence that some time point t is after and not before a specific subjectivity point by taking the mean of the “after” and “not before” fuzzy sets. Other mean operations, such as the geometric mean or the harmonic mean would also be possible. It would also be possible to take only one of the two sets into consideration (e.g., only the “after” set) because the role of the $C_i(t)$ value is basically to smooth the transition between two subjectivity values.

- Finally, when we combined the subjectivity point intervals, we simply defined crisp validity intervals of the $X_i(t)$ parts based on their supports. A more complicated combination algorithm that considers also overlapping $X_i(t)$ areas and constructs some transition in such areas (similarly to step 4) would also be possible.

6.6.3 GUI of the fuzzy interval construction tool

The described algorithm was implemented in Java as a stand-alone fuzzy interval construction tool. This tool allows domain experts to follow the described step-by-step construction process.

Experts start by specifying the probability points of a subjectivity point. As was described, a probability point can be either a simple timepoint, or a possibility distribution. To keep things simple, experts can specify a very simple type of possibility distribution with the tool, which consists of a “core period” and a “marginal period” (see Figure 6.17). This effectively creates a distribution with a trapezoid shape. E.g., Figure 6.18 shows the distribution of the probability point for the “Fall of the Western Roman Empire”, which happens gradually during the period from 337 to 476, as was already discussed. It is modeled here so that the core period is from 0475-01-01 to 0476-12-31, and the marginal period is from 0337-01-01 to 0476-12-31.


Define ProbabilityPoint

Time point: [✓ Time interval:]

Time point: AD 0001-01-01

Begin: AD 0475-01-01 End: AD 0476-12-31

Begin (M): AD 0337-01-01 End (M): AD 0476-12-31

Figure 6.17: The dialog to specify a probability point

As the next step, the tool allows experts to specify the probability distribution for a group of probability points. The software validates the probability distribution, i.e., checks that the sum of the probabilities is one. E.g., the probabilities for the “Charlemagne is born” event from the example introduced in Figure 6.2 should sum up to one[7].

Finally, the experts can specify the degree of membership of the time points after the subjectivity point in the abstract time interval. The tool automatically constructs the fuzzy interval and also displays it visually. The tool uses the algorithm for constructing the fuzzy interval that was discussed previously in detail. As an example, Figure 6.19 shows the result of specifying the time period for the Dark Ages.

[7]The probabilities in this example are 0.3, 0.1, and 0.6, respectively.

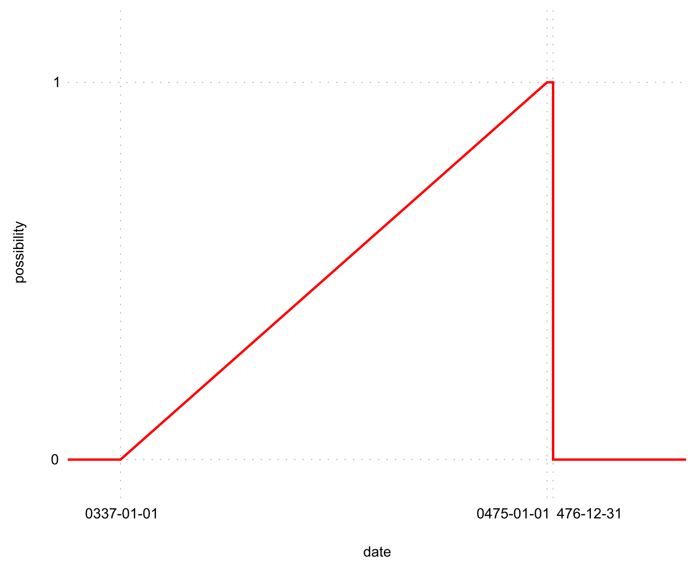
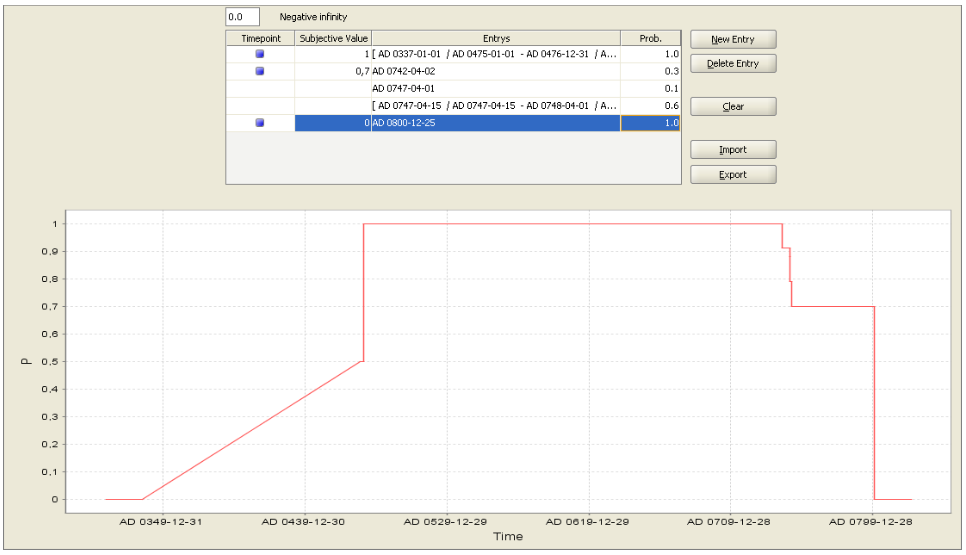


Figure 6.18: Possibility distribution of the “Fall of the Western Roman Empire”



Timepoint	Subjective Value	Entrys	Prob.
●	1	[AD 0337-01-01 / AD 0475-01-01 - AD 0476-12-31 / A...	1.0
●	0,7	AD 0742-04-02	0.3
		AD 0747-04-01	0.1
		[AD 0747-04-15 / AD 0747-04-15 - AD 0748-04-01 / A...	0.6
●	0	AD 0800-12-25	1.0

Figure 6.19: Specifying the Dark Ages with the fuzzy interval creation tool

6.7 Fuzzy Temporal Relations

As was discussed in Section 6.1.1, a proper temporal model should support at least the temporal relations defined by Allen. Therefore, to demonstrate that the introduced fuzzy temporal model is a proper temporal model, I show in this section how to realize the relations from Table 6.1 in my fuzzy model.

It is important to observe that since the intervals are not crisp, the relations between the intervals will not be crisp, either. After all, since the intervals are not exact, we cannot exactly determine whether one interval precedes the other one or not. Hence, given two imperfect intervals i and j and a crisp temporal relation θ, the fuzzy temporal relation Θ will take the fuzzy intervals **I** and **J** and produce a number $c \in [0,1]$, giving the confidence that the classical temporal θ relation holds between i and j.

Extending classical temporal relations to fuzzy sets is not easy, since classical relations in Table 6.1 are defined using interval endpoints. However, for fuzzy intervals the notion of endpoints is meaningless. Therefore, I define fuzzy temporal relations in an alternative way, which is still compatible with the crisp case.

I do this in several steps: first I reformulate the definition of crisp temporal relations using set operations on intervals, thus eliminating references to interval endpoints. In doing so, I introduce several auxiliary unary operators on intervals (e.g., "negative extension"), which create new intervals representing timepoints with some particular relationship to the original interval (e.g., timepoints which are before the last timepoint of the original interval). After that, I extend the definitions of temporal relations to the fuzzy case by providing a fuzzy counterpart of auxiliary operators and by reusing the usual fuzzy set operations.

6.7.1 Defining Crisp Temporal Relations using Set Operations

The basic idea for eliminating references to interval endpoints is the following. First, we can notice that for all $t_1, t_2 \in \mathcal{T}$, if $t_1 < t_2$, then the interval between t_1 and t_2 is not equal to the empty set. This we can be written as

$$t_1 < t_2 \;\;\Leftrightarrow\;\; (t_1, t_2) \neq \emptyset \tag{6.16}$$

Second, if $t_1 = t_2$, then we have to make sure that both intervals (t_1, t_2) and (t_2, t_1) are empty sets, thus expressing that neither t_1 is after t_2 or vice versa. This can be written as

$$t_1 = t_2 \;\;\Leftrightarrow\;\; (t_1, t_2) = \emptyset \;\;\wedge\;\; (t_2, t_1) = \emptyset \tag{6.17}$$

Further, we define several auxiliary unary operators on intervals. The role of these operators is to construct the intervals commonly used in definitions of temporal relations. The following eight operators $<^-$, $\leq^-$, $>^-$, $\geq^-$, $<^+$, $\leq^+$, $>^+$, $\geq^+$, take a crisp interval and construct a new crisp interval containing all of the time points which are (strictly) before or (strictly) after the starting or ending point of the original interval. E.g., $<^-(i) = (-\infty, i^-)$. The definition of these operators is given in Table 6.4.

Now, we are ready to redefine the temporal relations using the ideas presented in (6.16) and (6.17) and the unary operators from Table 6.4. I explain how this is done for the `starts` relation, since its definition uses both endpoint equality and inequality. Other relations are defined similarly and are given in Table 6.5. I did not redefine the `after`, `overlapped-by`, `contains`, `met-by`, `started-by` and `finished-by` relations, as they are simply the inverse of other relations.

We start the redefinition of the `starts` relation by repeating the definition from Table 6.1:

$$i \,\texttt{starts}\, j \;\overset{\text{def}}{=}\; (i^- = j^-) \wedge (i^+ < j^+) \tag{6.18}$$

Table 6.4: Auxiliary unary operators

Operator	*Result*
$<^-$	$<^-(i) = (-\infty, i^-)$
$\leq^-$	$\leq^-(i) = (-\infty, i^-]$
$>^-$	$>^-(i) = (i^-, +\infty)$
$\geq^-$	$\geq^-(i) = [i^-, +\infty)$
$<^+$	$<^+(i) = (-\infty, i^+)$
$\leq^+$	$\leq^+(i) = (-\infty, i^+]$
$>^+$	$>^+(i) = (i^+, +\infty)$
$\geq^+$	$\geq^+(i) = [i^+, +\infty)$

The constraint $i^- = j^-$ can be expressed as (cf. (6.17))

$$\left[(i^-, j^-) = \emptyset\right] \wedge \left[(j^-, i^-) = \emptyset\right] \tag{6.19}$$

which can be written with the help of auxiliary unary operators as

$$\left[>^-(i) \cap <^-(j) \;=\; \emptyset\right] \wedge \left[>^-(j) \cap <^-(i) \;=\; \emptyset\right] \tag{6.20}$$

because we know that

$$(t_1, t_2) = (t_1, +\infty) \cap (-\infty, t_2) \tag{6.21}$$

This last step is needed to eliminate all references to interval endpoints in the definition.

Similarly, the constraint $i^+ < j^+$ can be expressed using (6.16) and (6.21) as

$$>^+(i) \cap <^+(j) \neq \emptyset \tag{6.22}$$

Hence, the `starts` relation can be defined by means of set operations on intervals as

$$i \,\texttt{starts}\, j \overset{\text{def}}{=} \begin{array}{l} >^-(i) \cap <^-(j) \;=\; \emptyset \;\wedge \\ <^-(i) \cap >^-(j) \;=\; \emptyset \;\wedge \\ >^+(i) \cap <^+(j) \;\neq\; \emptyset \end{array} \tag{6.23}$$

Finally, it is important to note that the `intersects` relation has not been derived from the definition in Table 6.1. Instead, simply the fact was used that it expresses the constraint that the intersection of i and j is not empty.

Table 6.5: Transcribed Relations

Temporal Relation	*Definition*
i `before` j	$>^+(i) \cap <^-(j) \neq \emptyset$
i `overlaps` j	$>^-(i) \cap <^-(j) \neq \emptyset \ \wedge$ $<^+(i) \cap >^-(j) \neq \emptyset \ \wedge$ $>^+(i) \cap <^+(j) \neq \emptyset$
i `during` j	$<^-(i) \cap >^-(j) \neq \emptyset \ \wedge$ $>^+(i) \cap <^+(j) \neq \emptyset$
i `meets` j	$>^+(i) \cap <^-(j) = \emptyset \ \wedge$ $<^+(i) \cap >^-(j) = \emptyset$
i `starts` j	$>^-(i) \cap <^-(j) = \emptyset \ \wedge$ $<^-(i) \cap >^-(j) = \emptyset \ \wedge$ $>^+(i) \cap <^+(j) \neq \emptyset$
i `finishes` j	$<^-(i) \cap >^-(j) \neq \emptyset \ \wedge$ $>^+(i) \cap <^+(j) = \emptyset \ \wedge$ $<^+(i) \cap >^+(j) = \emptyset$
i `equals` j	$>^-(i) \cap <^-(j) = \emptyset \ \wedge$ $<^-(i) \cap >^-(j) = \emptyset \ \wedge$ $>^+(i) \cap <^+(j) = \emptyset \ \wedge$ $<^+(i) \cap >^+(j) = \emptyset$
i `intersects` j	$i \cap j \neq \emptyset$

6.7.2 Extending Auxiliary Interval Operators to Fuzzy Intervals

In this section, we extend the auxiliary interval operators to operate on fuzzy intervals. I denote the extended operators with the same symbols as in the crisp case, i.e., as $<^-$, $\leq^-$, $>^-$, $\geq^-$, $<^+$, $\leq^+$, $>^+$, $\geq^+$. Each fuzzy auxiliary operator O will be a function $O : \mathcal{I} \to \mathcal{I}$, i.e, it will take a fuzzy interval and yield another fuzzy interval. The semantics of $O(\mathbf{I})$ should be understood as follows: $O(\mathbf{I})(t)$ gives our confidence that t is in $\omega(i)$ for the corresponding crisp auxiliary operator ω. For example, $<^-(\mathbf{I})(t)$ represents our confidence that t is in $<^-(i)$. In order to make my notation simpler, I will sometimes write $O(\mathbf{I})$ as $\mathbf{I}_O$.

It is important to stress again that the crisp interval i is just an abstract interval that usually cannot be defined (as was explained in Section 6.5.4). This was the motivation for fuzzy interval in the first place. Consequently, we can apply the crisp operators on these crisp intervals only in an abstract sense. The analogy between the crisp intervals and operators and the fuzzy intervals and operators is used only to make the definitions of the fuzzy intervals and operators more intuitive and not to suggest that the crisp intervals could really be defined.

The relation among the crisp and the fuzzy operators is shown in Figure 6.20, on the example of the $>^-$operator.

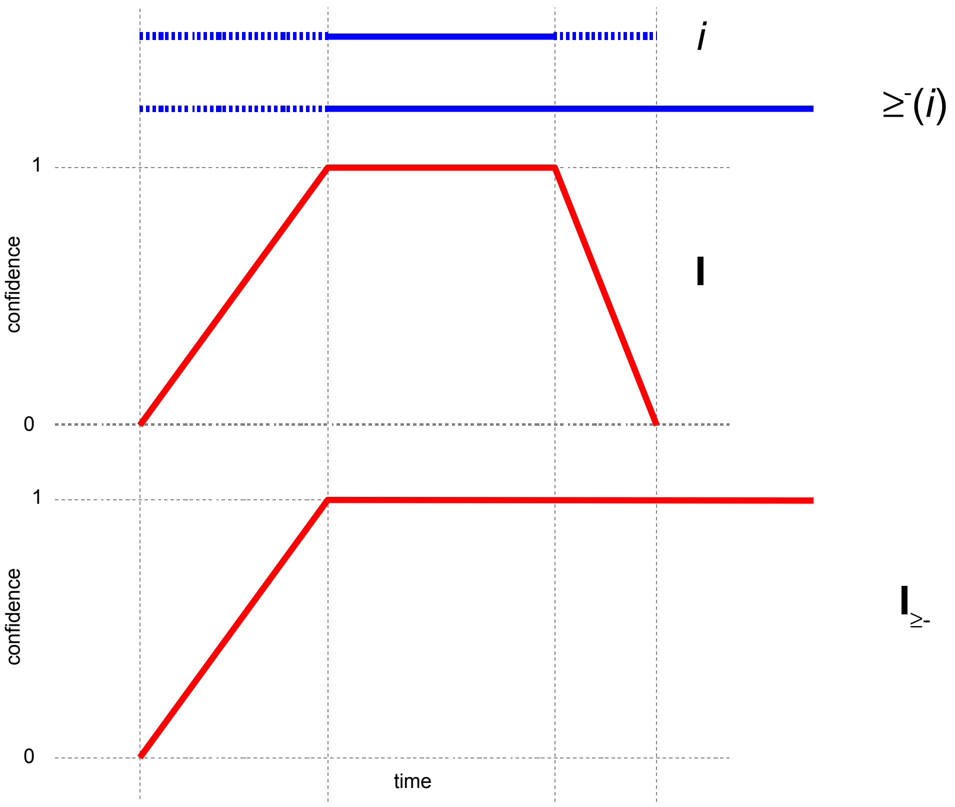


Figure 6.20: Crisp and fuzzy operators

In the rest of this subsection, I will show how to extend the operators $\geq^-$ and $<^-$ to the fuzzy case. The other operators can be extended in a similar manner and their definitions are shown in Table 6.6.

The operator $\geq^-$ should, for some fuzzy interval $\mathbf{I}$ representing interval i, give a fuzzy interval $\mathbf{I}_{\geq-}$, representing the interval $\geq^-(i)$. Let $s_\mathbf{I}^-$ and $s_\mathbf{I}^+$ denote the starting and ending points of $S_\mathbf{I}$ (i.e., of the the support of $\mathbf{I}$). By assumptions from Section 6.5.4, we know that $i \subseteq S_\mathbf{I}$. Therefore, $\mathbf{I}_{\geq-}(t)$ should be 0 for each $t < s_\mathbf{I}^-$ and should be 1 for each $t > s_\mathbf{I}^+$ because we know that each time point before $s_\mathbf{I}^-$ are before i^- and we know that each time point after $s_\mathbf{I}^+$ is after i^+ and therefore also after i^-. For a $t \in S_\mathbf{I}$, we can tell that our confidence that $t \in \geq^-(i)$ should be as big as our confidence that $s \in \geq^-(i)$ for any $s \leq t$. Therefore, I define the operator $\geq^- : \mathcal{I} \to \mathcal{I}$ as follows:

$$\mathbf{I}_{\geq^-}(t) = \begin{cases} 0 & \text{if} \quad t < S_\mathbf{I}^- \\ \sup_{s \leq t} \mathbf{I}(s) & \text{if} \quad t \in S_\mathbf{I} \\ 1 & \text{if} \quad t > S_\mathbf{I}^+ \end{cases} \tag{6.24}$$

The definition of the operator $<^-$ is easy, if we already defined the $\geq^-$ operator. It should be noted that if $t \in <^-(i)$, then $t \notin \geq^-(i)$. Therefore I simply define the $\mathbf{I}_{<^-}$ fuzzy interval as the fuzzy complement of the $\mathbf{I}_{\geq^-}$ fuzzy interval:

$$\mathbf{I}_{<^-}(t) = \mathbf{I}_{\geq^-}^C(t) = 1 - \mathbf{I}_{\geq^-}(t) \tag{6.25}$$

Table 6.6: Fuzzyfied Unary Operators

Operator	*Definition*
$<^-$	$1 - \mathbf{I}_{\geq^-}(t)$
$\leq^-$	$1 - \mathbf{I}_{>^-}(t)$
$>^-$	$\begin{cases} 0 & \text{if } t < S_{\mathbf{I}}^- \\ \sup_{s<t} \mathbf{I}(s) & \text{if } t \in S_{\mathbf{I}} \\ 1 & \text{if } t > S_{\mathbf{I}}^+ \end{cases}$
$\geq^-$	$\begin{cases} 0 & \text{if } t < S_{\mathbf{I}}^- \\ \sup_{s\leq t} \mathbf{I}(s) & \text{if } t \in S_{\mathbf{I}} \\ 1 & \text{if } t > S_{\mathbf{I}}^+ \end{cases}$
$<^+$	$\begin{cases} 0 & \text{if } t < S_{\mathbf{I}}^- \\ \sup_{s>t} \mathbf{I}(s) & \text{if } t \in S_{\mathbf{I}} \\ 1 & \text{if } t > S_{\mathbf{I}}^+ \end{cases}$
$\leq^+$	$\begin{cases} 0 & \text{if } t < S_{\mathbf{I}}^- \\ \sup_{s\geq t} \mathbf{I}(s) & \text{if } t \in S_{\mathbf{I}} \\ 1 & \text{if } t > S_{\mathbf{I}}^+ \end{cases}$
$>^+$	$1 - \mathbf{I}_{\leq^+}(t)$
$\geq^+$	$1 - \mathbf{I}_{<^+}(t)$

The results of applying the $\geq^-$ and $<^-$ operators on the "Dark Ages" fuzzy interval are shown in Figure 6.21 and Figure 6.22.

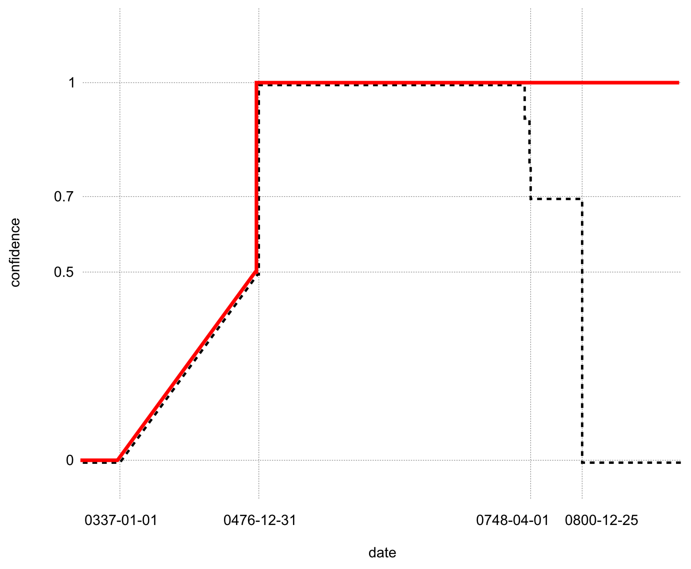


Figure 6.21: Applying $\geq^-$ on the Dark Ages interval

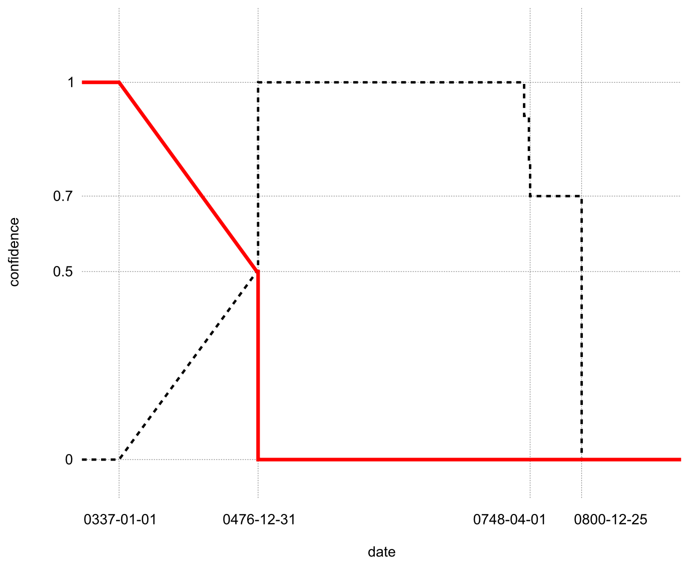


Figure 6.22: Applying $<^-$ on the Dark Ages interval

6.7.3 Constraints using Comparison with the Empty Set

Before we can finally extend the definitions of the temporal relations to fuzzy intervals, we must extend constraints of the form $a \cap b \neq \emptyset$ and $a \cap b = \emptyset$ to fuzzy intervals. I use the following main idea: the value $\sup_t \mathbf{I}(t)$ (i.e., the maximum confidence of membership of any time point in the interval) gives the confidence that some t is in i, i.e., our confidence that i is not empty.

Since fuzzy intersection is expressed using the $\min$ operator (cf. Section 6.2.3), our confidence that $a \cap b \neq \emptyset$ can be represented as

$$\sup_t \min(\mathbf{A}(t), \mathbf{B}(t)) \tag{6.26}$$

Since $a \cap b = \emptyset$ is simply the negation of $a \cap b \neq \emptyset$, our confidence in that this constraint is fulfilled is given as

$$1 - \sup_t \min(\mathbf{A}(t), \mathbf{B}(t)) = \inf_t \max(\mathbf{A}^C(t), \mathbf{B}^C(t)) \tag{6.27}$$

6.7.4 Temporal Relations on Fuzzy Intervals

Now we are ready to extend the definition of the temporal relations to fuzzy intervals based on the transformed definitions from Table 6.5. I define a fuzzy temporal relation Θ as a function $\Theta : \mathcal{I} \times \mathcal{I} \rightarrow [0, 1]$. In other words, a temporal relation takes two fuzzy intervals and gives the confidence that the crisp temporal relation holds between the abstract intervals represented by the respective fuzzy intervals. I denote the fuzzy temporal relations with big letters to distinguish them from their crisp counterparts.

I only discuss the definition of relation `STARTS`(**I**,**J**). Other relations can be defined in a similar way and they are shown in Table 6.7. We start from the definition of the crisp relation `starts`, which was defined in Subsection 6.7.1 as

$$i\,\texttt{starts}\,j \overset{\text{def}}{=} \begin{array}{rcl} >^{-}(i) \cap <^{-}(j) & = & \emptyset \;\wedge \\ <^{-}(i) \cap >^{-}(j) & = & \emptyset \;\wedge \\ >^{+}(i) \cap <^{+}(j) & \neq & \emptyset \end{array} \tag{6.28}$$

After applying the rules for transcribing the constraints (cf. Subsection 6.7.3) we obtain:

$$\begin{array}{rl} \texttt{STARTS}(\mathbf{I},\mathbf{J}) = & \min(\\ & \inf_t \max(\mathbf{I}_{\leq -}(t), \mathbf{J}_{\geq -}(t)), \\ & \inf_t \max(\mathbf{I}_{\geq -}(t), \mathbf{J}_{\leq -}(t)), \\ & \sup_t \min(\mathbf{I}_{>+}(t), \mathbf{J}_{<+}(t))) \end{array} \tag{6.29}$$

Table 6.7: Fuzzy Temporal Relations

Relation	*Definition*
`BEFORE`(**I**,**J**)	$\sup_t \min(\mathbf{I}_{>+}(t), \mathbf{J}_{<-}(t))$
`OVERLAPS`(**I**,**J**)	$\min(\sup_t \min(\mathbf{I}_{>-}(t), \mathbf{J}_{<-}(t)),$ $\sup_t \min(\mathbf{I}_{<+}(t), \mathbf{J}_{>-}(t)),$ $\sup_t \min(\mathbf{I}_{>+}(t), \mathbf{J}_{<+}(t)))$
`DURING`(**I**,**J**)	$\min(\sup_t \min(\mathbf{I}_{<-}(t), \mathbf{J}_{>-}(t)),$ $\sup_t \min(\mathbf{I}_{>+}(t), \mathbf{J}_{<+}(t)))$
`MEETS`(**I**,**J**)	$\min(\inf_t \max(\mathbf{I}_{\leq +}(t), \mathbf{J}_{\geq -}(t)),$ $\inf_t \max(\mathbf{I}_{\geq +}(t), \mathbf{J}_{\leq -}(t)))$
`STARTS`(**I**,**J**)	$\min(\inf_t \max(\mathbf{I}_{\leq -}(t), \mathbf{J}_{\geq -}(t)),$ $\inf_t \max(\mathbf{I}_{\geq -}(t), \mathbf{J}_{\leq -}(t)),$ $\sup_t \min(\mathbf{I}_{>+}(t), \mathbf{J}_{<+}(t)))$
`FINISHES`(**I**,**J**)	$\min(\inf_t \max(\mathbf{I}_{\leq +}(t), \mathbf{J}_{\geq +}(t)),$ $\inf_t \max(\mathbf{I}_{\geq +}(t), \mathbf{J}_{\leq +}(t)),$ $\sup_t \min(\mathbf{I}_{>-}(t), \mathbf{J}_{<-}(t)))$
`EQUALS`(**I**,**J**)	$\min(\inf_t \max(\mathbf{I}_{\leq +}(t), \mathbf{J}_{\geq +}(t)),$ $\inf_t \max(\mathbf{I}_{\geq +}(t), \mathbf{J}_{\leq +}(t)),$ $\inf_t \max(\mathbf{I}_{\leq -}(t), \mathbf{J}_{\geq -}(t)),$ $\inf_t \max(\mathbf{I}_{\geq -}(t), \mathbf{J}_{\leq -}(t)))$
`INTERSECTS`(**I**,**J**)	$\sup_t \min(\mathbf{I}(t), \mathbf{J}(t))$

6.8 Discussion of the fuzzy temporal model

In this section I discuss the model and relations presented in this chapter and demonstrate how they fulfill the requirements from Section 6.1.

6.8.1 Representing Imprecise Information.

As discussed briefly in Section 6.5.4, fuzzy intervals are capable of representing all three causes of impreciseness in history. They represent the net confidence of the history expert about the statement $t \in i$, where the lack of confidence can be caused by any combination of vagueness, uncertainty and subjectivity.

With this approach it is possible to model not only the core period of an event but also transition, development etc. periods, which are only partially relevant to a specific event. This is possible because I define fuzzy intervals directly, without referencing the endpoints. In this sense my approach is superior to the other approaches, which were discussed in Section 6.4.

As examples for fuzzy intervals, possible interpretations of the intervals "late twenties – early thirties" (Figure 6.23), "320? – 330"[8] (Figure 6.24), and "Russian Revolution" (Figure 6.25) are shown. Each of the fuzzy intervals shows one of the special characteristics of historical knowledge, namely vagueness, uncertainty, and subjectivity, respectively.

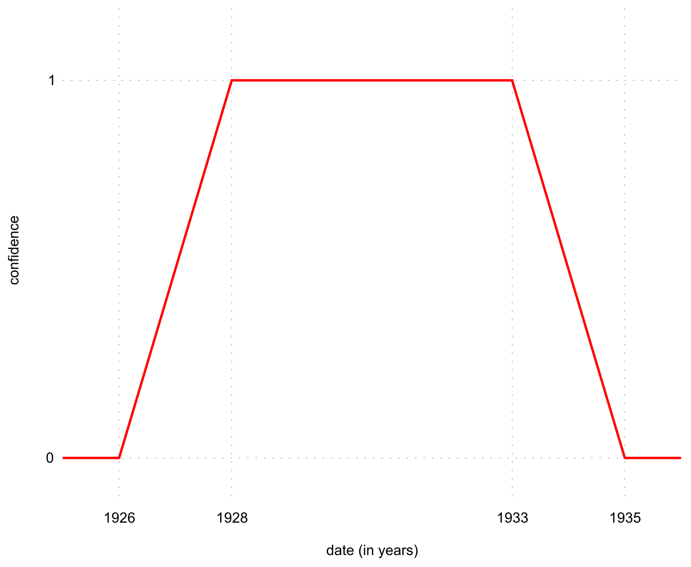


Figure 6.23: Fuzzy interval of "late twenties" – "early thirties" (vagueness)

[8]The beginning of the interval is uncertain, it is assumed that the interval starts in 318 with 0.3 probability and in 320 with 0.7 probability.

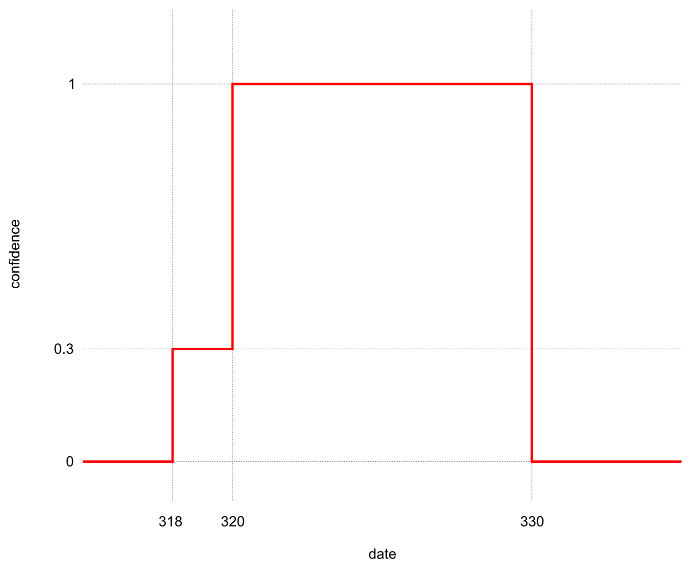


Figure 6.24: Fuzzy interval of 320? – 330 (uncertainty)

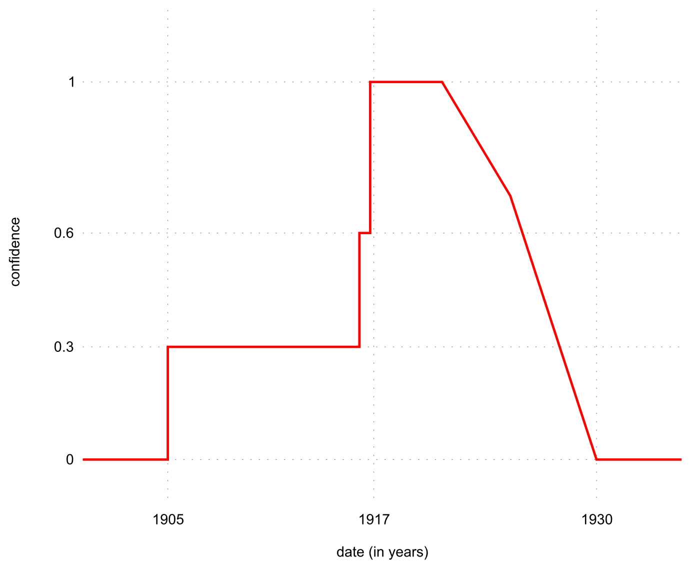


Figure 6.25: Fuzzy interval of “Russian Revolution” (subjectivity)

6.8.2 Compatibility with Crisp Case

It is easy to see that fuzzy relations are natural extensions of the classical ones, as they give the same results on crisp intervals as the classical ones. This is because I defined the fuzzy relations based on the original definitions of the classical relations. Hence, the requirements on the compatibility with the crisp case is completely fulfilled.

6.8.3 Intuitiveness of Fuzzy Relations.

I believe that my fuzzy relations yield intuitive results, where I mean with “intuitive” that the relation yields a result of 1 if we are completely certain that the classical relation exists between the abstract intervals represented by the fuzzy intervals. Moreover the relation should yield 0, if we are certain that this is impossible. A result between zero and one is given if neither of these possibilities are sure.

E.g., in the case of the `BEFORE`(**I**,**J**), it should yield 1, if $S_{\mathbf{I}}$ `before` $S_{\mathbf{J}}$ holds (i.e., we are sure that i `before` j holds) and it should yield 0, if $C_{\mathbf{I}}$ `before` $C_{\mathbf{J}}$ does not hold (i.e. we are sure that i `before` j does not hold). These three cases are shown graphically in Figures 6.26 to 6.28.

Intuitiveness can similarly be checked for other fuzzy relations.

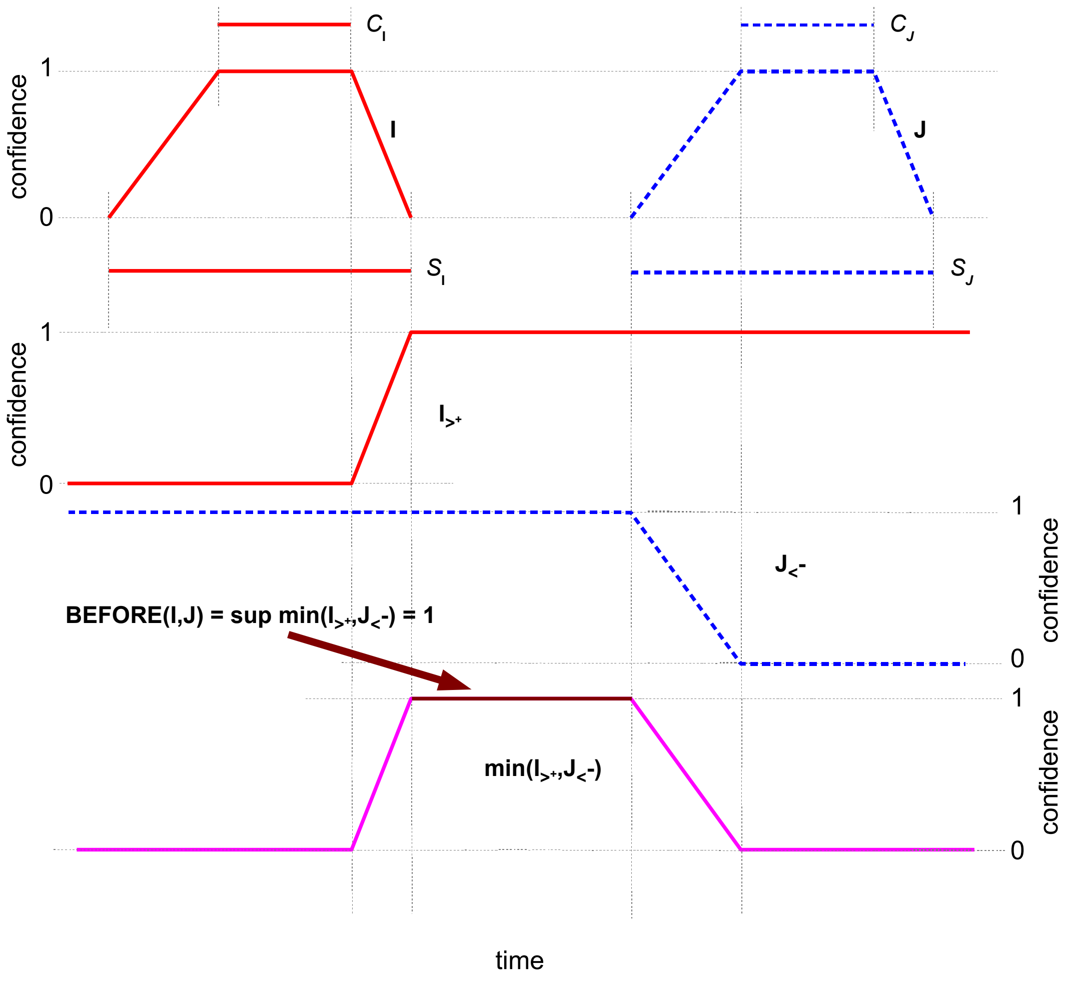


Figure 6.26: BEFORE(**I**,**J**) = 1

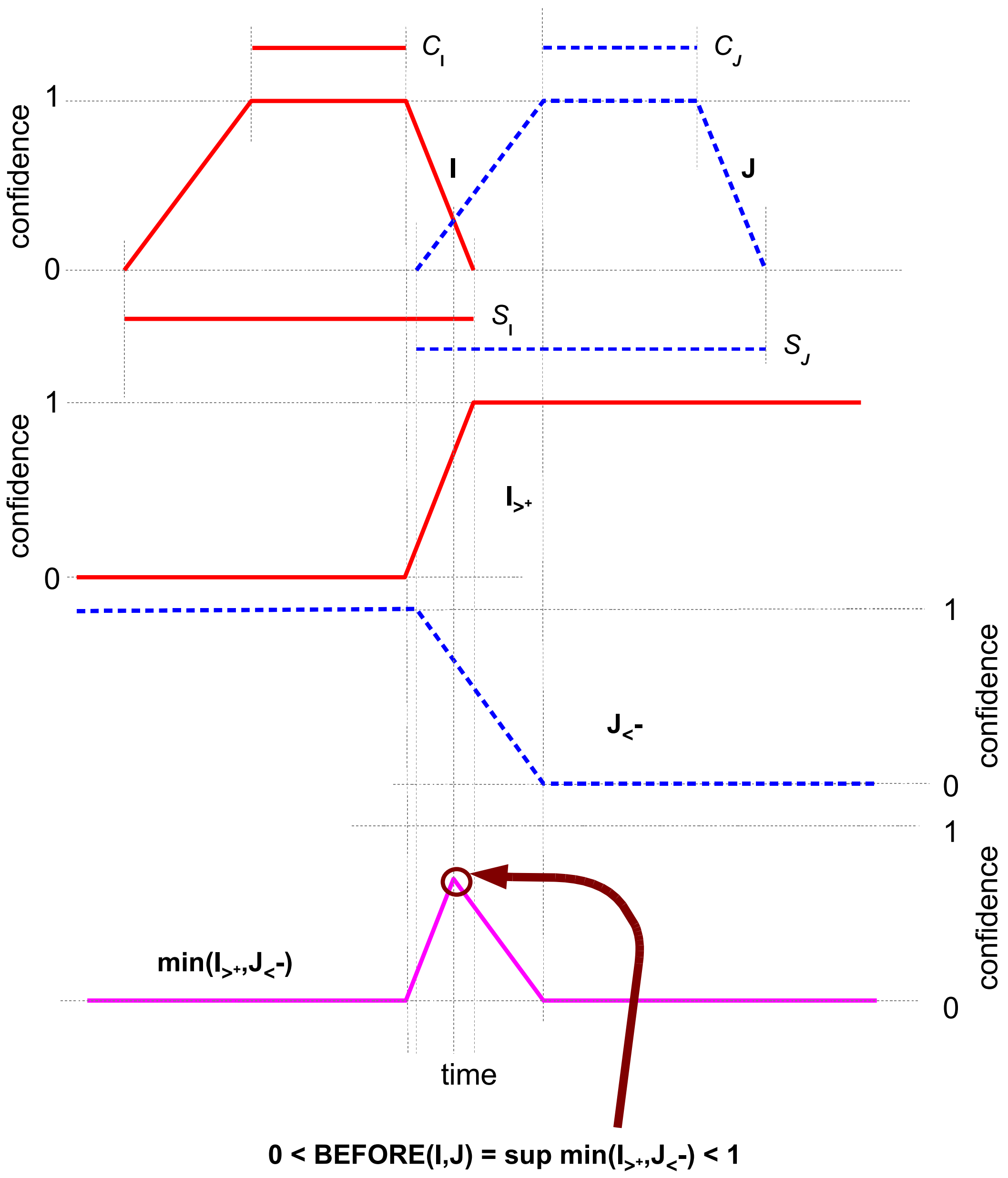


Figure 6.27: $0 < \texttt{BEFORE}(\mathbf{I},\mathbf{J}) < 1$

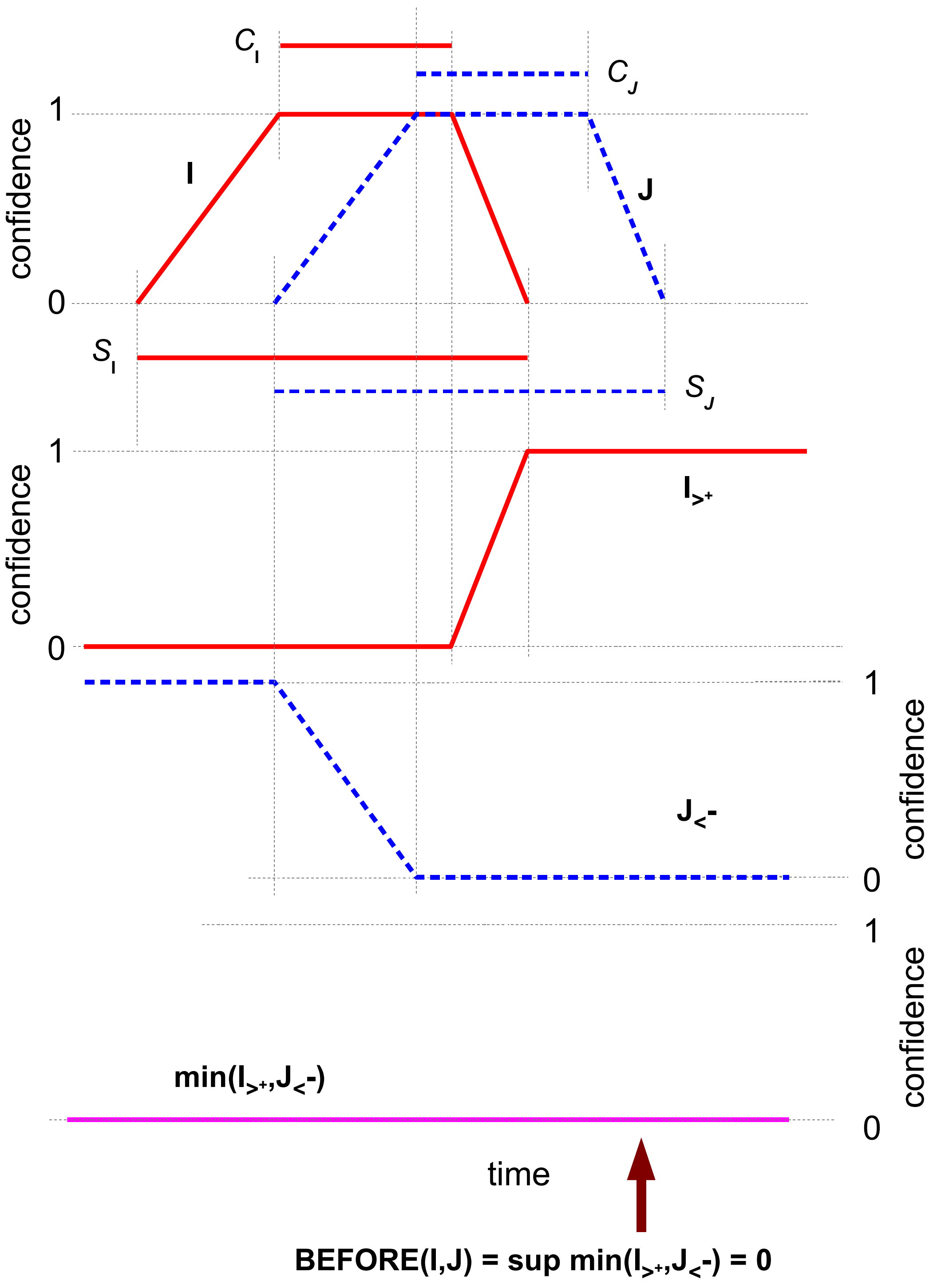


Figure 6.28: BEFORE(**I**,**J**) = 0

6.9 Alternative approaches

The temporal model described here was published in [NM03]. To the best of my knowledge, we were the first to propose a temporal model, which is capable of capturing all aspects of imperfection in historical temporal information[9] and also provides a set of temporal relations, which allows reasoning on these temporal specifications.

Later, alternative approaches for representing fuzzy temporal relations were also proposed, which are discussed here. The alternative approaches also build on the idea to use fuzzy sets directly to encode imperfect temporal intervals. This fact is very encouraging because it shows that the basic idea of my temporal model has been accepted in the research community.

One of the relevant works is that of Ohlbach [Ohl04]. He also proposes to use fuzzy sets to represent whole temporal intervals, instead of starting or ending points of intervals. He also fuzzifies Allen's relations but with a different focus. He proposes to fuzzify the relations themselves also for the crisp case, i.e., to represent relations such as "long before". Operationally, he also uses the positive and negative extensions of fuzzy sets to define interval relations but he chooses to calculate the overlapping area of fuzzy sets to determine the degree to which a specific fuzzy relation holds. It is also a difference that he uses point-to-interval relations as a basic unit of his fuzzy temporal algebra, while I use point-to-point relations as basic units. For our application domain the major disadvantage of Ohlbach's approach is that his fuzzy relations are not compatible with the crisp case. I.e., it is not possible to express the classical crisp before, after, etc. relations with his fuzzy relations. On the other hand, his approach provides much more flexibility to specify different definitions of the various fuzzy relations.

Recently, partly motivated by my work, Schokaert and his colleagues also proposed a time model based on fuzzy time intervals [SCK, SCK05]. They build their representation on fuzzy point-to-point relations. With their approach, they are able to make some statements about the transitivity, reflexivity and symmetry property of their fuzzy relations and they can show that using some specific fuzzy t-norms[10] to define the temporal relations, these properties coincide with the crisp case. In that sense, their model is much more elaborate than my model or the model of Ohlbach. On the other hand, their model is mathematically much more involved than my model and the interval relations can be calculated less efficiently. Because the additional features provided by their model are not needed in our application domain, I use my simpler, more efficient model in this thesis.

6.10 Generality of the approach

The introduced approach to fuzzify temporal relations is general and can be applied to other domains. As an example, I show how to fuzzify spatial relations using the same fundamental idea.

[9]i.e., *uncertainty*, *subjectivity*, and *vagueness*

[10]In fuzzy logic, minimum and maximum are not the only possibilities to define the fuzzy intersection and union operations. The operators which are used to calculate fuzzy intersection are called "t-norms", the operators to represent fuzzy union are called "t-conorms".

In most information systems where spatial data is managed, location information of places is stored with the help of a so-called *bounding shape*. This bounding shape specifies the relevant geographical points of an entity (city, country, river etc.) in a two (sometimes three) dimensional space of geographical coordinates. The typical operation in such information systems is to find stored bounding shapes which have overlapping regions with a reference bounding shape. Such way one can find cities located in a country, countries which a river flows through etc.

Similarly to the case of the presented fuzzy temporal model, where the notion of temporal interval was extended to a fuzzy temporal interval, it is also possible to extend the notion of the bounding shape to the notion of a *fuzzy bounding shape*.

6.10.1 Fuzzy Bounding Shapes

In the following presentation, we assume that we need to represent a crisp bounding shape s inside which an entity is located. A crisp bounding shape has a characteristic function $s(x, y)$ which gives 1 for a specific coordinate if it is part of the bounding box, and 0 if it is not. We denote the set of all possible x coordinates as $\mathcal{X}$ and the set of all possible y coordinates as $\mathcal{Y}$. We denote crisp bounding shapes as s and t and their characteristic function with $s(x, y)$ and $t(x, y)$, respectively.

Similarly to the temporal case, it is possible that we are not completely confident, whether a specific coordinate belongs to the bounding shape of an entity or not. Again, similarly to the temporal case the cause of this lack of confidence can be uncertainty or vagueness[11] but also level of preference[12].

We can say that an entity is clearly located inside an abstract bounding shape but we do not know that bounding shape precisely. We call such a bounding shape *imperfect* and model it by means of a two dimensional fuzzy set $\mathbf{S}$, defined by its membership function $\mathbf{S} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$. $\mathbf{S}(x, y)$ represents our confidence level that (x, y) is in s. If $\mathbf{S}(x, y) = 0$, we are completely confident that (x, y) is not in s; if $\mathbf{S}(x, y) = 1$, we are completely confident that (x, y) is in s. I term such a fuzzy set representing an abstract bounding shape as a *fuzzy bounding shape*. I denote the set of all fuzzy bounding shapes as $\mathcal{S}$ and elements of $\mathcal{S}$ as $\mathbf{S}$ and $\mathbf{T}$.

Of course, similarly to the temporal case we also need to extend crisp spatial relations to the fuzzy case. In present spatial information systems the most widely used relation is the *overlaps* relation on bounding shapes. This relation is defined for the crisp case as:

$$\texttt{overlaps}(s, t) = \begin{cases} 1 & \text{if } \exists x \in \mathcal{X} \text{ and } \exists y \in \mathcal{Y} \\ & \text{such that} \\ & s(x, y) = t(x, y) = 1 \\ & \\ 0 & \text{otherwise} \end{cases} \tag{6.30}$$

[11] i.e., missing information about the location

[12] e.g., a citizen of Karlsruhe may view the neighboring city of Ettlingen as “somewhat part of” Karlsruhe

This crisp overlaps relation can be extended to the fuzzy case easily by providing a fuzzy *OVERLAPS* operation:

$$\texttt{OVERLAPS}(\mathbf{S}, \mathbf{T}) = \sup_{x,y}(\max(\mathbf{S}(x, y), \mathbf{T}(x, y))) \tag{6.31}$$

6.10.2 Bounding Boxes vs. Bounding Shapes

In real-world information systems, bounding shape membership is not defined for each possible (x, y) coordinates for performance reasons but bounding shapes are rather represented with the help of simple geometric forms (rectangular, polygon, ellipse etc.), or as a set of such simple forms. In the case of a simple rectangular shape bounding shapes are called *bounding boxes*.

The advantage of representing bounding shapes with simple geometric forms is the dramatically decreased amount of needed storage space[13] and the increased performance when calculating the results of bounding shape operations[14]. Of course, the drawback of the approach is that the spatial specifications will be less precise.

Similarly to the crisp case, it is also possible to construct fuzzy spatial bounding shapes as a union of simpler fuzzy bounding shapes. E.g., one may define fuzzy bounding shapes using bounding boxes, where each point inside such a sub-bounding box has the same membership value. An example for that is shown on Figure 6.29.

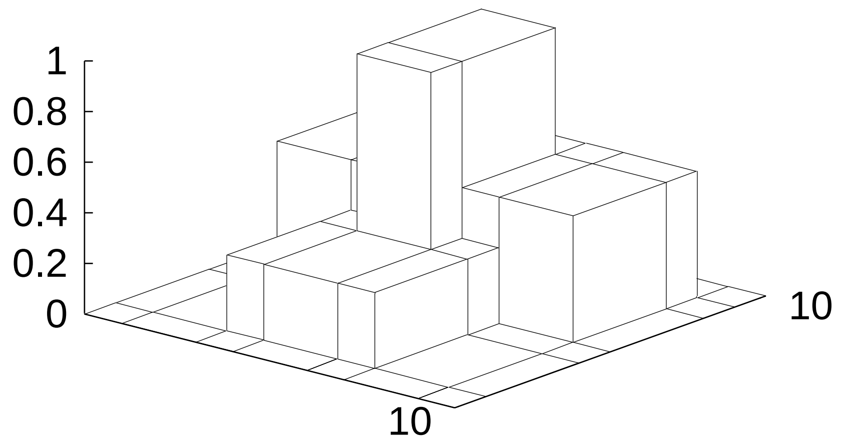


Figure 6.29: An example fuzzy spatial bounding box

[13]e.g., only four real numbers are needed to represent a bounding box

[14]e.g., to determine whether two rectangular areas overlap only several simple real-number comparisons have to be done

6.11 Summary

In this chapter a novel temporal model was introduced. The model is built on the theory of fuzzy sets and represents imperfect temporal intervals as fuzzy temporal intervals. It was shown that this model can represent all aspects of temporal imperfection in a unified model. Later it was discussed how such fuzzy temporal intervals can be created and a user-friendly graphical tool was introduced for this purpose. Finally, the generality of the approach to define fuzzy temporal intervals was demonstrated by defining fuzzy bounding shapes to represent imperfect spatial specifications, using the same fundamental ideas that were applied on time intervals.

The introduced model is a full-featured temporal model that can represent all of the usual temporal relations. Therefore, it is a valuable contribution on its own. Moreover, the model is used in the ontology-supported IR process during the metadata generation step, which will be discussed in Chapter 8 in detail.

Chapter 7

Ontology formalism

We have seen in Section 2.5 that the definition of an ontology is rather vague. Although I have constrained the original definition of Gruber, there are still many possibilities to define an ontology formalism that fulfills the definition. Moreover, during Chapter 3 I identified some additional requirements that an ontology formalism should meet. In this chapter, I analyze these requirements from the ontology formalism point of view. I define and describe an ontology formalism that fulfills all these requirements and that will be used in this thesis.

7.1 Requirements

The first requirement toward an ontology formalism is that it should be an *ontology* according to our definition in Section 2.5. I.e., it should have a "well-defined mathematical interpretation", and should at least be "capable to represent a subconcept taxonomy, concept instances and user-defined relations between concepts". Moreover, to fulfill the DOMAIN IMPERFECTION requirement, the ontology formalism should allow us to represent fuzzy temporal intervals, and relations between such intervals.

A further analysis of history as an application domain reveals that many of the relations are time-dependent. E.g., a person is member of a group or organization only during a limited period of time, or a person has a role only during a limited period of time etc. Moreover, time-dependency can be combined with other common properties of ontology relations, such as symmetry, transitivity and inverse relations.

An example for a time-dependent symmetric relation is the "married with" relation. In a modern society marriage can be broken up, i.e., two persons are married to each other only during a specific time period. Marriage is a symmetric relation, if a husband is married to his wife, the wife is also married to her husband.

An example for a time dependent transitive relation is the "part of" relation between locations. The relation is transitive, e.g., if Strasbourg is part of Alsace, and Alsace is part of France, then Strasbourg is also part of France. On the other hand, the relation is time dependent because the ownership of territories and cities can change. E.g., Alsace changed its owner many times during the history, it was sometimes part of France, sometimes part of Germany.

It is easy to see that time dependent relations can have inverse relations, too. E.g., the "part of" relation can have an inverse relation "has part".

Therefore, we can state as a final requirement that *an ontology formalism should be able to represent symmetric, transitive and inverse relations, both timed and non-timed.*

7.2 Choosing the appropriate formalism

7.2.1 Alternative formalisms

When choosing the appropriate ontology formalism, one should always strive to choose an already available standard formalism to guarantee interoperability with other systems using semantic technology. Based on this principle, the most straightforward choice would be one of the OWL languages: OWL-Lite, OWL-DL or OWL-Full [PSHH04]. Unfortunately because the OWL language family builds on the description logic (DL) formalism [BCM$^+$03], the OWL variants support only binary relations out-of-the-box. To represent timed relations, however, at least ternary relations are needed because a timed relation connects the source and the target of the relation with a time specification.

The standard way to solve the problem of relations with higher arity in the OWL-world is to reify them. Reification means that the relation is transformed into a new instance of a new concept which is connected with the original entities participating in the relation. E.g., instead of specifying a "married with" timed relation connecting two persons, one could specify a new MARRIAGE concept and represent the marriage as an instance of this new concept. I.e., the statement MARRIEDWITH(Bill, Mary, "after 2001-01-01") is transformed to a new instance MARRIAGEOFBILLANDMARY of the MARRIAGE concept which is connected with the instances BILL, MARY and the user-defined datatype representing "after 2001-01-01" (using three binary relations).

The problem is that transitivity of relations is a built-in construct in DL-like formalisms, and cannot be expressed using axioms. This means, if we reify a transitive relation, we loose the possibility to declare that it is a transitive one. Therefore, reification cannot be used in this case, which means that OWL cannot fulfill the requirement to represent timed transitive relations, such as "part of".

Besides OWL another popular ontology formalism is F-Logic [KLW95] which is also used by, e.g., the WSML formalism for modeling semantic web service ontologies [dB05]. This formalism allows one to represent relations of arbitrary arity, therefore the reification problem does not occur. Moreover, all of the relation properties (symmetry, transitivity and inverse of) can be expressed with valid axioms. The only problem which ruled out this formalism was that there were no freely available F-Logic reasoner implementations known to me that supported user-defined datatypes.

7.2.2 A Datalog-based ontology formalism

Finally, I decided to use the Datalog formalism [Ull88, CGT90, Dah96]. Datalog is a formalism for deductive databases. It is relationally complete and in addition it also supports recursion. Datalog can represent relations with arbitrary arity and when I started the implementation of the research prototype, an efficient Datalog reasoner that allowed the definition of user-defined datatypes was already available — the KAON2 reasoner [HMS04].

It must be noted, however, that Datalog itself is not an ontology formalism according to my definition from Section 2.5. In particular, it does not support the usual ontology modeling constructs: concepts, instances and relations. Datalog is a logical formalism, therefore the only modeling constructs are logical predicates and logical axioms (rules). Consequently, it was necessary to design my own mini-ontology language on top of Datalog that axiomatized the semantics of usual ontology constructs.

The modeling constructs of this ontology language are shown in Table 7.1. As it can be seen, the constructs are the "usual" ones that can be found in almost all ontology languages. The only unique feature is the possibility to use temporal relations and freely use relation metaproperties (such as symmetry or transitivity) also on temporal relations. This has two advantages: first, existing ontology editors can be reused to model large parts of the ontology. Second, major parts of ontologies defined in my formalism can be exported to widely used ontology formalisms, such as OWL.

Table 7.1: Ontology modeling constructs

Modeling construct	*Datalog predicate*
Concept	CONC(x)
Instance	INST(x)
"Normal" relation	NREL(x)
Temporal relation	TREL(x)
Attribute	ATTR(x)
Instance of	ISA(i,c)
Domain concept	DOMAIN(r, c)
Range concept	RANGE(r,c)
Symmetric relation	SYMM(r)
Transitive relation	TRANS(r)
Inverse relations	INV(p,r)
"Normal" relation value	NRVAL(r,is,it)
Temporal relation value	TRVAL(r,is,it)
Attribute value	AVAL(r,i,v)
Subconcept of	SUBC(c,d)
Instance of	ISA(i,c)
Subproperty of	SUBP(i,c)

To simplify the Datalog axioms that formally define the language semantics, some auxiliary modeling constructs were defined (shown in Table 7.2). Although these constructs also make sense and can be found in many ontology formalisms, they are not used directly by users but only internally in the meta-axioms that define the language semantics.

Table 7.2: Auxiliary modeling constructs

Modeling construct	*Datalog predicate*	*Remark*
Relation	REL(r)	Either a normal or a temporal relation
Property	PROP(p)	A relation or an attribute

Finally, the axioms that define the semantics of the ontology formalism are shown in Figures 7.1 and 7.2[1]. As can be seen, most of the elements have the usual semantics. In contrast to OWL, however, my formalism uses the so-called "closed world assumption" (CWA), i.e., domain and range declarations on properties are interpreted as constraints and not as inference rules. In my ontology language the "LOVES relation has the range PERSON" statement creates a constraint, where it is checked, that each ontology instance, which has an incoming LOVES relation, is really a person. CWA is used by all relational database implementations and also by many ontology formalisms, including most F-Logic implementations.

By contrast, OWL uses the so-called "open-world assumption" (OWA), where no constraints are used in the language, only inference rules. In OWL, the same "LOVES relation has the range PERSON" statement would create an axiom that would infer that every instance, which has an incoming LOVES relation, is an instance of the PERSON concept.

In general, it cannot be said that either OWA or CWA is better suited for knowledge representation. There are drawbacks and advantages of both formalisms. Using CWA, however, has the advantage that formalisms using this assumption usually have much more efficient reasoning procedures than formalisms using OWA. Moreover, I felt that CWA fits the human intuition better in the application context of history.

[1]In addition, axioms are needed that check that parameters of the predicates are of the right type. E.g., in the case of the ISA(i,c) predicate, it must be checked that INST(i) and CONC(c) hold. Those trivial "boilerplate" axioms are not included in the tables.

```
A normal relation is a relation:
        REL(x) :- NREL(x)
A temporal relation is a relation:
        REL(x) :- TREL(x)
Relations are properties:
        PROP(x) :- REL(x)
Attributes are properties:
        PROP(x) :- ATTR(x)
No metamodeling, an entity has only one type:
        !- CONC(x), PROP(x)
        !- CONC(x), INSTC(x)
        !- PROP(x), INSTC(x)
        !- NREL(x), TREL(x)
        !- NREL(x), ATTR(x)
        !- TREL(x), ATTR(x)
Subconcept is transitive:
        SUBC(x,z) :- SUBC(x,y), SUBC(y,z)
Subproperty transitive:
        SUBP(x,z) :- SUBP(x,y), SUBP(y,z)
Indirect instance:
        ISA(i,d) :- ISA(i,c), SUBC(c,d)
Relation value is valid for superproperties:
        NRVAL(r,is,it) :- NRVAL(p,is,it), SUBP(p,r)
Timed relation value is valid for superproperties:
        TRVAL(r,is,it,t) :- TRVAL(p,is,it,t), SUBP(p,r)
Attribute value is valid for superproperties:
        AVAL(r,i,v) :- AVAL(p,i,v), SUBP(p,r)
Transitive relation values:
        NRVAL(p,x,z) :- TRANS(p), NRVAL(p,x,y), NRVAL(p,y,z)
Transitive timed relation values:
        TRVAL(p,x,z,t3) :- TRANS(p), TRVAL(p,x,y,t1),
                TRVAL(p,y,z,t2), INTERSECTION(t1,t2,t3)
Symmetric relation values:
        RVAL(p,y,x) :- SYMM(p), RVAL(p,x,y)
Symmetric timed relation values:
        TRVAL(p,y,x,t) :- SYMM(p), TRVAL(p,x,y,t)
```

Figure 7.1: Datalog axioms defining the semantics of the ontology formalism

Property inverse is symmetric:
INV(y,x) :- INV(x,y)
Inverse relation values:
NRVAL(r,y,x) :- INV(p,r), NRVAL(p,x,y)
Inverse temporal relation values:
TRVAL(r,y,x,t) :- INV(p,r), TRVAL(p,x,y,t)
Domain constraint on relations:
!- DOMAIN(r,c), NRVAL(r,si,ti), ¬ ISA(si,c)
Domain constraint on timed relations:
!- DOMAIN(r,c), TRVAL(r,si,ti,t), ¬ ISA(si,c)
Domain constraint on attributes:
!- DOMAIN(a,c), AVAL(a,i,v), ¬ ISA(i,c)
Range constraint on relations:
!- RANGE(r,c), NRVAL(r,si,ti), ¬ ISA(ti,c)
Range constraint on timed relations:
!- RANGE(r,c), TRVAL(r,si,ti,t), ¬ ISA(ti,c)

Figure 7.2: Datalog axioms (continued...)

7.3 Fuzzy time in the ontology formalism

7.3.1 General considerations

In this section, it is discussed how the fuzzy temporal model described previously in Chapter 6 can be used in ontology modeling.

The approach for integrating the temporal model into ontological definitions follows the pattern of *modular semantics*[2] which I believe will gain importance in the near future when the complexity of domains being modeled increases. This pattern is based on the observation that a particular formalism may be good for some modeling tasks but totally inappropriate for other ones. Trying to apply the most general formalism (e.g., first-order logic) to all modeling tasks usually results in cumbersome systems with inadequate performance. Rather, a more promising approach is to combine various formalisms in a modular way, thus harvesting the best of each of them. Here I apply this principle to time modeling. However, it is easy to imagine a spatial algebra being orthogonally added to the temporal and ontology formalisms in a similar manner.

My approach may schematically be described as in Figure 7.3. On the left-hand side of the figure is the ontology model, on the right-hand side is the fuzzy temporal model with the semantics as described in Chapter 6. These two heterogeneous semantics are orthogonal and need to be integrated at the syntactic and at the semantic level.

[2]Although the term “modular semantics” was coined by Boris Motik and myself in [NM03], the idea of integrating different logical formalisms in one system is not completely new and there are different existing solutions. E.g., consider the concept of concrete domains in description logics [BH91], or the work of Pan and Horrocks to extend the W3C OWL standard with datatype groups [PH03].

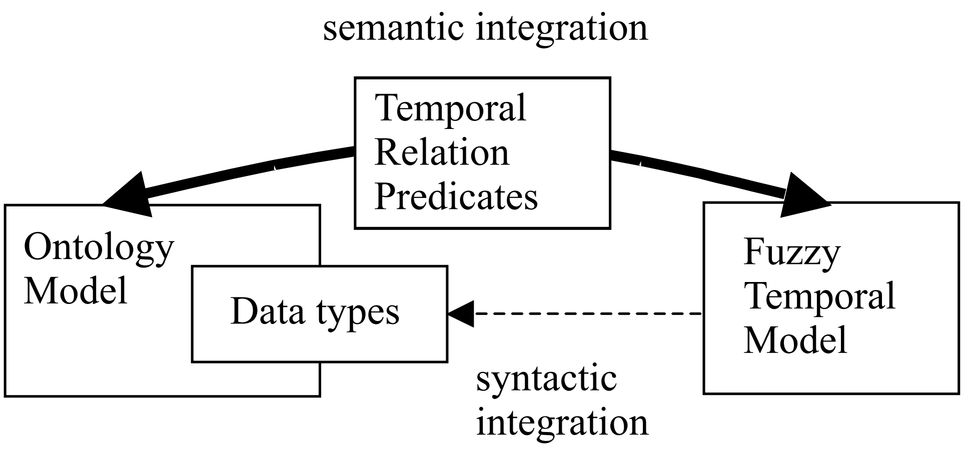


Figure 7.3: Integrating Ontology and Fuzzy Temporal Models

Integration at the syntactic level defines how to physically connect elements from one model with another. Datatypes provide an excellent mechanism for this purpose. Many ontology languages (e.g., OWL [PSHH04]) offer the capability of modeling atomic objects whose semantics is out of scope of the logical theory.

In the semantic interpretation of the ontology, instances of datatypes are interpreted as members of some concrete domain (often denoted as Δ_D). On the other hand, ontology instances are interpreted as members of the abstract domain (often denoted as Δ^I).

The concrete and abstract domains must be disjoint, thus causing the semantics of data types and of the ontology model to be separated. In my case, I introduced a separate data type which is responsible for representing fuzzy temporal information.

Integration at the semantic level defines how properties of one model semantically relate to the other model. In our case this means that we need to specify how the temporal relations from Section 6.7 are represented in the ontology formalism.

This is done by introducing a predicate into the ontology model for each temporal relation from Table 6.7. One can think of these predicates as built-ins: the arguments of the predicate are fuzzy intervals whose content is opaque to host ontology formalism. The predicates serve as a gate between the two worlds, providing an abstract interface to the interval model. One must observe that although the semantics of the predicates is not axiomatized in the ontology formalism, reasoning may still be performed on the arguments and results of the predicates. Each predicate has an additional argument receiving the fuzzy value of the relation. For example, if the fuzzy interval **I** is before interval **J** with the confidence level 0.8 then the statement `BEFORE`(**I**,**J**,0.8) is true. Note that constraints on the confidence level can be expressed by using variables[3]:

$$\texttt{BEFORE}(\mathbf{I}, \mathbf{J}, X) \;\wedge\; X > 0.8 \tag{7.1}$$

To summarize the discussion on integrating fuzzy time into ontologies, we can state that an ontology formalism should support the definition of user-defined datatypes and should also

[3]if the ontology formalism supports them

provide some means to connect the predicates of the datatype to the main ontology formalism.

7.3.2 Integration into the Datalog-based formalism

In the Datalog-based formalism, fuzzy time is represented as a user-defined datatype and special predicates connect those datatypes with the main ontology language. Luckily, KAON2 supports user-defined datatypes and user-defined predicates operating on those datatypes. Therefore, it was not a problem to implement this feature.

In addition to the predicates defining fuzzy temporal relations, I also needed to integrate a fuzzy operation into the language, namely the fuzzy intersects operation. This operation is needed for transitive temporal relations, where a new temporal interval has to be created. Here the fuzzy intersection operation seemed to be the most suitable. E.g., if we know that Alsace is part of Germany between 1940 and 1945; and we also know that Strasbourg is part of Alsace after ca. 357, we can infer that Strasbourg is part of Germany between 1940 and 1945 by intersecting the time intervals 1940–1945 and ca. 357–today.

7.4 Summary

In this chapter, first the requirements toward an ontology formalism were analyzed. I showed that time dependent relations are of crucial importance in the domain of history and that the common ontology modeling constructs (such as reflexivity or transitivity) should also be supported for those relations. I also showed that none of the existing ontology formalisms completely fulfill these requirements.

Consequently, I devised a Datalog-based ontology formalism. This formalism fulfills all of the identified requirements. Moreover, it uses the principle of "modular semantics" to efficiently integrate fuzzy temporal reasoning into the ontology language. This new ontology formalism was implemented using the freely available KAON2 reasoner.

Chapter 8

Metadata generation

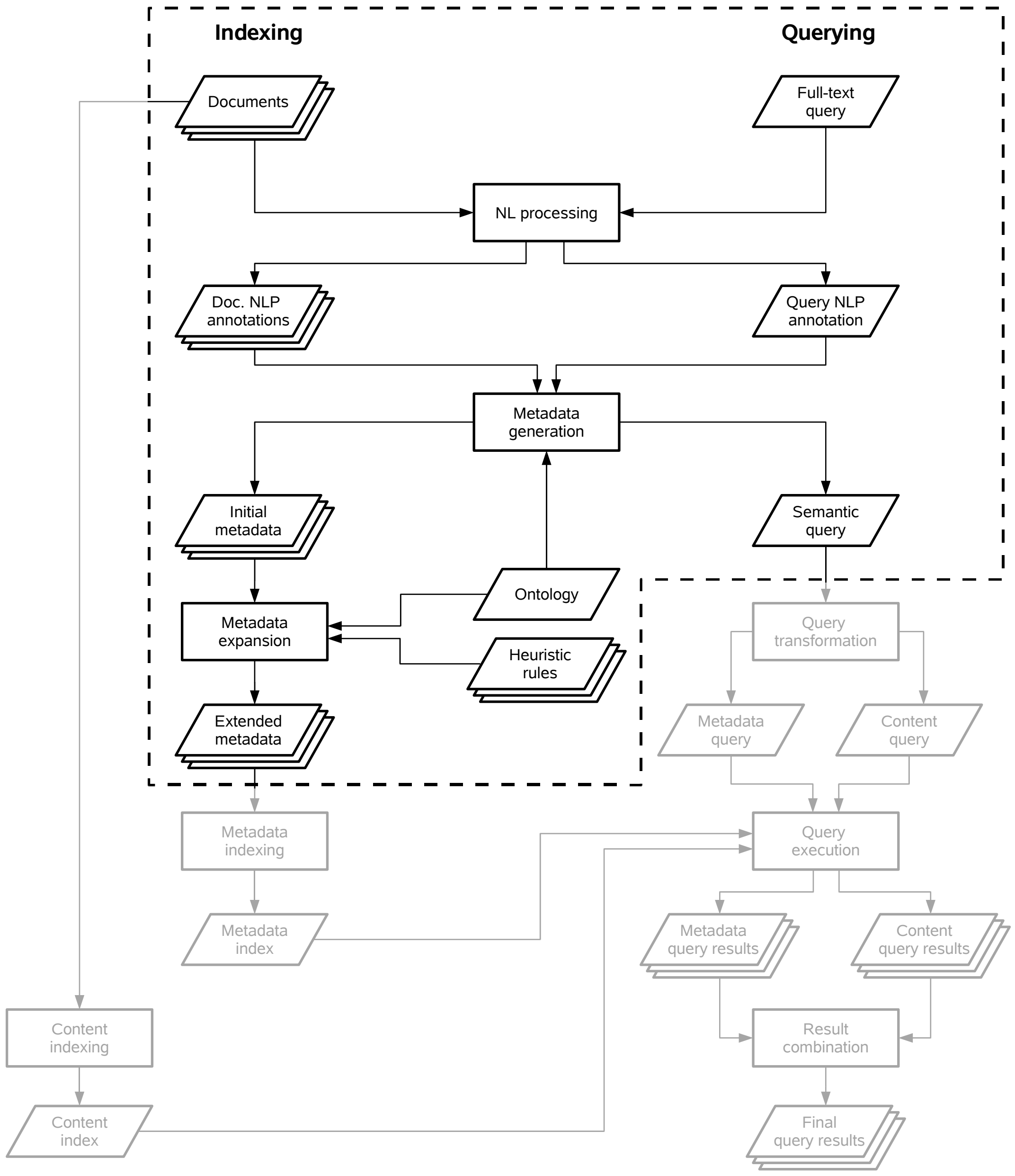


Figure 8.1: Metadata generation in the ontology-supported IR process

In this chapter, I give a detailed overview of my solution for automatically generating semantic metadata which fulfills the METADATA GENERATION requirement. The metadata generation process exploits the semantic relations and temporal information that are stored in the ontology (SEMANTIC RELATIONS and TIME DIMENSION requirements) and it tolerates ontology and domain imperfection (ONTOLOGY IMPERFECTION and DOMAIN IMPERFECTION requirements).

The solution can also be applied to parse the full-text query provided by the user into its semantic representation. Figure 8.1 shows the topics that are discussed in this chapter in the context of the whole ontology-supported IR process.

8.1 Information model

8.1.1 Requirements

As was discussed in Chapter 5, semantic metadata of the documents and the semantic query use a common information model. Based on the analysis of the requirements in Chapter 3, this information model should have the following characteristics:

- Represents information semantically, to solve language vagueness and to represent indirectly relevant entities (NL VAGUENESS requirement).
- Represents temporal information (TIME DIMENSION requirement).
- Deals with domain imperfection in the temporal dimension (DOMAIN IMPERFECTION requirement).
- Deals with ontology imperfection (ONTOLOGY IMPERFECTION requirement).
- Supports the ranking of results during information retrieval (see the discussion in Section 2.2).
- Supports scalable retrieval of results (SCALABILITY requirement).

8.1.2 Related work

Most of the related work to the information model was already discussed in Chapter 4. Based on the analysis of the related work, I identified some ideas and solutions, which can be reused in my approach. Although these ideas were already discussed in Chapter 5 before, I recapitulate here the ones that are most relevant for the information model for a better understanding.

First, systems that try to deal with some aspects of ontology imperfection combine full-text search with semantic search to diminish the negative effect of missing ontological information [VFC05, KPT+05, PKO+04]. As was discussed, I use this idea, too.

Second, to achieve scalability, most of the systems use full-text search engines and represent semantic information so that it can be indexed by full-text search engines [RSA04, KPT+05, PKO$^+$04, FMJ$^+$05, SFJ$^+$02]. Of course, this approach has some disadvantages. It is very complicated to represent structured information in the information model, if it has to be compatible with the "bag of words" approach of the full-text search engines. Although it is possible to do it — e.g., consider the swangling approach used by the OWLIR system — it is questionable whether the ranking algorithms optimized for the bag of words approach yield meaningful results on such "hacks". In spite of these potential disadvantages of the approach, it seems to me that currently there is no other alternative to achieve a scalable information retrieval system. Therefore, I will follow this way, too, and require that the model is compatible with full-text search engines.

Some features are missing from the approaches introduced so far. First, none of the models represent the time dimension explicitly. Second, although these systems exploit natural language processing (NLP) techniques to improve the quality of generated semantic metadata, they discard the NLP-results and do not represent those in the model. This is problematic because complex nominal phrases and named entities — the results of NLP — are better representations of the document content than a simple set of words.

The information model that inspired the IRCON information model the most is the information model used in the VICODI system [NDO05]. The VICODI model was motivated by the simplicity and good results of the vector space model (VSM). VICODI extended the classical VSM to include semantic information, i.e., ontology entities and temporal information.

Document metadata in the VICODI model consists of a weighted set of elements from a suitable ontology (conceptual part) and of a weighted set of temporal intervals (temporal part). For example, a possible (partial) metadata of a document that reports on some aspects of the Gulf War could be the following:

```
{ George_H_W_Bush:0.7, 1990-1991:1.0,
USA:0.8, Iraq:1.0, Gulf_War:1.0 }
```

where GEORGE_H_W_BUSH, USA, IRAQ and GULF_WAR are elements (instances) of the ontology and the numbers represent relevancy weights.

What is missing from the VICODI model is the possibility to represent fuzzy time intervals and results of NLP in order to deal with ontology and domain imperfection.

8.1.3 The IRCON information model

The IRCON model is an extension of the original VICODI model. The goal of the extensions is to create a model that represents some aspects of imperfection better than the original VICODI model.

The IRCON model consists of three parts: a textual, a conceptual and a temporal part.

The *textual part* is almost identical with the classical VSM: It is a weighted set of terms. In contrast to the classical VSM, however, the textual part contains phrases instead of simple words. In other words, I use a "bag of phrases" model instead of the usual "bag of words" approach. I.e., in my model the textual part contains phrases such as “President George H. W. Bush”. Elements of the textual part are termed *Weighted Terms* (WTerms).

My motivation to do this is to save the results of NLP in the information model. Normally, the bag of words approach is used only because it is too expensive to execute NLP on document texts. In my case, however, NLP is essential for metadata generation, therefore no efficiency penalty is incurred. On the other hand, it would be a waste of resources to discard the NLP results and only use simple words in the textual part. Especially the *named entities* (names of persons, companies or locations) that are extracted during NLP constitute a very important part of the document semantics [KPT+05].

The *conceptual part* contains a weighted set of ontology entity URIs. Entity URIs uniquely identify elements of the ontology (such as a concept or an instance). Elements of the conceptual part are termed *Weighted Ontology Instances* (WOIs)[1].

Finally, the *temporal part* consists of a weighted set of fuzzy temporal intervals. Elements of the temporal part are termed *Weighted Temporal Intervals* (WTIs).

An example (partial) semantic metadata in this information model for a document describing some aspects of the Gulf War could be the following[2]:

```
{
textual: {"Invasion of Kuwait":1.0, "Burning Oil Towers":0.5}
conceptual: {George_H_W_Bush:0.7, USA:0.8,
             Iraq:1.0, Gulf_War:1.0}
temporal: {1990-1991:1.0}
}
```

8.1.4 Discussion

The advantage of the IRCON model is that it is very similar to the classical vector-space model (VSM, see Section 2.3). Therefore, it is expected that the similarity measure that the VSM uses will also provide good results in the semantic case.

The drawback of the model is that it is much simpler than, e.g., the swangling approach mentioned above. This makes executing more advanced queries, such as “all wars led by the US”, impossible. However, based on my experience in the VICODI project, such advanced queries are not typical, most users prefer simple queries[3].

My experiences are in line with web studies that show that the average query length on the web contain 2.21 terms, with 62% of the queries containing only one or two terms and less than 4% of the queries more than 6 terms [JSS00].

[1]Although also other types of ontology entities could be stored in the conceptual part, during my work I only stored instances in the conceptual part of the metadata. That is why I use the term weighted ontology instance instead of the more general term weighted ontology entity.

[2]For the sake of simplicity, traditional temporal intervals are used in the examples.

[3]In this case “US wars” would be a typical user query.

8.2 Existing approaches for semantic metadata generation

Generating high-quality semantic metadata with the least possible human effort is generally agreed to be an important step toward the Semantic Web (see e.g. [DEG$^+$03]). Thus, there are several projects that aim to support or replace human experts in the metadata generation task.

Approaches such as OntoAnnotate [SMH01], SHOE [HH00], Annotea [KKPS02], or WebKB [ME99] propose frameworks for manual annotation of metadata. E.g., a GUI[4] application which facilitates the annotation of semantic tags is provided by [HH00]. However, fully manual approaches do not scale well for large information systems and these approaches also violate the METADATA GENERATION requirement.

There are a couple of automatic annotation systems as well, most of them are parts of complete IR solutions that were already analyzed in Chapter 4.

8.2.1 The KIM system

As was mentioned in Chapter 4, the KIM system [KPT$^+$05] is based on the GATE framework. It matches the labels of ontology entities with text snippets as one of the first steps during the natural language processing pipeline. Later, the concept information of matching candidates is exploited during a disambiguation step, where the type information that is automatically generated by the GATE framework is corrected [PKO$^+$03]. Using this disambiguation technique, it is mainly possible to avoid false categorization of text snippets. E.g., the phrase "U.S. Navy" would be recognized as a person name using the standard NLP heuristics[5] but if it is contained in the ontology as an organization, this false categorization can be avoided. I.e., the KIM system exploits the labels and the concept information of ontology instances during semantic metadata generation.

8.2.2 The system of Vallet et al.

Vallet and his colleagues [VFC05] exploit the concept taxonomy in the ontology, as well. They attach category information to each concept in the ontology and use automatic classification of the document text to determine the category of the content. If a text snippet ambiguously matches more than one ontology instance, they choose the match whose concept's category fits the document category. Using this technique, it is possible to distinguish between "Irises" as a kind of flower and as a famous picture of Van Gogh. Moreover, the system also uses the "longest match principle" to disambiguate matching ontology instances: it prefers instances with labels that have a longer match. E.g., "Real Madrid" is preferred to "Madrid".

[4]Graphical User Interface

[5]Because it matches the rule that also recognizes correct person names, such as "P. C. Lockemann", or "G. Nagypal".

8.2.3 OWLIR and SCORE

Although the OWLIR [FMJ$^+$05, SFJ$^+$02] and SCORE [SBA$^+$02] systems also support automatic metadata generation, they use commercial systems for this task, therefore it is not possible to gain deep insights about their approach. What is important to note about these systems is that they expand the initial set of metadata gained by mapping the text to ontology instances using the semantic relations (and attributes) stored in the ontology.

8.2.4 The SemTag system

The SemTag system provides a scalable solution for automatic semantic annotation [DEG$^+$03]. This system also uses mainly the textual labels that are attached to ontology entities. They also exploit, however, the concept taxonomy of the ontology by considering not only the label of the ontology entity but also all concept labels in the taxonomy path up to the ontology root. Moreover, they match not only the actual text snippet with the ontology label but they use a so-called "spot" — the text snippet and 10 words surrounding it. They use the common TF-IDF formula on the spot–taxonomy path pairs to find the most similar ontology entity to a given spot. Their claim is that considering the 10 word context of a text snippet provides enough information to successfully disambiguate most of the cases.

8.2.5 The S-CREAM system

The goal of S-CREAM [HSC02] is to generate full-featured RDF statements about the documents. E.g., it could be encoded in RDF that GEORGE W. BUSH is a PERSON and he LEADS the UNITED STATES. This model is much more general and powerful than my annotation model. It is important to note, however, that in my case, most of the information is already encoded in the ontology. Therefore, it is not necessary to state the same information again in document annotation. In this sense, S-CREAM can be also viewed as a tool for automatic ontology population.

S-CREAM follows a two-step approach for semantic metadata generation. First, it uses the Amilcare tool[6] for information extraction. Amilcare can annotate documents with user-defined XML tags using a user-defined XML vocabulary. To do that, Amilcare needs a manually annotated training corpus. Based on this corpus, it automatically generates extraction rules using various machine learning techniques. Using these rules, Amilcare can automatically annotate new, unseen documents.

At the next step, these XML annotations must be transformed into RDF statements. The main challenge is that in many cases, the information contained in the XML annotations are not sufficient to generate the needed RDF statements. The authors of S-CREAM use ontology-based heuristic rules to infer the best possible mapping between the XML annotations and the RDF statements. During the mapping, the main task is to generate the relations between the entities (such as the relation between George Bush and the United States); the mapping

[6]`http://nlp.shef.ac.uk/amilcare/`

between the XML and RDF representations of ontology instances and concepts is straightforward. Of course, it also helps if the set of XML tags is designed so that the XML-to-RDF mapping is easy, e.g., by encoding ontology relation names in such XML tag names (such as `<country_lead_by_person>` instead of `<country>`). There is a trade-off, however, between the number and complexity of XML tag names and the quality of information extraction results.

Based on this discussion, it is clear that the ontology-based rules and also the set of XML tags are highly domain-dependent. On the other hand, it is possible to provide high-quality results by carefully hand-crafting the XML tag names and the ontology-based rules and by manually annotating a training corpus.

8.2.6 The C-PANKOW system

The authors of C-PANKOW [CLS05] propose an advanced annotation and disambiguation system without any machine learning technique. Their approach is to identify correct conceptual entities by measuring statistical information from Google search results. The system uses, however, only syntactical information, i.e., it cannot find indirectly mentioned instances. Moreover, the purpose of the system is only to assign the correct semantic type information to the text snippets in the text (such as “River” or “Country” to “Niger”) but they do not connect the text snippets with an ontology. In other words, they do not generate semantic metadata in our sense.

8.2.7 Summary

To summarize, we can state that most state-of-the-art systems still concentrate on the task of matching text snippets with the most appropriate ontology entity (usually instance). For this task, they usually only exploit the lexical information of the ontology and the concept taxonomy. They ignore all other information contained in the ontology, including relations between instances and attribute values. There are some exceptions, however, including OWLIR, SCORE and S-CREAM, that use semantic relations in the ontology.

The reviewed systems ignore temporal information and do not deal with ontology and domain imperfection.

Most systems cannot identify indirectly relevant ontology entities, they can identify only those entities that are explicitly mentioned in the document text. OWLIR and SCORE are exceptions in that regard because they perform metadata expansion that has the potential to find indirectly relevant entities.

8.3 Overview of the metadata generation steps

This section provides a high-level overview of my metadata generation approach. The approach is a slightly improved version of the solution, which was already published in [YN06].

As we have seen, many systems use NLP as a first step for semantic metadata generation because it is easier to work with the information generated by NLP than with the raw document text. Therefore, I start metadata generation with an *NLP step*, which generates various NLP annotations (discussed in detail in the next section). Based on this information, it is possible to *generate an initial version of the semantic metadata*, using the information model introduced in Section 8.1. This includes the matching of document text snippets with their corresponding ontology entities. During this initial metadata generation step, *disambiguation of entities* is also performed, based on the *temporal context* of the document. With temporal context I denote the weighted set of (fuzzy) time intervals that will ultimately form the temporal part of the semantic metadata.

This initial metadata contains only ontology entities that are directly mentioned in the document text because it is based on the NLP output. Therefore, to find indirectly relevant ontology entities, a *metadata expansion step* is executed. In this step, the initial metadata is successively expanded using *heuristic rules* that exploit ontology information. The expanded metadata constitutes the final semantic metadata, which is stored in the IRCON repository.

The whole metadata generation process is shown in Figure 8.2 (the figure is a part of Figure 8.1). In the following, I describe the individual steps in more detail.

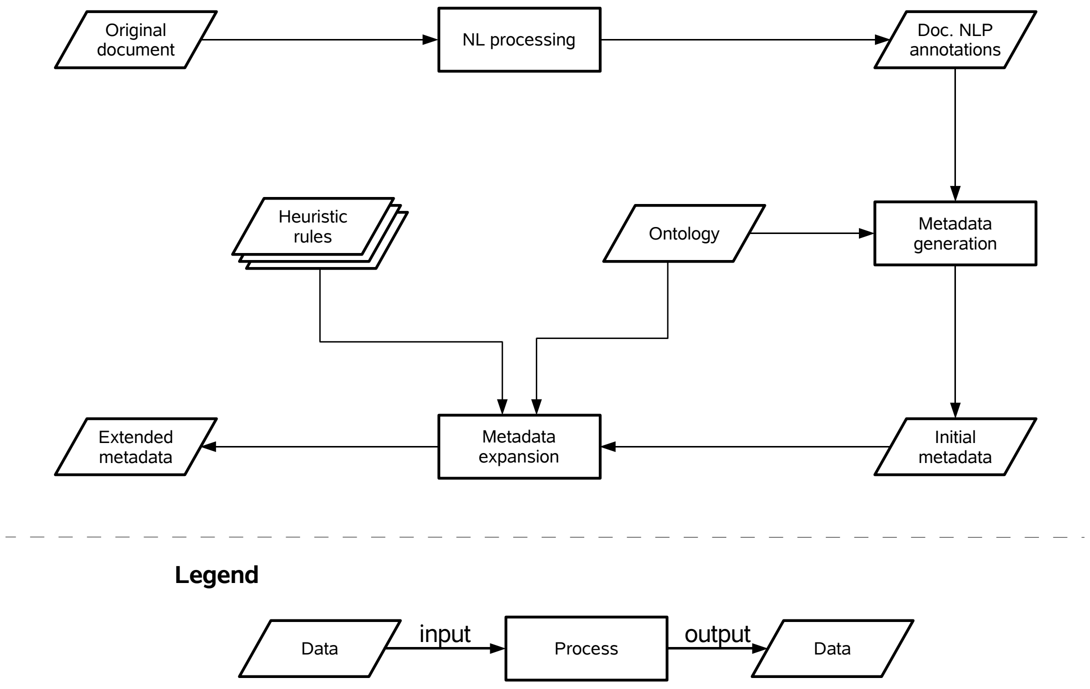


Figure 8.2: Steps of the automatic metadata generation process.

8.4 Natural language processing

All of the systems that were introduced so far, use only a subset of the NLP repertoire. This subset is sometimes referred to as *shallow parsing* or *shallow NLP* because unlike pure NLP applications it usually does not include a full (costly) linguistic analysis of the text. The costs of full-NLP are prohibitive for most applications [Kur04] and experience of the introduced state of the art systems[7] shows that shallow NLP provides acceptable results for the task of semantic metadata generation. Therefore, I also use shallow NLP in my system.

Shallow NLP methods include many techniques, such as token and sentence splitting or part-of-speech detection. The linguistic information obtained by these steps serve as input to the named entity recognition (NER) step. Named entities (NE) are entities that are usually represented as instances in an ontology. These include concrete persons, locations, organizations etc. Normally, also date specifications are considered as named entities. Named entities are important because experience shows that NE occurring in text documents constitute a major part of their semantics [KPT+05].

Detected named entities (NEs) may also have referring phrases in the text with different content. These are called *coreferences* and can be distinguished into nominal and pronominal type. An example for a nominal coreference is the term “president of the United States” referring to the NE “George W. Bush” in a document. The “he” reference pointing to “George W. Bush” is a pronominal coreference.

To the best of my knowledge, currently there is no existing IR system that exploits coreference information to create semantic metadata[8]. However, I consider this an important step, since coreference resolution[9] can improve term relevance estimation, as our tests have shown [YN06].

For shallow NLP, I use the established text engineering framework GATE[10] that includes components for various NLP tasks, including NER and coreference resolution. I use the standard ANNIE components, which are included in the standard GATE installation.

The result of the described NLP operations are GATE annotations following a special annotation scheme. They contain detailed linguistic information about each identified term, like its position within the sentence, part-of-speech information, and a list of its coreferences. The annotations also contain the token type information generated by the NER step (such as person, organization etc.). These GATE annotations are automatically stored for each document in a relational database for later use during the initial metadata generation step.

This separation of costly NLP operations from subsequent ontology-dependent tasks has some significant advantages. First, linguistic annotations can be generated independently from ontology lookup operations and thus are independent from any changes in the ontology[11]. Second, different ontology-based heuristics can be applied and tested without complete regeneration of GATE annotations.

[7]e.g., KIM [KPT+05] or the system of Vallet et al. [VFC05]

[8]Of course, coreference information is extensively used for other purposes, such as indexing full-text documents [KL05] or generating document summaries [BWK+03].

[9]modern implementations may achieve an F-measure of up to 70 percent

[10]General Architecture for Text Engineering, `http://gate.ac.uk/`

[11]For better coreference recognition among named entities, it is sometimes necessary to update the gazetteer lists of GATE based on the ontology labels. In most cases, however, GATE does a good job in properly identifying named entities using various syntactic heuristics.

8.5 Initial semantic metadata generation

8.5.1 Identifying ontology instances

To successfully create semantic metadata, the system must be able to identify the appropriate ontology instances for the NEs identified during the NLP step[12]. It is a difficult task because a term in the document can syntactically match many ontology instances. To reduce this ambiguity to a minimum, my implementation follows the already mentioned "longest match principle", which is also used by [VFC05]. According to this principle, always the longest possible text snippet is matched with the ontology instance labels. I.e., we prefer "U.S. Marines" to "U.S.". It is still possible, however, that more than one ontology instance (OI) matches the longest text snippet. In this case, all OI candidates are stored for the text snippet.

Next, linguistic annotations covering an ontology instance are transformed to *ontology instance annotations* (OIAnnotation). Every OIAnnotation consists of URIs of possibly matching ontology instances (OI) and the number of occurrences of its candidate OIs. The resolved coreferences are taken into account simply by increasing the occurrence counter of the OIAnnotation. E.g., if a pronoun is detected as a coreference to a certain entity and that entity is known to be an OI, the occurrence counter of the OIAnnotation is increased by one.

8.5.2 Handling non-ontological named entities

The remaining GATE NE annotations are categorized as *term annotations* (TermAnnotation) and *date annotations* (DateAnnotation). Term annotations contain the (normalized) terms from the text, together with their occurrence counters; whereas date annotations are special term annotations, where the term text represents a valid date (or time) specification, such as "May 12, 2006" or "today"[13].

8.5.3 Calculating metadata weights

After this step, all annotations are transformed to the initial document metadata, using the model introduced in Section 8.1. The mapping is straightforward. WTIs are generated from DateAnnotations, WTerms from TermAnnotations and WOIs from OIAnnotations. The main question is how to calculate the weight of the metadata elements. The main principle here is that the weight of the metadata element should increase when the corresponding annotation appears more frequently in the document. I.e., I follow the basic principle that is used by classical full-text indexing.

[12]Actually we consider all tokens (text snippets) in the text during this step, not only the text snippets that were identified by GATE as NEs. This is needed because GATE sometimes fails to correctly identify text parts as NEs.

[13]The current implementation can only handle absolute date specifications, such as "May 12, 2006". Relative date specifications, such as "today", are ignored.

Based on this principle, weights of metadata elements are calculated according to the following logarithmic function:

$$w(x) = \left(\frac{log(x+1)}{log(r_{max}+1)} \right)^2 \tag{8.1}$$

where $w(x)$ denotes the resulting weight of a new metadata element; x denotes the occurrence counter of the corresponding annotation element and r_{max} the largest occurrence counter of all annotation elements for the document. This exact function is the empirical result of many experiments that were executed during the work described in [Yol06]. These experiments showed that a simple linear function of the occurrence frequencies does not yield good results because it usually generates almost-zero weights for entities that are not the most important in the text but still relevant. This logarithmic function "smoothes" the results and gives a little bit more weight also for the not-so-important entities.

Table 8.1 illustrates by example the transformation from NLP annotations to an initial document representation (with $r_{max} = 36$) in a document about the United Airlines Flight 93[14].

Table 8.1: Calculating metadata element weights

Entity	*Occurrence*	*Metadata weight*
`United-Airlines-Flight-93`	36	1.0
`Ziad-Jarrah`	7	0.33
`World-Trade-Center`	3	0.15
`George-W.-Bush`	2	0.093

8.5.4 Weights for time intervals

The introduced weight calculating function is directly applied on WOIs and WTerms, where the maximum occurrence parameter is calculated by considering all OIAnnotations and TermAnnotations. For WTIs, however, it would be problematic to directly apply this function because the coreference semantic usually does not apply for time intervals. If we would only consider the exactly same time specifications as coreferences, in a typical document all time intervals would get the same weight. This is because in a typical document an exact time specification appears only once. This would also mean that all time intervals would get an extremely low weight if we applied the maximum occurrence value calculated by considering ontology and term annotations.

My solution for time interval weight calculation is based on the insight that the classical coreference semantic should be relaxed for time intervals[15]. More precisely, I consider overlapping

[14]All examples in this chapter are based on a Wikipedia document describing the United Airlines Flight 93. This flight was hijacked during the September 11, 2001 attacks and crashed in Pennsylvania after the passengers tried to gain back the control over the machine. Ziad Jarrah was the terrorist pilot flying the machine. This document was processed during the evaluation of the IRCON system (described in Chapter 12), i.e., the presented examples are results of the implemented system.

[15]I also consider simple dates as time intervals with the length of one day.

time intervals as co-occuring. If two time intervals overlap, I create a new time interval, which is the combination (union) of the two original time intervals. The occurrence value of the new interval is the sum of the occurrence values of the original intervals. I iteratively check the pairwise overlap among the time intervals until there are no overlapping time intervals in the set of WTIs.

The exact temporal relation to check overlapping intervals depends on how conservative the strategy one takes. The *intersects* and the *during* relations (see Table 6.1) seem to be the most appropriate to model the semantics of the natural language "overlaps" relation, where the during relation means a more conservative strategy. For my experiments I used the intersects relation which seemed to give more intuitive results. After this procedure, I apply the introduced weight calculation algorithm for the remaining WTIs to calculate the final weights, where I consider only the WTIs for calculating the r_{max} parameter of the function.

As an example, Table 8.2 shows a partial list of identified time intervals in the United Airlines Flight 93 document. It can be seen that most of the intervals have the occurrence value 1. The only overlapping intervals in this set are the intervals **(2002-09-10 – 2002-09-11)** and **(2002-09-11 – 2002-09-12)**. Merging these two intervals results in a new interval **(2002-09-10 – 2002-09-12)** with an occurence value of $1 + 1 = 2$ (see Table 8.3). Finally, Table 8.4 shows the final list of intervals with the calculated occurrence weights.

Table 8.2: Initial temporal part

Time interval	*Occurrence*
2001-09-11 – 2001-09-12	3
2001-09-19 – 2001-09-20	1
2001-10-01 – 2001-10-31	1
2002-09-10 – 2002-09-11	1
2002-09-11 – 2002-09-12	1

Table 8.3: Merged temporal part

Time interval	*Occurrence*
2001-09-11 – 2001-09-12	3
2002-09-10 – 2002-09-12	2
2001-09-19 – 2001-09-20	1
2001-10-01 – 2001-10-31	1

Table 8.4: Final temporal part

Time interval	*Weight*
2001-09-11 – 2001-09-12	1.0
2002-09-10 – 2002-09-12	0.63
2001-09-19 – 2001-09-20	0.25
2001-10-01 – 2001-10-31	0.25

8.5.5 Handling fuzzy time specifications

Another issue that had to be considered during the parsing of time intervals was the case of fuzzy time specifications. As was already discussed in Chapter 3, many documents in the history and/or news domains contain vague, natural language specifications. Examples for such specifications include “early October 2006”, “the mid-1970s”, or “the late 80s”. The fuzzy time model that was introduced in Chapter 6 can represent such time specifications without any problems. I parsed those fuzzy specifications by specifying some pre-defined operators for the fuzzy hedges “early”, “mid” and “late” that created a new fuzzy set based on the original crisp set representing a traditional temporal interval. E.g., Figure 8.3 shows the fuzzy interval that was automatically created for the time specification “early October 2006”.

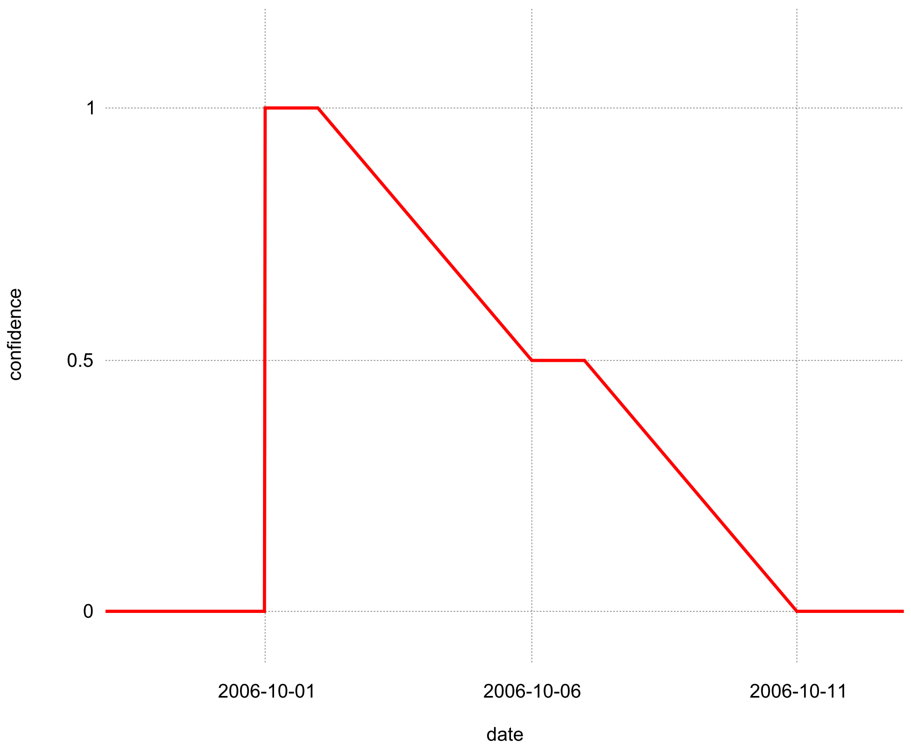


Figure 8.3: The fuzzy interval representing “early October 2006”

To implement the overlap checking on fuzzy temporal intervals, the fuzzyfied versions of the *intersects* or the *during* relations can be used (see Table 6.7). As was discussed above, I used the intersects relation. For creating the merged intervals, I used the *fuzzy union* operation, which is the standard counterpart of the crisp union operation on traditional intervals. This way, the whole transformation and weight calculation process can also be applied to fuzzy intervals.

8.5.6 Disambiguation of ontology instances

The last issue that remains for the initial metadata generation step is the disambiguation of the OntologyAnnotations. Clearly, it is possible that even using the longest match principle, more than one ontology instance matches a specific text snippet. E.g., the phrases "George Bush" or "President Bush" would find both GEORGE_H_W_BUSH and GEORGE_W_BUSH. As was discussed in Section 8.2, there are several possible approaches to deal with this problem. Most state of the art approaches apply some kind of syntactical heuristic that tries to decide based on some syntactic characteristics of the text snippet, of the ontology instance label, and/or of the surrounding document, which one is the most probable OI candidate. To justify these approaches, examples are usually presented that include OIs of different concepts. However, in the history and news domain, most ambiguity appears among the same types of OIs, among person names, organization names etc. The case of the two President Bushes is such an example. In this case, alternative disambiguation approaches are needed.

My solution for the problem is to exploit the temporal information in the ontology to solve the disambiguation problem. In many cases it helps to decide which is the proper OI, if we consider the temporal context of a document. E.g., in a document that has the time context of the year 2001, it is much more likely that "President Bush" matches GEORGE_W_BUSH than in a document with the time context of the year 1991.

For disambiguation, exactly this idea was implemented in my system. If an OntologyAnnotation has more than one OI candidate, all candidates are checked whether they fit into the temporal context of the document[16]. Currently this check is implemented by comparing the existence time of the instance with all time intervals in the temporal part of the metadata using the fuzzy intersects relation and taking the maximum value of these checks as our confidence that the instance is in the time context of the document[17]. The candidate with the biggest confidence value is then selected as the matching OI. If more than one candidate has the same maximal confidence value, all such candidates are added to the semantic metadata and the weight is distributed evenly among them.

8.6 Metadata expansion

8.6.1 General approach

The main idea that distinguishes my approach from most other state of the art solutions[18] is the metadata expansion step. During this step, the semantic relations stored in the ontology are exploited to find ontology elements that are indirectly relevant for the document but are not mentioned explicitly. This goes beyond a syntactic mapping between text snippets and labels of OIs, as in other approaches. As a result, the system is capable of automatically drawing some basic conclusions about the relevance of abstract concepts, similar to human readers' cognitive processing.

In general, the cognitive process to infer the relevance of abstract, indirect concepts can be very complex. Indeed, based on my analysis, some features are needed that are not supported even by the most expressive (but decidable) well-known ontology formalisms, such as OWL

[16]The temporal part of the metadata is generated before the conceptual part, so this is possible.

[17]Of course, using other fuzzy relations would be also possible. E.g., the fuzzy during relation would define a more conservative strategy.

[18]with the exception of OWLIR and SCORE

or F-Logic. E.g., counting the ontology entities that fulfill some specific constraints was not possible in either of these formalisms. Therefore, some parts of the reasoning algorithm had to be implemented in a Turing-complete programming language (in my case in Java) and only parts of the algorithms could be delegated to the ontology reasoner.

Generally, however, this would mean that to implement the inference rules one would always write a special purpose program. This would make the process too complex and thus infeasible. Therefore, we identified some common patterns of reasoning by analyzing documents in the history and news domains. These patterns described the great majority of inferences about the relevance of abstract concepts. In my system, it is only possible to define inference rules that follow these identified patterns. On the one hand, this naturally limits the expressiveness of the inference rules. On the other hand, it constraints the search space for meaningful inference rules, and thus makes the rule specification process easier. I term the rules fitting the simplified inference patterns *heuristic rules*, to stress the fact that they can only incompletely model the complex reasoning process about the relevance of abstract concepts that happens in the human brain.

When the inference rules are known, the metadata expansion step is conceptionally simple: an algorithm iteratively applies these heuristic rules and terminates when no further adaptations can be made to the document metadata. The resulting metadata is the final result of the whole metadata generation process.

8.6.2 Evaluation ontology

For evaluation purposes, an ontology and some heuristic rules were developed. Although the evaluation and its results are discussed only later in Chapter 12, I describe the ontology already here and some of the heuristic rules in the next subsection.

The evaluation ontology was strongly inspired by the ontology that was designed for the VICODI project[19] [Nag04] and is a simplified version of the latter. However, the new ontology also contains some new elements that did not exist in the original VICODI ontology, such as the time-dependent relations that were not supported by the ontology formalism used in VICODI[20].

The high-level structure of the ontology is shown in Figure 8.4. The concept hierarchy starts with the THING concept. This is common to almost all ontologies[21]. The three subconcepts of THING are EVENT, LOCATION and PARTICIPANT, representing events, locations and entities that can participate (ISINVOLVEDIN relation) in an event that happens in (possibly many) location(s) (HAPPENSAT relation). This major structure shows how important time (events) and space (locations) are in this application domain. Locations can be parts of other locations (ISPARTOF relation) and events can be subevents of other, bigger events (ISSUBEVENTOF relation).

Among participants I distinguish between agentive entities (the AGENT concept) and non-agentive entities (currently only the OBJECT concept). Agentive entities can initiate events

[19]Available for download at `http://www.vicodi.org`

[20]VICODI used the ontology formalism of the KAON system that is a slight extension of RDFS.

[21]In many cases this main concept is called “Root” or “Entity”.

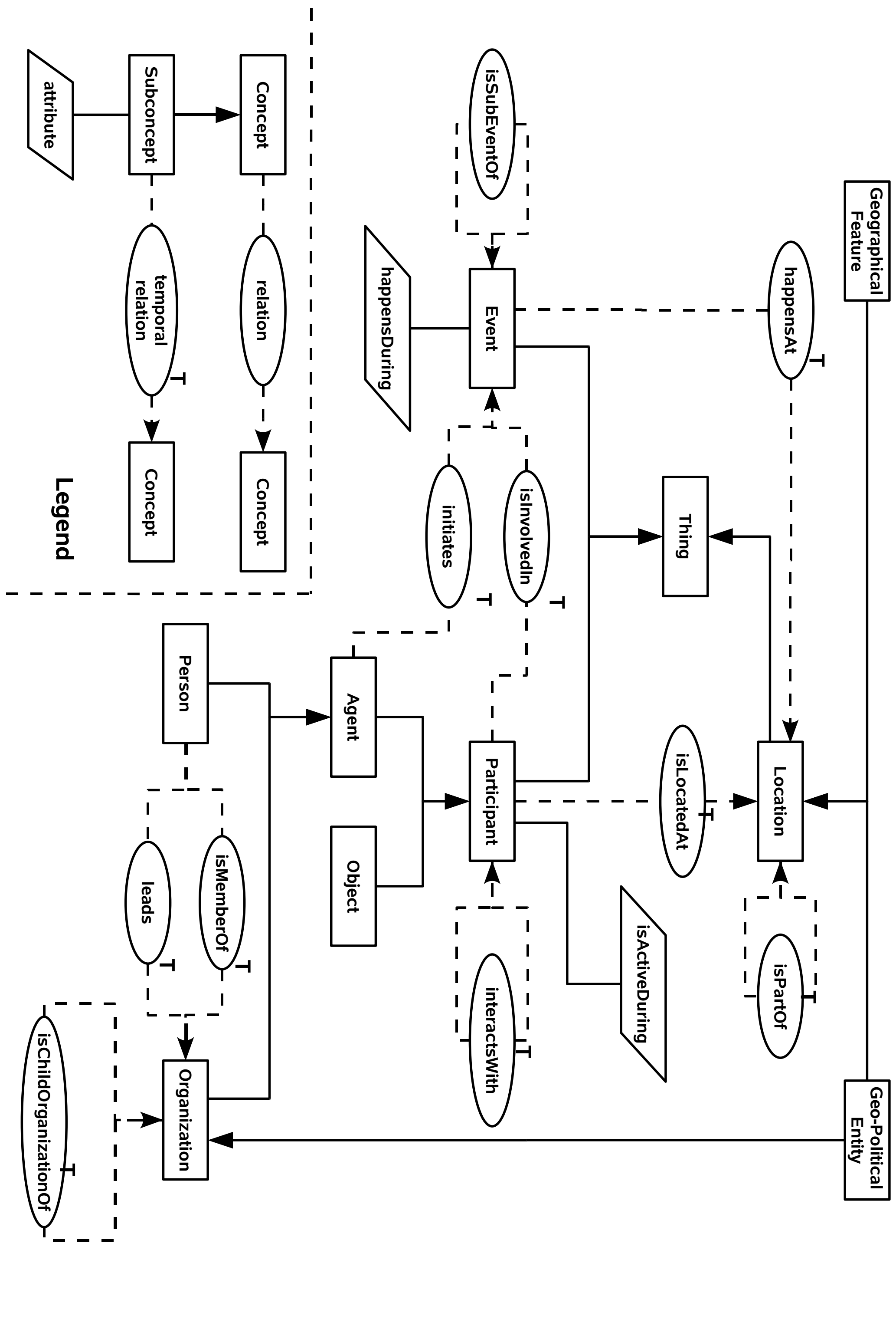


Figure 8.4: Structure of the evaluation ontology

(INITIATES relation), while objects can only passively participate in events. In this simple ontology there are currently two types of agents: persons (PERSON concept) and organizations (ORGANIZATION concept).

It is interesting to note that one kind of the locations, namely geo-political entities (GEO-POLITICAL ENTITY concept) are also modeled as organizations. This is needed because geo-political entities, such as countries or cities, are typically viewed as locations (e.g., George Bush lives in the USA) but on the other hand, they are also agentive entities that can initiate events (e.g., USA attacks Iraq).

There are some custom relations among persons and organizations and among organizations, such as LEADS or ISCHILDORGANIZATIONOF. There is also a generic relation among participants (INTERACTSWITH). Finally, all entities except for geographical features (GEOGRAPHICAL FEATURE concept) have an attribute that constraints the time period when they can be considered as relevant for a document (HAPPENSDURING attribute for events and ISACTIVEDURING attribute for participants).

It is also important to note that there are many time-dependent relations in the ontology, i.e., relations that are valid only during a specific period of time. Such relations include, e.g., the LEADS or the ISPARTOF relations.

The ontology itself contains 217 instances, 47 relation instances, 188 timed relation instances, and 11 temporal attributes. One of the temporal attributes (the HAPPENSDURING time of the 2003 INVASION OF IRAQ instance) is defined using the fuzzy time model introduced in Chapter 6. The ontology was developed approx. in 5 workdays. I.e., it is a typical example of a casual, imperfect ontology.

8.6.3 Heuristic rules

The heuristic rules follow the pattern shown in Figure 8.5. To put it simple, the pattern allows domain experts to specify rules that add new OIs to the conceptual part of the metadata

- if they exist in the temporal context of the document and
- if they are connected with other instances in the metadata through relations that are valid in the temporal context of the document.

For checking whether an OI or a relation is in the temporal context, I use the same algorithm that was described previously for the disambiguation of OIs.

For the further discussion, I will call the instances, whose existence in the metadata is checked, the *antecedent OIs* and the instance that is selected by the rule for addition the *consequent OI*.

Concrete examples of rules following this pattern are shown in Figure 8.6 and Figure 8.7. The first rule adds new events to the metadata based on participants of these events, while the second rule adds new, abstract events based on their subevents. Parallel with the evaluation ontology, 13 such heuristic rules were defined.

Table 8.5 shows an example for the application of the “3 participants rule”, while Table 8.6 shows an example application of the “subevent rule”.

Add x to M if all of the following conditions hold

- $C(x)$
- $a^T(x)$
- $\exists x_{1,1} \cdots x_{1,n_1} : n_1 \geq N_1 \wedge x_{1,1} \in M \wedge \cdots \wedge x_{1,n_1} \in M \wedge C_1(x_{1,1}) \wedge \cdots \wedge C_1(x_{1,n_1}) \wedge r_1^T(x_{1,1}, x) \wedge \cdots \wedge r_1^T(x_{1,n_1}, x)$

  $\vdots$
- $\exists x_{k,1} \cdots x_{k,n_k} : n_k \geq N_k \wedge x_{k,1} \in M \wedge \cdots \wedge x_{k,n_k} \in M \wedge C_k(x_{k,1}) \wedge \cdots \wedge C_k(x_{k,n_k}) \wedge r_k^T(x_{k,1}, x) \wedge \cdots \wedge r_k^T(x_{k,n_k}, x)$

where M denotes the current document metadata; $C(x)$ denotes that x is instance of the concept C; $r^T(x, y)$ denotes that the instances x and y are connected with a relation r, whose validity time is in the temporal context of the document; and $a^T(x)$ denotes that the value of the attribute a on instance x is in the temporal context of the document.
The input parameters of the pattern are M, the $C_1 \cdots C_k$ concepts, the $N_1 \cdots N_k$ minimal cardinalities, the C concept, the $r_1 \cdots r_k$ relations, and the a attribute. The specification of a is optional and $k \geq 1$ must hold.

Figure 8.5: Rule pattern

Add x to M if all of the following conditions hold

- $Event(x)$
- $happensDuring^T(x)$
- $\exists x_1 \cdots x_n : n \geq 3 \wedge x_1 \in M \wedge \cdots \wedge x_n \in M \wedge Participant(x_1) \wedge \cdots \wedge Participant(x_n) \wedge isInvolvedIn(x_1, x) \wedge \cdots \wedge isInvolvedIn(x_n, x)$

Idea: If at least three instances of the Participant concept – e.g., instances of the concepts PERSON or ORGANIZATION – are contained in the current semantic metadata, these participants are related via the ISINVOLVEDIN relation to the same EVENT instance and the temporal context of the document is compatible with the time interval given by the temporal attribute HAPPENSDURING of this event, then this event is considered relevant for the document.

Figure 8.6: 3 participants rule

Table 8.5: Example application of the three participants rule

Participants		*Event*

`United-Airlines-Flight-93`
`Ziad-Jarrah`
`World-Trade-Center`
	$\Rightarrow$	`Sept-11-2001-Terrorist-Attack`

Add x to M if all of the following conditions hold

- $Event(x)$
- $happensDuring^T(x)$
- $\exists x_1 \cdots x_n : n \geq 1 \wedge x_1 \in M \wedge \cdots \wedge x_n \in M \wedge Event(x_1) \wedge \cdots \wedge Event(x_n) \wedge isSubEventOf(x_1, x) \wedge \cdots \wedge isSubEventOf(x_n, x)$

Idea: At least one ontology instance of the concept EVENT in the metadata leads to the addition of all related EVENT instances (via relation ISSUBEVENTOF).

Figure 8.7: Subevent rule

Table 8.6: Example application of the subevent rule

Subevent		*Abstract event*
`Sept-11-2001-Terrorist-Attack`	$\Rightarrow$	`War-on-Terrorism`

In addition to the rule scheme, it is also possible to define thresholds on the minimum weights of the considered antecedent OIs. E.g., if a threshold 0.3 is specified for the subevent in the subevent rule, an event that has the weight 0.2 in the metadata will not be considered as a valid antecedent OI for the rule. Moreover, it is also possible to specify a threshold on the time context check. This makes sense because the metadata contains fuzzy time intervals. Therefore, the temporal context check will not necessary yield a result of 0.0 or 1.0 but an OI or a relation can also be partially in the temporal context. E.g., if a threshold 0.3 is specified on the HAPPENSDURING attribute in the subevent rule, consequent event OIs that are not in the time context with at least 0.3 confidence will not be proposed by this rule.

8.6.4 Weight calculation

The heuristic rules identify new, relevant OIs but to create the WOIs, also the OI weights have to be defined.

Our weight calculation scheme for the WOIs introduced by the heuristic rules is the following:

$$\overline{w} = \left(\sum_{i=1}^{p} \frac{w_i}{p} \right) \cdot \prod_{j=p+1}^{n} \left(1 + \frac{w_j}{2+p} \right) \tag{8.2}$$

where $\overline{w}$ denotes the resulting weight of the new WOI; $w_1, w_2, ..., w_p, w_{p+1}, ..., w_n$ are the weights of all n WOIs in decreasing order, which are used as input values of the rule with a minimum cardinality of $p = \sum N_i$. This means, at least p WOIs of the current metadata have to meet the rule requirements. The more additional elements are contained in the metadata, the higher the resulting weight of the WOI gets (until a maximum of 1.0). The exact formula is the result of empirical experiments [Yol06].

In addition to this weight calculation formula, it is also possible to specify an *amplification factor* for each rule. The result of the formula multiplied by the amplification factor yields the final weight of the WOI.

With the amplification factor, it is possible to express our general confidence in a given rule. E.g., we can be more confident in the 3 participants rule than in the subevent rule. It can be expressed by setting the amplification factor for the subevent rule lower than the amplification factor of the 3 participants rule.

Table 8.7 shows the weight calculation process for the 3 participants rule, with an amplification factor of 0.98. Because there are exactly three participant OIs, the second part of the formula is not executed. To show the effect of additional antecedent OIs, Table 8.8 shows the weight calculation result for a modified version of the same rule where only the existence of two participants is required. It can be seen that additional antecedent OIs significantly increase the weight of the consequent OI.

Table 8.7: Weight calculation example for the three participants rule (p=3)

OI	*Weight*
`United-Airlines-Flight-93`	1.0
`Ziad-Jarrah`	0.33
`World-Trade-Center`	0.15
$\Downarrow$	
`Sept-11-2001-Terrorist-Attack`	0.48

Table 8.8: Weight calculation example for the two participants rule (p=2)

OI	*Weight*
`United-Airlines-Flight-93`	1.0
`Ziad-Jarrah`	0.33
`World-Trade-Center`	0.15
$\Downarrow$	
`Sept-11-2001-Terrorist-Attack`	0.68

8.6.5 Expanding the metadata

After the proposed WOI of a rule is calculated, the metadata should be changed. If the metadata does not yet contain the WOI, it will be simply added to the metadata. If another WOI with the same OI part already exists in the metadata, only the weight of that WOI is adjusted. In the current implementation, the new weight of the metadata WOI will simply be the maximum of its existing weight and the weight of the WOI proposed by the rule. We also experimented with other possibilities, such as calculating the average of the weights but empirically the simple maximum strategy provided the most intuitive results.

8.6.6 The metadata expansion algorithm

Based on the previous discussion, the whole algorithm for the metadata expansion is as follows:

1. Set the current document metadata as the initial metadata.
2. Apply the following steps iteratively
   a) Execute all applicable rules on the current metadata.
   b) Extend the current metadata with WOIs added by the rules (or adjust the weights of existing WOIs, if they are already there). If WOIs with the same OI are proposed by different rules, use only the WOI with the highest weight.
   c) If the metadata has been modified, restart the iteration.
3. Return the current metadata as the final metadata.

The resulting metadata is the final semantic metadata of the IR system and it is stored in the central repository for later indexing.

8.7 Manual correction of metadata

As we have seen, the metadata generation process is fully automatic. It cannot be expected, however, that the quality of the generated metadata reaches the quality of manually created metadata. Therefore, the possibility is provided for domain experts in the IRCON prototype to review the generated metadata, validate, and (if needed) correct it. I.e., the system also provides for a semi-automatic annotation process.

It is important to see that the metadata generation process can be started many times. Regenerating semantic metadata is rather the rule than the exception and this is the most convenient way to react to ontology changes. Clearly, it would be catastrophic if during automatic regeneration of metadata, already validated or manually added information would be changed or even lost, or manually deleted information would be added again. To avoid this situation, the information whether a metadata element has been validated or not is stored directly in the information model using a “validated” flag[22]. Elements that bear this flag are not changed during the regeneration process.

The possibility to validate parts of the metadata has many interesting uses. E.g., identifying relevant time information in the document text is a very challenging task, which has (at least nowadays) a relatively high error rate. Erroneous temporal information in the metadata can also decrease the quality of the conceptual part of the metadata. If after an initial run, the domain expert manually corrects the temporal part it can greatly increase the quality of the conceptual part. Even though the expert has to review the temporal part, she or he is still relieved from the laborious work of mapping text snippets to ontology instances.

[22]Manually added information automatically gets the validated flag. Manually deleted information is not deleted completely, just kept in the model with weight zero.

8.8 Query parsing

If one considers the ontology-based IR process introduced in Section 5.4, it is easy to see that the tasks of metadata generation and query parsing are very similar. In both cases, the input is free text and the output is a construct in the information model. Therefore, the techniques introduced for automatic metadata generation can also be used for query parsing.

There are some minor differences, however. While during metadata generation, we can use the assumption that the major part of the document semantics is covered by NEs and consequently only include NEs into the textual part of the metadata, in the case of query parsing each and every query token counts. E.g., if the user submits a query "USA attacks Iraq", the "attacks" string should not be deleted from the query, even though it is not a named entity. Further, in the case of a query, it is not possible to effectively estimate the weight of information model elements because usually queries are simply too short: each token usually appears only once. This normally results in a situation, where each element of the query has a weight 1.0[23].

Therefore, in our example, the query would look like the following[24]:

```
{
textual: {"attacks":1.0}
conceptual: { USA:1.0, Iraq:1.0 }
temporal: {}
}
```

Based on this discussion, only one small change is needed to the existing metadata generation framework: When invoked in query parsing mode, the framework should keep all non-recognized tokens of the query and add them to the textual part of the information model. In the IRCON prototype, this change was implemented and the same framework is used for metadata generation and query parsing.

8.9 Comparison with related works

As was shown in Section 8.2, for the disambiguation of entities at the text-to-ontology matching step, most systems use some kind of heuristics that exploits the context of the information. "Context" means here, however, only the surrounding words in the document, i.e., no semantic information is exploited. While this approach seems to work for disambiguating entities of different types (such as Niger as country and Niger as river), this makes the disambiguation of cases where the entities are of the same type (e.g., deciding between George H. W. Bush and George W. Bush) practically impossible. In my work, I experimented using the time context of the document for disambiguation instead. This approach has the potential to disambiguate entities in the mentioned problematic case. This timed-based approach is orthogonal to the

[23]Of course, if a token is ambiguous, i.e., it matches more than one ontology instance, the weights can be lower than 1.0.

[24]Actually in a perfect ontology, the "USA attacks Iraq" event(s) would have been already added to the ontology and recognized by the query parser.

syntactic heuristics used by the other systems and the two approaches can be integrated in one system to further improve the results.

The system that follows a very similar approach to my solution is S-CREAM. They also build on the results of a legacy information extraction system and refine the results with ontology-based heuristic rules. The approach of S-CREAM is highly domain-dependent, similarly to my approach. This has some advantages (better results in the domain) but also some drawbacks (more human effort needed). S-CREAM has a different focus, however. Instead of identifying indirectly relevant entities, they concentrate on finding the relations between ontology entities in the text. This task is complementary to my approach and can be considered as part of the ontology building or ontology population, respectively. I.e., the two tasks can be combined.

The OWLIR system also expands semantic metadata using the information stored in the ontology. However, they do not exploit the temporal context of the document to guide the inference. This has the danger of inferring wrong facts because some of the relations do not always hold among ontology entities and therefore may not be relevant for a specific document.

The SCORE system also expands the initial semantic metadata using ontology relations. They do not exploit, however, the temporal context of the document, similarly to the OWLIR system. An interesting idea of the SCORE system is to use ontology relations during the disambiguation step. Clearly, if a candidate ontology entity is better connected with other entities in the current metadata than the other candidate, the better connected candidate is more likely to be the proper one. I use the time context of the document during the disambiguation step but the idea of SCORE could also be integrated into my system, the two approaches are orthogonal.

Although the system of Rocha et al. [RSA04] (discussed in Section 4.1.4) is a system for query expansion, we have seen in this chapter that metadata generation has many commonalities with query expansion. Therefore, the approach presented in that work could be used at the metadata expansion step in my system. It would be an interesting future work to check whether the results provided by the naive approach of Rocha et al., where the propagation rules are domain-independent, are better or worse than the results provided by my domain-dependent approach. It is a possibility to make the system more robust and cheaper to set up by using some default, domain-independent heuristics like in the system of Rocha et al. and apply hand-crafted, domain-specific rules to improve the results only if needed.

8.10 Summary

In this chapter, I introduced my solution for automatic metadata generation. After a review of the related approaches, I discussed the steps of my metadata generation process in detail. This included the initial shallow NLP-based information extraction step, the metadata generation step (including the disambiguation of the matched ontology entities), and finally the expansion of metadata using ontology-based heuristic rules. The major contributions of my approach are the following:

- It can find indirectly relevant concepts not explicitly mentioned in the document text. To do that, it applies ontology-based heuristic rules during the metadata expansion step that exploit semantic relations and the time information stored in the ontology. Only few other state of the art systems perform metadata expansion, most other systems only match text snippets to ontology entities, i.e., they can only find concepts that are explicitly mentioned in the document text.

- In my approach, the temporal context of the document is also considered. This can effectively guide both the disambiguation of ontology entities and the metadata expansion step. My approach is unique in this regard.

Later, I discussed the possibility of defining a semi-automatic annotation process to further increase the quality of metadata. Next, I showed how the introduced framework can be used for query parsing as well. Although viewing query parsing as a special kind of metadata generation seems to be quite intuitive, this idea is not used by any other system.

I closed this chapter by comparing my approach with other state of the art solutions and identifying ideas for possible further improvements.

Chapter 9

Indexing and querying

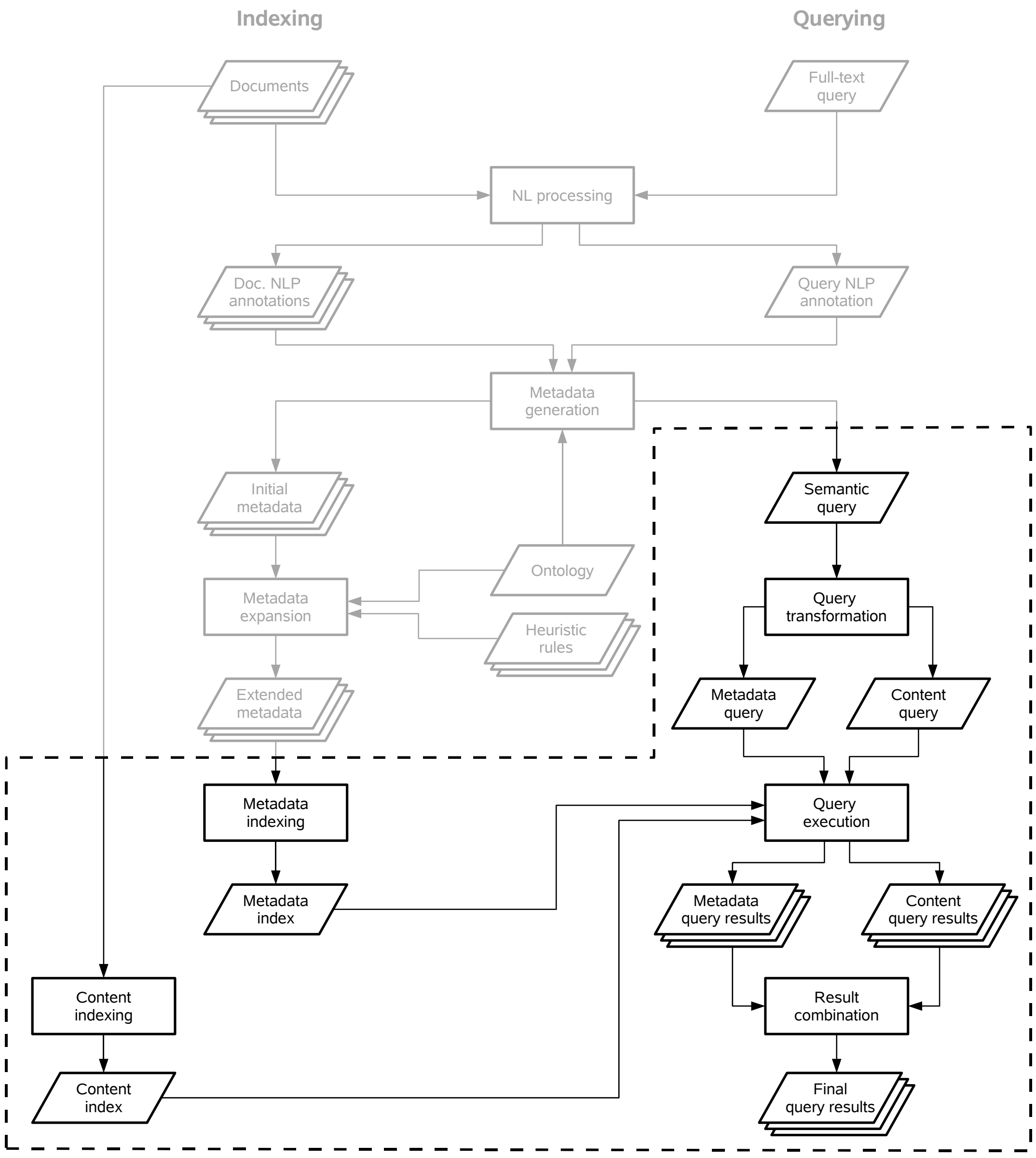


Figure 9.1: Indexing and querying in the ontology-supported IR process

In this chapter, I will show how to exploit the generated semantic metadata, and how to implement semantic querying that meets the requirements SCALABILITY and ONTOLOGY IMPERFECTION.

As already discussed in Chapter 5, to meet the requirements two major problems have to be solved:

- The *semantic metadata* (and also the metadata query) *has to be represented in a form that is processable by legacy full-text search engines*. It is also important that the syntactic similarity of the full-text representation indicates semantic similarity of the original semantic metadata. If this is achieved, the syntactic similarity measure that is used by the full-text search engine will provide intuitive results.

- To deal with ontology imperfection, the *results* of the metadata query *should be combined* with the results of the content query. There are many possibilities to do this, and the best possible solution should be chosen after a careful analysis.

This chapter will discuss these two problem areas in greater detail. The topics discussed here are shown in the context of the whole IR process in Figure 9.1.

9.1 Indexing

9.1.1 Problem analysis

As was already described in Chapter 5, the indexing step generates two indexes: one index for the semantic metadata, and one traditional full-text index on the document content. The latter task is trivial and therefore not discussed further. In the following, the creation of the semantic metadata index is discussed in detail.

As was shown before, the IRCON information model consists of three parts: the textual, the conceptual and the temporal parts. All parts contain a weighted set of metadata elements.

It is quite straightforward to create a full-text search friendly representation of the textual and conceptual parts. The textual part contains text snippets that can be indexed without any change. The conceptual part contains ontology instances. These instances have a unique identifier (URI) that can be represented as a string, and can be indexed as a usual, natural language term[1].

[1]Of course, the full-text representations of different kinds of metadata elements must be clearly identifiable later on. E.g., if we search for the ontology instance with the URI “Bush”, we do not want to get any documents that contained “Bush” in the textual part of the metadata. Lucene supports the definition of different index fields in the text index, so it is easy to do this separation. In search engines that do not support such index fields, an alternative solution would be to encode the metadata part information into the generated full-text term. E.g., generate “uri_Bush” instead of “Bush” for the conceptual part.

The vector space model[2] that is used by many full-text search engines (FTSE), including the Lucene library that I used, is based on the assumption that document terms are independent[3], and the only meaningful relation among them is equality. I.e., two words are either equal (the same), or not equal. Using these assumptions, the list of all t terms in all documents form a t-dimensional vector space where the cosine measure between the t-dimensional term vectors can be used as a meaningful similarity measure[4]. These assumptions are valid for the textual and conceptual parts, therefore it is easy to create the mapping between these parts and the VSM.

The temporal part, however, is more complicated because these assumptions do not hold. Temporal intervals are clearly not independent, and there are many possible relations among them (see Table 6.1). It is therefore not trivial how to make the time dimension compatible with a full-text search engine. Probably this is one of the causes why the existing approaches ignore temporal information. Therefore for the temporal part, a custom solution is needed, which will be explained in the next subsections.

Before discussing the temporal part, however, a minor issue has to be solved. The IRCON information model contains a weighted list of elements, whereas classical FTSE are designed to index a flat list of words (the document content). So, it must be shown how to represent the metadata element weights in the full-text index.

9.1.2 Representing metadata element weights

FTSE use some variation of the TF-IDF measure to estimate the weight of a term in the document content[5]. Therefore, the task is to achieve that the TF-IDF measure used by the FTSE yields the same weight for the full-text term representation of a metadata element as the metadata element originally had in the semantic model.

A possible approach to achieve that would be to control the term frequency part of the TF-IDF measure, i.e., to increase the frequency of the full-text representations of more important metadata elements. I.e., if one metadata element has twice as much weight than the other element, the frequency of its full-text term representation should be twice as much, too.

There are two problems, however, with this naive approach. First, the IDF part of the TF-IDF measure is still there, and it is very hard to control because it is calculated over the whole document repository. Luckily, some FTSE allow users to adjust the TF-IDF measure. E.g., it was possible to set the IDF part of the Lucene TF-IDF measure to a constant 1.0. Another problem is that by manipulating the frequency of terms we have to discretize the continuous weights of the metadata elements. I.e., we have to decide what will be the frequency of the elements

[2]or "bag of words" model

[3]This assumption is clearly not valid in the general case because there are many words that appear together in many cases. Consider e.g., the words "computer" and "network". Experience with full-text search systems shows, however, that this simplification does not significantly decrease retrieval performance and it simplifies the model dramatically.

[4]For more details on the VSM see Section 2.3.

[5]In the TF-IDF measure, TF stands for term frequency, IDF stands for inverse document frequency. The basic idea of this measure is that a specific term is the more important for a document, the more frequent it appears in that document, and the less frequent it appears in other documents in the collection. For more details see Section 2.3.

with the weight 1.0. E.g., if we decide that the frequency will be 10, we can only represent the metadata weights with a 0.1 accuracy. Clearly, there is a trade-off between the accuracy of the weights and the size of the full-text index: bigger accuracy means more generated terms, and a bigger index. In the IRCON protoype, I transform metadata weights with 0.1 accuracy.

E.g., with 0.1 accuracy the following conceptual metadata part

```
{George_H_W_Bush:0.7, USA:0.8}
```

would be represented in the full-text index as

```
"George_H_W_Bush George_H_W_Bush George_H_W_Bush
George_H_W_Bush George_H_W_Bush George_H_W_Bush
George_H_W_Bush USA USA USA USA USA USA USA USA"
```

9.1.3 Representing the temporal part

Kalczynski and Chou provide a very interesting solution for the problem of representing temporal information in a full-text search friendly way [KC05]. They discretize time intervals into temporal granules, such as days. E.g., the time interval **(2006-01-01 – 2006-01-03)** can be represented as a sequence of three days:

```
{ 2006-01-01, 2006-01-02, 2006-01-03 }
```

This approach works even for fuzzy time intervals. In this case, a weight is also associated with the granules, which represents the membership value of the granule in the fuzzy time interval. E.g.,

```
{ 2006-01-01 [1.0], 2006-01-02 [0.7], 2006-01-03 [0.3] }
```

For these granules, it can be assumed that they are independent from each other, and that the only interesting relation among them is equality. Using such assumptions, time granules can also be represented using virtual words, and indexed by full-text search engines. I also follow this approach to create a representation of the temporal part of the information model that is compatible with FTSEs. I use days as granules because day granularity is adequate in the application domain of history. It is important to note, however, that the transformation to granules is independent of the chosen granularity level. Finer (or coarser) granules than days can be chosen if required by the application domain.

What is missing from the approach of Kalczynski and Chou is a solution for scalability. For long time intervals, and fine granules, the transformation described here will result in a very large number of granules. E.g., to represent the **(1000-01-01 – 2000-01-01)** time interval with day granules, more than 360000 granules are needed. Such a big number of granules cannot be efficiently indexed with full-text search engines[6].

[6]Many full-text search engines index only the first several thousand words of a document.

9.1.4 Solving the scalability problem

To solve the problem of scalability of the naive approach on long time intervals[7], I use the idea of aggregation that is very common in the area of data warehousing (see e.g. [Kim96]). The main idea is that if the number of the generated granules exceeds a given threshold[8], the time interval should be converted using coarser granules. E.g., the time interval **(1001-01-01 – 2000-12-31)** should be encoded using centuries as granules, instead of days. Of course, some information is lost with this approach. My hypothesis is, however, that this information loss causes only a minimal decrease of IR effectiveness because in the case of very long time intervals the user information need is usually not expressed using very fine granules. E.g., using the example above, a typical user would express the interval as “the second millennium”, and would not use days.

The switch between the granularity levels happens simply by taking the average of membership values on the smaller granularity as the new membership value on the bigger granularity. E.g., if we have the following granule/membership value pairs on the year granularity

```
1001/0.1, 1002/0.2, 1003/0.3, 1004/0.4, 1005/0.5,
1006/0.6, 1007/0.7, 1008/0.8, 1009/0.9, 1010/1.0
```

the membership value for the **(1001 – 1010)** decade granule would be 0.55.

9.2 Querying

As was already described in Chapter 5, *two full-text queries are generated* from the semantic query. The first, *metadata query* searches the full-text index representing the semantic metadata of the documents. The second, *content query* searches the full-text index containing the indexed document contents. After executing these two queries using an FTSE, the *results are combined*, and the combined result constitutes the final query results that are returned back to the user.

9.2.1 Generating the metadata query

The same idea that was used during indexing can also be applied to generate the metadata query out of the semantic query. However, some search engines, including Lucene, also support weights on query terms. Using such search engines, the trade-off between the accuracy of weights and the length of the query does not exist, and more precise queries can be used.

E.g., for the semantic query:

```
{George_H_W_Bush:0.7, USA:0.8}
```

[7]which are frequent in the domain of history

[8]In my work I use a threshold of 400 granules.

the generated Lucene query with 0.01 accuracy would be:

```
"George_H_W_Bush^70 USA^80"
```

The `term^weight` notation is an element of the Lucene query language and denotes the weight of the given query term. In my prototype, I generated query weights with 0.01 accuracy.

9.2.2 Searching with multiple temporal granularities

Although the aggregation approach introduced in Section 9.1.4 solves the problem of scalability for indexing, it introduces a new problem. If we have more than one granularity, it is no longer clear which granularity should be used in the metadata query. Even if the user specifies the time dimension in the query very precisely (e.g., using the day granularity), we still would like to retrieve documents which could be indexed only with coarser granularities (e.g., with centuries). Conversely, if the user searches using a coarse granularity, we still would like to return documents that were indexed using finer granularities.

Clearly during query time, the system cannot know for each document, which granularity was used when that very document was indexed. My solution to the problem is to index documents using all granularities, where the threshold on the number of granularities is not exceeded. Similarly, the temporal part of the query is also transformed into queries using all of the possible granularities (where the threshold is not exceeded). In this way it does not matter which granularity was used during document indexing, and which is used during query time, matching documents will always be found.

Of course, the approach is redundant because a document will be found for all of the granularities, which were used at indexing time, and which are the same or coarser than the smallest query granularity. Based on my practical experience, however, this redundancy does not cause performance problems.

9.2.3 Supported temporal relations in metadata queries

Because the temporal part must be transformed to granules before indexing, and because the VSM supports only equality on query terms, the only temporal relation that can be expressed among temporal intervals is a kind of “intersects” relation. More precisely, the full-text search engine finds documents for a temporal interval specification in the query, where at least one temporal interval in the temporal part of the document metadata intersects one of the query intervals (see the “intersects” relation in Chapter 6).

However, the “intersects” relation, which is indirectly expressed using full-text search, has some nice operational characteristics. Please recall that the classical “intersects” relation is binary, and the fuzzy version considers only the maximal membership value of the time points in the overlapping area of the time intervals. In contrast to that, full-text search results on the temporal granules also consider the size of the overlapping area. E.g., if the user searches for the date 1945-05-08, documents describing exactly the requested date will have higher scores than documents describing the whole forties, or the whole 20th century. It is because the TF-IDF score incorporates also the length of the document vector (see Equation (2.1)). The longer the "document" is (i.e., the bigger the total number of granules is) the less score will be given for the same number of matching granules.

Of course, this means that none of the (fuzzy) temporal relations that were discussed in Chapter 6 are effectively used during retrieval. On the other hand, the fuzzy temporal model and the fuzzy temporal relations are used during metadata generation as discussed in the previous chapter.

An alternative approach for the temporal part would be that it is implemented using a special solution, instead of using various "hacks" to tweak the temporal model into the less powerful VSM. Indeed, we experimented with a solution for fuzzy temporal querying on top of the relation database technology [Zhe04]. In this solution, arbitrary fuzzy relations could be used in queries. It was even possible to efficiently calculate the results because we used various optimizations to avoid calculating the result of fuzzy relations for each fuzzy interval stored in the database. A small set of result candidates could be selected using out of the box relational indexes based on some mathematical characteristics of the fuzzy interval relations.

It must be noted, however, that his approach returns a non-ranked set of temporal intervals that are in the required relation with a reference fuzzy interval. Ranking these intervals in memory can be very expensive if there are many results.

Another problem with this approach is that one has to deal with two separate indexes: one for the temporal information, the other for the conceptual and textual part. This means that a user query should be split into two parts, and the results should be combined after running the two parts separately. In the typical case, however, both parts would simply return too many results, and it would be very inefficient to combine the results later in main memory. E.g., consider the query about "Bush" in "2001". If we run the query with both query terms together, there will be not so many documents that are about Bush in the year 2001. However, if we run the queries "Bush" and "2001" separately, there will be lots of results for both subqueries (but especially for the very general temporal query). Based on some experiments, one can expect result sizes that are orders of magnitude bigger than the result size of the combined query. These huge results must be completely loaded into main memory, and scanned to find documents that fulfill both query phrases at the same time. For big document collections this is clearly not a feasible solution. With the completely full-text solution, on the other hand, it is not even necessary to load the full result set into the main memory, if the user is interested only in the first several hundred results (or even less), which is the typical case in Internet applications [SJ04, JP01, JSS00, JS03].

9.2.4 The need for query combination

As was discussed in Chapter 5, the results of the metadata query are not adequate in many cases because of various imperfection issues. For a generic document repository it cannot be guaranteed that the ontology covers all topics of the repository (or covers them well enough). Moreover, even if the ontology would theoretically provide adequate coverage, it can still happen that for a specific document no semantic metadata was generated. In these cases, the metadata query would clearly not return adequate results. As was shown in Chapter 5, the combination of content search results with the metadata query results can help to solve this problem.

9.2.5 Generating the content query

Before the results of the content query can be combined with the metadata query results, the content query has to be generated based on the semantic query.

For the textual part of the semantic query it is easy to specify the content query: the text snippets are simply copied, and the query weights are specified as described above at the metadata query.

For the conceptual part, the lexical layer of the ontology is exploited. For each OI in the conceptual part, the OI label, and also the OI synonyms, are added to the content query. While the label added with the weight of the OI, the weights for synonyms are reduced[9] to compensate the fact that synonyms contain some “noise”[10]. E.g., for the OI GEORGE_H_W_BUSH the text phrases “George H. W. Bush”, “George Bush”, “President Bush”, “President George Bush” and “Bush” are all added to the full-text query, where “George H. W. Bush” is the OI label and the other phrases are the synonyms[11].

Finally, for the temporal part, a similar strategy can be followed that was described during indexing. An additional problem is, however, how to represent the granules in text form. I.e., for the day granule **2006-10-01** the following textual representations are all valid: “2006-10-01”, “October 1, 2006”, “October 1, 2006”, “October 1st 2006”, “10/1/2006” etc. Clearly, all these representations can blow up the final content query significantly, and it is still cannot be guaranteed that all relevant documents are found because in the document texts many time specifications are actually time intervals. E.g., the text “early October 2006” would semantically match **2006-10-01** but syntactically does not match any of the textual representations. Still, the granule strategy seems promising but when setting the threshold on the number of granules, the syntactic variability has to be considered. E.g., if each granule has 10 syntactic representations on the average, the threshold on the number of granules at a specific granule level should be 10 times less than during indexing time to avoid a blow up in the query size[12].

9.2.6 Combining search results

After we discussed how to generate the semantic and content queries, the last unresolved issue is how to combine the results of these queries. There is no easy answer in general because it is not a trivial question which is the best strategy for query combination.

[9] in the current implementation by 30%

[10] e.g., both George W. Bush and George H. W. Bush has the synonym “Bush” in the ontology

[11] Of course, the exact list depends on the actual list of synonyms in the ontology.

[12] In the actual IRCON implementation, I generated full-text representation of the temporal part only for the year granules because here the syntactic variability is minimal. Experimenting with other granules in the content query, and especially the negative or positive effect of having them in the content query, is a subject of future research.

In the IR literature, there are many approaches for combining evidences of relevance or search results. I reviewed these approaches, and selected some promising algorithms to experiment with.

IR systems that are built on the inference network model [MC04], or on the belief network model [RM96], are capable of combining different sources of evidence to estimate the probability that a document is relevant for a given user information need. E.g., the approach described in [SRN04] combines the results of various subqueries that are generated from the original user query using different linguistic relations from a thesaurus. These approaches use Bayesian networks [Pea94] for the combination of evidences. In these networks, results returned by the subqueries are viewed as probability estimations that a specific document is relevant for the user's information need. Although in general, there are many possibilities to combine these probability estimations to estimate the combined probability that a document is relevant for the user's information need, most approaches uses the logical OR combination. With this strategy, a document is deemed to be relevant for the information need, if any of the subqueries returned it as relevant. To estimate the combined probability of a document, i.e., its relevance score, the following formula is used:

$$rel(d) = 1 - \prod_i (1 - rel_i(d)) \tag{9.1}$$

where $rel_i(d)$ denotes the relevance estimation returned by the i$^{\text{th}}$ subquery for a document d[13]. I will denote this formula as *Bayes OR* from now on.

A second big area of IR that is concerned about combining search results is the area of *metasearch* [Cro00, AM01]. At metasearch, the task is to combine the results of different, independent query engines into one coherent list of results. Because I also have two independent queries[14], the various metasearch algorithms can be directly applied in my case. Although there are lots of different approaches, it seems that the so-called *CombMNZ*[15] algorithm provides very good results, in spite of its simplicity [AM01]. Probably for this reason it is one of the standard algorithms in this area. The formula to calculate the relevance score of a document according to CombMNZ is the following:

$$rel(d) = n_d * \sum_i rel_i(d) \tag{9.2}$$

where n_d denotes the number of systems that returned non-zero relevance score for a document d.

Finally, Vallet et al. also proposed a custom combination algorithm in their work [VFC05] for the combination of semantic and full-text search results. Although this algorithm is not well-known, I included it in my experiments because this work was the first to explicitly examine

[13]The formula uses the fact that $p_1 \vee p_2 = \neg(\neg p_1 \wedge \neg p_2)$ for the p_1 and p_2 logical variables. Moreover, it is also a well-know axiom of the probability theory that $Pr(\neg p) = 1 - Pr(p)$ where $Pr(p)$ denotes the probability that the logical variable p is true.

[14]Although the queries are executed using the same FTSE, they are operating on two independent indexes, therefore their results can be considered as independent.

[15]The MNZ part in the name is an acronym of "multiply by number non-zero".

the combination of full-text and semantic search. The formula to calculate the relevance score for a document according to this strategy is the following:

$$rel(d) = t * \sum_i rel_{sem}(d) + (1 - t) * rel_{fts}(d) \tag{9.3}$$

where t is a parameter of the algorithm, $rel_{sem}(d)$ denotes the relevance score returned by the semantic search for a document d, and $rel_{fts}(d)$ denotes the relevance score returned by the full-text search for a document d. Vallet et al. propose to set the t parameter to 0.5 by default, and change it to 1.0 when $rel_{fts}(d) = 0$, and change it to 0.2 when $rel_{sem}(d) = 0$. It is clear that these values are somehow ad-hoc, and changing the t value affects the results of the algorithm. I did some experiments with various t values with the data that I used for the evaluation of my solution (see Chapter 12). Indeed, I found that on the one hand, the exact value of t greatly affects the results but on the other hand, this effect is inconsistent across different user queries. I.e., increasing the t value helped increase the quality of the results for one query but it decreased retrieval performance for other queries at the same time. Therefore, I used the same t values that were proposed by Vallet et al.

In general, it is impossible to say which of these strategies for query combination is better than the others. Therefore, I ran experiments with the different strategies as part of the evaluation of my system. The results of this evaluation are reported in Chapter 12.

9.3 Comparison with related work

The exact implementation of the indexing and querying steps are highly dependent on the chosen information model. Therefore, a direct comparison with other state of the art approaches is nearly impossible. There are some aspects of the systems, however, that can be compared. The systems that I mention here were already analyzed in Chapter 4 in general but the following discussion focuses more on the issue of indexing and querying. I consider here only systems that somehow involve full-text search techniques because the decision to use full-text search for IR is fundamental to my approach.

The first category of systems avoids the combination of indexes (or queries) and encodes all needed information into the full-text index. The QuizRDF system [DW04] encodes the text literal part of simple RDF statements into the same full-text index that is used to store the full-text information. The OWLIR system [FMJ+05, SFJ+02] stores arbitrary RDF triples in the full-text index that are encoded using the swangling method. These encoded triples can be indexed and queried together with the document text using traditional FTSE. The KIM system changes the original document content by adding unique semantic identifiers to the text where named entities were identified. After this expansion, the text can be indexed and queried with a traditional FTSE. Naturally, these systems do not have the problem of combining the results from many query sources. However, they also lose the possibility to control how the full-text search affects the results of the semantic search.

The second category of systems explicitly combines the results of full-text search and semantic search or semantic reasoning. In the fuzzy-DL based system of Zhang et al. [ZYZ+05] the results of full-text search are integrated into the DL ontology as special fuzzy DL statements, and classical ontology reasoning is applied on this ontology to answer queries. In the system of Rocha et al. [RSA04], a FTS is executed over the textual representations of ontology instances. Later, these initial results are expanded using a spread activation algorithm. Finally, Vallet et al. [VFC05] combines the results of a semantic search (executed using ontology reasoning on an ontology that closely resembles the VSM), and full-text search using their custom query combination strategy. As was discussed in previous chapters, I do not consider IR based on ontology reasoning as a viable approach. The system of Vallet et al., however, uses an approach that is very similar to my solution. The only important difference is that they use ontology reasoning for the semantic search part, while I use FTS also for this part (the metadata query) to achieve scalability. Although in these approaches it would be in principle possible to exploit the lexical layer of the ontology, they do not make use of this possibility and generate only simple full-text searches.

As was previously discussed in Chapter 4, none of the systems deal with temporal information. As was shown in this chapter, representing and querying temporal information is far from trivial, therefore I consider this part a significant contribution of my work when compared to the related approaches.

9.4 Summary

In this chapter, I gave a detailed account on the indexing and querying steps of the ontology supported IR process. First, I showed how to represent the semantic metadata so that it can be indexed and queried by traditional full-text search engines. Representing the textual and the conceptual part was easy because the assumption of the vector space model (VSM) holds that the elements are independent and the only meaningful relation among them are (in)equality. The only problem to solve was how to represent element weights in the full-text index.

Representing the temporal part was much more challenging because there are lots of relations among time intervals that we do not want to lose during indexing. My approach to this problem was to discretize time intervals into so called granules. For these granules the assumptions of VSM hold. I also addressed the problem of performance that may occur with this approach if too many granules are generated.

In the second part of the chapter I discussed how to combine the results of metadata and content queries to achieve results that are robust against ontology imperfection. I introduced several query combination algorithms that were used during system evaluation. I also described how to enhance the content query by exploiting the lexical layer of the ontology and the temporal part of the semantic query.

To summarize, my approach is unique in the sense that it supports the indexing and querying of temporal information and that it allows for exploiting the lexical layer of the ontology in the content query.

Chapter 10

Ontology development

An ontology plays a central, crucial role in an ontology-based information system. It is not easy, however, to build an ontology. The development of ontologies is comparable in complexity with the design and development of complex software. Therefore, similarly to software development, methodological support is needed to guide the development of ontologies.

This chapter reviews the state of the art of ontology development, and reports on our hands-on experiences during the VICODI project. Based on this, I describe the methodology that evolved during the project and was customized to the needs of ontology-based systems focusing on information retrieval. The methodology was also used in this thesis to build an ontology for evaluation purposes.

Parts of this chapter, and a more detailed discussion of ontology development methodologies and design principles were already published in [Nag05] and in [Nag07].

10.1 State of the art of ontology development

Ontology development is a very complex, creative process, therefore a methodology that coordinates its various activities is crucial for its success [GPFLC04]. Ontology development is comparable in complexity with software engineering, a field where already lots of mature methodologies exist such as the Rational Unified Process [Hun03] or Extreme Programming [Bec00]. Unfortunately, because the field of ontologies is not so mature yet as the field of software engineering, currently there is no set of established, generally accepted methodologies. There were numerous methodologies proposed, however, and some of them are quite elaborate.

The main purpose of a methodology is to define the life cycle of the ontology development process, i.e., the order in which the various ontology development activities should be executed. Most of the methodologies propose the *evolving prototypes* approach where development activities are executed in cycle, producing more and more mature versions of the ontology.

In the recent decades, lots of methodologies were proposed. Some good studies and overviews on ontology methodologies, which can be recommended for more details, are the following: [FLGP02], Chapter 3 in [GPFLC04], and [JBCV98]. Here I briefly review two methodologies: the On-to-Knowledge methodology and the METHONTOLOGY methodology. These are relatively mature, detailed methodologies that influenced our work in VICODI the most.

10.1.1 The On-to-Knowledge methodology

The *On-to-Knowledge methodology (OTK)* [SAB+03, Sur03, SS02] concentrates on building knowledge based systems where ontologies form an important part of the system. This methodology defines two orthogonal processes, the *Knowledge Process* and the *Knowledge Meta Process*. The former describes the process of ontology usage, the latter guides the ontology creation. We are interested here only in the Knowledge Meta Process.

OTK defines the following steps as part of the Knowledge Meta Process (see also Figure 10.1):

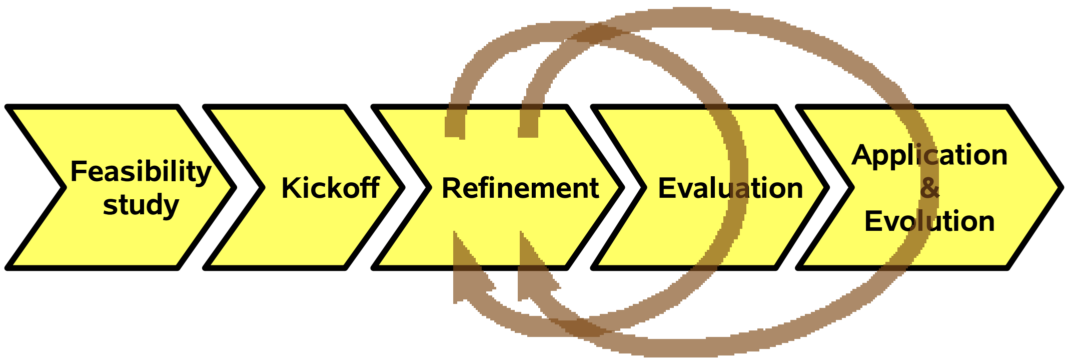


Figure 10.1: OTK steps

Feasibility Study: OTK follows the CommonKADS methodology [SAA+99] in order to decide whether it makes sense to start the project, i.e., to build the ontology.

Kickoff: During this phase the ontology requirements are finalized: the exact goal and the scope of the ontology is determined. According to the middle-out strategy, an initial list of important entities is collected during a brainstorming session and a list of relevant experts and knowledge sources is compiled. Design guidelines are also provided which will guide the development process. Competency questions are collected which can be used later to validate the ontology.

Refinement: In this phase, relevant knowledge is extracted from the identified knowledge sources (and from human experts) and is formalized. This is the main development phase of the methodology.

Evaluation: OTK describes three types of evaluation that can be conducted:

- *Technology-focused evaluation*: This includes checking the syntactic correctness and semantic consistency of the ontology, its performance, modularity, maintainability etc.

- *User-focused evaluation*: This includes checking whether the ontology contains all of the information which was identified in the ontology specification document. A usage pattern based evaluation is also part of this process where it is checked that all parts of the ontology are really used, i.e., there are no unnecessary parts in it.
- *Ontology-focused evaluation*: This checks the semantic correctness of the ontology. Both philosophical methods (such as OntoClean [GW02]) and ontology evaluation rules [GP01] can be used to find incorrect conceptualizations.

Application & Evolution: It is important to see that no ontology is ever complete as our understanding of the world (i.e., our conceptualization) evolves constantly. It is also possible that the needs of the ontology users or applications change over time, i.e., other parts of the domain will become relevant. During this phase, these changes are reflected, and the ontology is evolved.

OTK proposes a cyclic ontology development process, i.e., the *Refinement* and *Evaluation* phases are iterated until a stable, high-quality ontology version is reached. Later, the lessons learned during the *Application & Evolution* phase can also initiate a new development cycle.

10.1.2 The METHONTOLOGY methodology

The METHONTOLOGY methodology [LGPSS99, GP97] describes a process similar to OTK but it focuses more on the ontology development (it does not address the Knowledge Process) and describes the conceptualization activity in much more detail.

METHONTOLOGY groups the ontology development activities in three categories. *Management activities* include activities that are common to all kinds of projects. *Development oriented* activities form the core of the development process, and they are normally conducted sequentially. The activities of OTK all belong to this category. Finally, *support activities* are crucial for the success of the development activities (and such for the whole project), and they are conducted in parallel with one or more development activities. Many of them (like documentation) are performed continuously throughout the whole project. A more fine-granular categorization and a detailed account on ontology development activities is found in Section 3.1 of [GPFLC04].

METHONTOLOGY defines an evolutionary type of life cycle (similarly to OTK), which is shown in Figure 10.2. This means that the development activities are executed iteratively throughout the development process. From the project management activities, planning is done at the very beginning, control and quality assurance are continuous. All of the integral activities are done continuously, although the amount of knowledge acquisition, integration and evaluation decreases as the ontology matures and its structure stabilizes.

In contrast to OTK, METHONTOLOGY separates the steps of conceptualization, formalization and implementation. During conceptualization, a model of the relevant domain knowledge is created that is is not necessary suitable for reasoning and can be in any form which is understood and accepted by domain experts (e.g., Excel sheets, a mind map, semi-structured text). This model is transformed into a chosen formalism (e.g., first order logic [Fit96], F-Logic

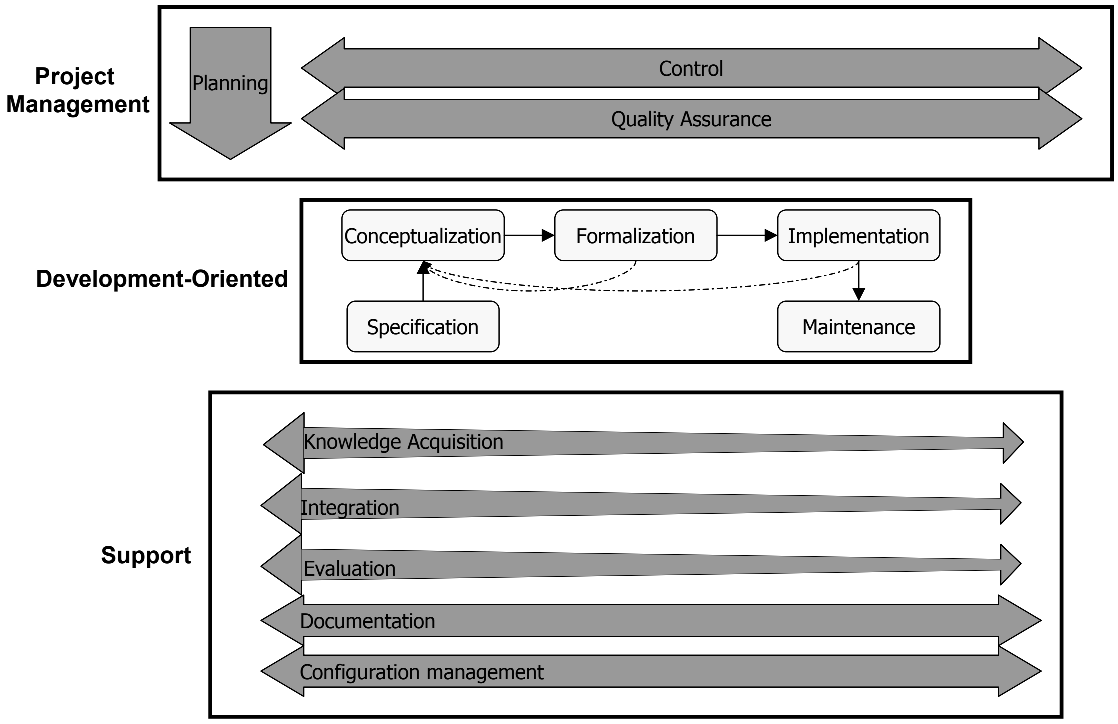


Figure 10.2: Methodology lifecycle

[KLW95], or description logic (DL) [BCM+03]) during the *formalization* step. This formal representation is semi-computable, i.e., it can be rewritten into a suitable syntax quite easily, which can serve as an input for a reasoner. Finally, during the *implementation* step, the formal representation is codified in a specific ontology language (such as OWL-DL [PSHH04]), which can be directly executed in a suitable reasoner.

The main strength of this methodology is the detailed description of the conceptualization activity. METHONTOLOGY describes various artifacts which specify different aspects of the conceptual model, such as the glossary of terms, the concept dictionary, the concept taxonomy or the table of instances. The conceptualization process is a controlled development of these artifacts.

More detailed descriptions of the documents forming the intermediate conceptual model of METHONTOLOGY and the steps that should be taken can be found in Section 3.3.5 of [GPFLC04] and in [GP97]. A detailed practical example using METHONTOLOGY is described in [LGPSS99].

10.1.3 Conceptualization strategies

Conceptualization is probably the most important, and definitely the most complex task of the ontology building process. There are different strategies for defining a model that provides a specification of our conceptualization [GPFLC04, SS02].

Following the *top-down* strategy one starts with the most general concept (e.g., THING) and tries to refine the ontology structure along different distinguishing notions. This strategy is usable mostly in the case of top-level, philosophical ontologies.

The *bottom-up* strategy starts with a suitable set of information resources (databases, documents) which should be described by an ontology. In the first step, interesting entities are collected from these resources that are worth to be included in the ontology. This process can be supported by information extraction (IE) and ontology learning tools like Text2Onto[1] or Amilcare[2]. Later on, the ontology engineer tries to find common superconcepts and superproperties of the identified concepts and properties, and specifies the proper concepts for the identified instances.

The advantage of the bottom-up strategy is that the ontology will definitely describe the target document corpus or database(s) properly, i.e., it will describe the "information supply" well. On the other hand, there is a danger with this approach that the ontology will be too focused on a specific information resource, thus it will not be reusable. This strategy is suitable for very low-level, application ontologies, where a specific information resource should be semantically described, and not recommended for high-level ontologies, which should be independently developed from specific databases or document corpora.

Finally, the *middle-out* strategy starts with a list of most important ontology entities (concepts, properties) which can be collected during a brain-storming session. This approach can be used for both low-level, application ontologies, or medium-level, domain or task ontologies where a list of most important concepts can be easily identified at the beginning.

10.1.4 Design guidelines

It is important to note that ontology methodologies only describe the high-level development process but they do not provide practical guidelines how a "good" ontology should look like, and how to model certain common issues. Such practical guidelines are very important, however, during the daily work of the ontology engineer. Just consider that design patterns [GHJV94] in the area of software development are very popular, and generally accepted as useful.

This fact was already realized by the Semantic Web community, and the Semantic Web Best Practices and Deployment Working Group was formed as part of the W3C[3], which started to develop guidelines for practical ontology modeling issues (e.g., [Noy04, Rec04]). A detailed list of practical guidelines was also published in [Nag05], and in [Nag07]. These guidelines are not detailed here because they are beyond the scope of this discussion.

10.1.5 Summary

To summarize the discussion on ontology development methodologies, the following important points can be enumerated:

[1]http://ontoware.org/projects/text2onto/

[2]http://nlp.shef.ac.uk/amilcare/

[3]`http://www.w3.org/2001/sw/BestPractices/`

- Most methodologies advocate a highly *iterative approach* to deal with the complexity of ontology development.
- There seems to be disagreement whether it is a good idea to separate the conceptualization, formalization and implementation steps. Some methodologies (such as OTK) merge these steps. As a matter of fact, most ontology development tools also assume that the conceptualization is directly specified in the implementation model.
- Ontology specification and scoping is important. It guides the process, and helps to make well-founded modeling decisions.
- A list of design guidelines that most practitioners can agree on is still missing for ontology development. Unfortunately, methodologies themselves do not define such guidelines. However, the first steps were already made to create guidelines how to model some common constructs in ontologies.

10.2 VICODI experiences

During the VICODI project a large ontology of European history was developed. This ontology was required for the VICODI prototype.

During the ontology development process we tried to apply the ideas presented in the previously described methodologies. As a result of our trial-and-error attempts to apply elements of these methodologies, it turned out which parts could be reused readily, which parts had to be changed, and which elements were missing. In the following I report on these experiences.

The VICODI experiences were already published in more detail in [NDO05, Nag04].

10.2.1 Requirements of an ontology supporting IR

The clear specification of the ontology goal is crucial for the successful development of the ontology [vSW97, SS02]. The goal of the VICODI ontology was the same as the goal of the ontology for the evaluation of IRCON: to *support the ontology-based IR process*. That is, to support the automatic creation of semantic metadata, and the search for this metadata. In the following, I discuss what requirements follow from this high-level goal.

The ontology defines the elements of the information model that was described in Section 8.1. I.e., if the ontology does not contain a specific element, it cannot be represented in the semantic metadata. This implies that for a successful and comprehensive metadata generation the ontology must contain *lots of ontology instances*.

In addition to instances, the ontology must also contain *relations* among instances and *time information* to support the heuristic rules for metadata expansion that were described in Section 8.6.3. It is also important to note that the heuristic rules do not profit from extensive, deep concept or property hierarchies. They only profit from concepts and properties that are directly used in the rule definitions.

To summarize, the ontology has to contain lots of instances and relations, and it is not required that it defines a complex concept and/or property taxonomy.

10.2.2 Conceptualization strategy

In VICODI, we first tried to follow the middle-out strategy that is advocated by OTK and which is also compatible with the METHONTOLOGY methodology. I.e., we started the ontology development process with a brain storming session, where most important concepts were identified. We also collected a list of competency questions. Finally, our domain experts manually marked-up some selected "miniworlds", i.e., a small list of documents in some selected domains. This latter was needed to validate whether the core concepts identified during the brain storming session really cover the entities that appear in documents.

The first real problems appeared when we tried to refine the concept and property taxonomies of the ontology. The domain experts started to specify very deep and detailed concept and property hierarchies in the specific domain of history where they were especially interested in. However, other important parts of the domain were not covered at all. Even worse, the domain experts could not understand the hierarchies that other experts proposed, simply because they did not have such deep expertise or interest in that very specific subarea. Therefore, no consensus of those detailed hierarchies was possible, and discussions ended up with endless philosophical debates.

Moreover, there were only few or absolutely no documents in the document repository about the domains that were modeled in detail, while some other, popular areas where there were lots of documents were not modeled by any experts. As a conclusion, it became clear that the domain was simply too wide to be modeled in a detail big enough to satisfy a historian. However, it also turned out that such a detailed modeling was not necessary at all because the heuristic rules that exploited the ontology were not prepared for such a detailed ontology structure. Therefore, the participants of the project agreed after a while that a very detailed concept and property hierarchy is actually a waste of project resources, and should be avoided.

I believe that the described situation is very common in ontology building projects. It seems extremely hard to decide how detailed a specific domain should be modeled. A complete model that domain experts tend to go for seems to be possible only in very limited, specific, technical domains. Measurement units in physics, or the elements of the periodic system in chemistry are good examples of that. In more complex domains, however, it is very hard to find the abstraction level where the ontology should stop. Unfortunately, this issue is not addressed adequately in the ontology methodology literature, where sometimes very small, toy ontologies are presented as examples (e.g., [LGPSS99]).

In the VICODI case, the solution was to concentrate on the instances that actually appeared in the documents, and extend the concept and property hierarchy only in cases where a new heuristic rule for metadata extension could be specified. Although the resulting ontology did not describe European history in the detail adequate for a historian, it was much more useful for the task on hand, namely supporting ontology-based metadata generation and IR. In other words, VICODI showed that scoping is crucial for an ontology development project. It also showed that for IR ontologies, a bottom-up approach for conceptualization is better than a middle-out or top-down strategy. However, an initial brain-storming session, which is characteristic for the beginning of a middle-out strategy, can help to focus the ontology development.

10.2.3 The need for an intermediate model

At first, we tried to define the ontology directly in the implementation formalism, as it is suggested by some methodologies (such as OTK). It seemed to be a pragmatic, simple approach to merge the conceptualization, formalization and implementation steps and directly use the available graphical ontology editors to develop the ontology.

Our experience showed, however, that the ontology formalism of the KAON system we used at that time was cumbersome to use for our domain experts. KAON uses a slight extension of RDFS that — similarly to OWL, the current W3C standard for ontologies — supports only binary relations between concepts. In history, however, time plays a very important role, and therefore most of the relations are timed. This requires relations of higher arity which can be represented in RDFS (or in OWL) through reification[4]. This resulted in ontology structures, however, that were not easily understandable for our domain experts, and which were very cumbersome to edit. This fact considerably slowed down the knowledge acquisition process, i.e., getting new instances and relation instances to the ontology.

Another problem was that most of the time, specifications in history were imperfect (see also Section 3.6), and this fact could not be represented in RDFS. Although elements of the temporal model that was described in Chapter 6 were already devised during the VICODI project [NM03], ontology support for the fuzzy time model was implemented only later, after the project ended. Still, our domain experts wanted to explicitly denote the imperfection of temporal specifications somehow.

These experiences show that there are some very good reasons to separate the conceptualization and implementation steps, and do not force domain experts to specify their conceptualization directly in the target ontology formalism.

The main problem with the direct approach is that most formal ontology languages are designed with the motivation to provide for several reasoning constructs. A natural requirement from the user side is that these reasoning operations should be calculated efficiently, or at least the reasoning should be decidable. Clearly, the more expressive an ontology language is, the more complicated the reasoning will be, and it is easy to reach the point where the reasoning will no longer be decidable. Therefore, these languages make various compromises in the area of expressiveness in favor of efficiency. This means that in many cases, the knowledge required to properly represent the target domain requires constructs that are not supported by the target ontology language. As a result, it is possible that changes in the conceptual structure (such as the mentioned reification) are required because of the limitations of the target formalism.

[4]for a brief explanation of reification refer to Section 7.2.1

10.2.4 Properties of an intermediate model

Instead of specifying our mental model directly in the implementation formalism during the conceptualization activity, a so-called *intermediate model* should be developed. This may not be suitable for ontology reasoning but it can include any knowledge representation constructs which seem to be natural for the problem at hand. This model may be defined in any form which is understood and accepted by domain experts (e.g., Excel sheets, a mind map, semi-structured text, or combinations thereof).

This approach has many advantages. First, even if some of the constructs can be represented in the target formal language using a workaround (e.g., representing n-ary relations as concept instances), this makes the understanding of the model for humans much more complicated, and thus reduces the possibility of ontology reuse. On the other hand, if a natural representation of the ontology is available at the conceptual level, people do not have to "parse" the low-level formal representation.

Second, if some information cannot be represented in the target formalism at all, we may still have it in the conceptual model. If we later switch to a more powerful formalism, we can readily reuse this information.

Finally, because the conceptual model uses constructs that naturally describe the domain of discourse, it is easier to communicate it toward the domain experts. It is sometimes even possible that domain experts can edit parts of the model, without active participation of the ontology engineer. This helps to solve the common dilemma: who should develop an ontology? Domain experts who understand the domain but have only a little or absolutely no idea of good ontology design; or the ontology engineer who has extensive knowledge about proper ontology constructs but only a limited knowledge of the domain of discourse? By using intermediate models, the domain experts can edit those representations, and the ontology engineer can devise the best ways to map those (possibly) non-computable conceptual representations into formal ontology constructs. In most cases, it is even possible to find ways to generate parts of the formal ontology out of parts of the conceptual model fully automatically.

Although the intermediate model is not necessary suitable for ontology reasoning, it should be machine processable. Only in this case it is possible to automatize the generation of the formal model out of the conceptual model, at least partially.

The value of intermediate models was already proved in many real-world systems (such as Galen [RWRR01], or in the OTK use cases [SS02]). It is also interesting to note that the idea of separating the conceptual model from the implementation (i.e., the concrete ontology formalism) is very similar to the vision of *Model Driven Architecture* (MDA), a movement which became popular recently in the software engineering field[5].

10.2.5 Intermediate models in VICODI

To solve the presented problems, we also started to use intermediate models in VICODI [Nag04]. Although this approach requires more advanced tool support for transforming in-

[5]See `http://www.omg.org/mda/` for details

termediate models into the target ontology formalism, the problems described above gave us enough motivation to try this path. In the intermediate model it was possible to

1. encode knowledge in a way that was familiar to the domain experts and
2. specify information that was not included into the final formal ontology but was considered useful by the domain experts.

This approach was a great success and significantly sped up knowledge acquisition in the project. However, creating the necessary tool support remained a non-trivial issue.

10.2.6 Tool support

The conceptualization phase of ontology development basically consists of the following major tasks: creating the concept and property structure of the ontology, populating the ontology with instances and relation instances that connect them, and specifying attributes of these instances.

Deficiencies of graphical ontology editors

For editing the concepts and properties of an ontology, there are several graphical ontology editors, such as KAON OIModeler[6], Protege[7] or SWOOP[8]. As we have seen, however, an ontology supporting IR should also contain lots of instances, relation instances, and attributes.

Collecting new instances, relation instances and attribute values can be a very cumbersome and time consuming work for domain experts. Ontology learning tools that build on shallow natural language processing (NLP) promise support for this process. A typical example for such a tool is Text2Onto [CV05], which is based on the GATE framework[9]. Unfortunately, based on our experiences during and after the VICODI project, the results of these systems are not yet adequate to serve as a direct input for an ontology. Therefore, ontology development and ontology population still remain predominantly manual, human resource intensive tasks.

Existing graphical ontology editors have various problems with ontologies containing lots of instances. First, they do not scale well, i.e., they are simply too slow on big ontologies. This is rather an implementation than a conceptual problem but a very important obstacle during real-world ontology editing. Second, their GUI does not support browsing and editing a large number of entities. E.g., most editors simply list all instances of a concept, which is clearly useless if the concept has more than a dozen instances[10].

As another example, at the time of writing of this thesis it was impossible to select more than one entity in the Protege editor, and run an operation on all of the selected entities. If several

[6] `http://kaon.semanticweb.org/`
[7] `http://protege.stanford.edu/`
[8] `http://www.mindswap.org/2004/SWOOP/`
[9] GATE is also used by KIM [PKO+04], or by my metadata generation system
[10] E.g., in the VICODI ontology the PERSON concept had more than 5000 indirect instances.

hundred instances should be moved between two concepts[11], this limitation clearly renders the editor unusable.

Finally, most of the editors operate on the formalization or the implementation level. I.e., they are tied to specific ontology formalisms (such as KAON to the KAON OIModel, and SWOOP to OWL), or to specific ontology modeling paradigms (such as Protege to frame-based ontologies, or recently to OWL). In other words, they are not flexible enough to support the intermediate models approach, which was used in VICODI.

VICODI's hybrid approach for ontology editing

The solution to this problem, which was first developed during the VICODI project [Nag04], and was later refined during my thesis work, is the following. We used a hybrid approach for ontology editing: visual editors for concepts and properties, and legacy spreadsheet editors for instance editing. Therefore, the scalability problem that was mentioned above, does not occur. Moreover, most of the intermediate model features, which are needed for concepts and properties, can be somehow modeled in visual editors.

During VICODI we used the KAON OIModeler editor for concept and property editing because it provided some nice features in comparison to other ontology editors. These features include a nice graph view of the concept and property structure, and the possibility to apply operations on many selected entities[12].

The KAON OIModel ontology formalism[13] does not have the notion of "timed properties" that were used in VICODI. However, the KAON OIModel supports metamodeling, i.e., ontology entities can be concepts, properties or instances at the same time. The instance interpretation of a concept or property is called its *spanning instance*. It was easy to exploit this feature to model timed properties. We simply defined a special TEMPORALPROPERTY concept, and declared the spanning instances of properties that were timed as instances of this special concept (see Figure 10.3 which shows an excerpt from the ontology for the HASPART timed property). That way, the KAON OIModel, as an intermediate model, contained all of the information that was needed to automatically transform it to the ontology formalism, which was described in Section 7.2.2.

Sometimes introducing such "hacks" into the intermediate model is not possible because it would not be intuitive for domain experts. In our case, it was possible to extend the visual editor to support the needed feature, i.e., to declare a property timed by setting a checkbox, instead of manually declaring the concept as an instance of the TEMPORALPROPERTY concept. This made this solution transparent to the domain experts.

Normally, ontology editors provide a plugin API, which allows to add specific extensions and features relatively easily. It is clear that it is best to choose a visual editor, whose native supported formalism is the closest to the one that is needed by the intermediate model, and therefore the fewest extensions are needed.

[11]This happens quite often, especially when introducing new, more specific subconcepts to the concept taxonomy.
[12]In contrast to the Protege editor, as was described above.
[13]a slight extension of RDFS, for more details see [MMV02]

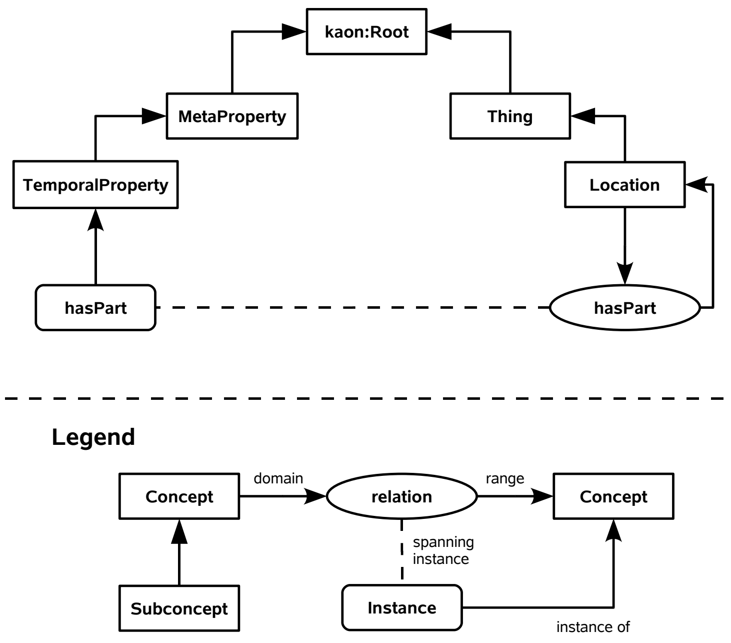


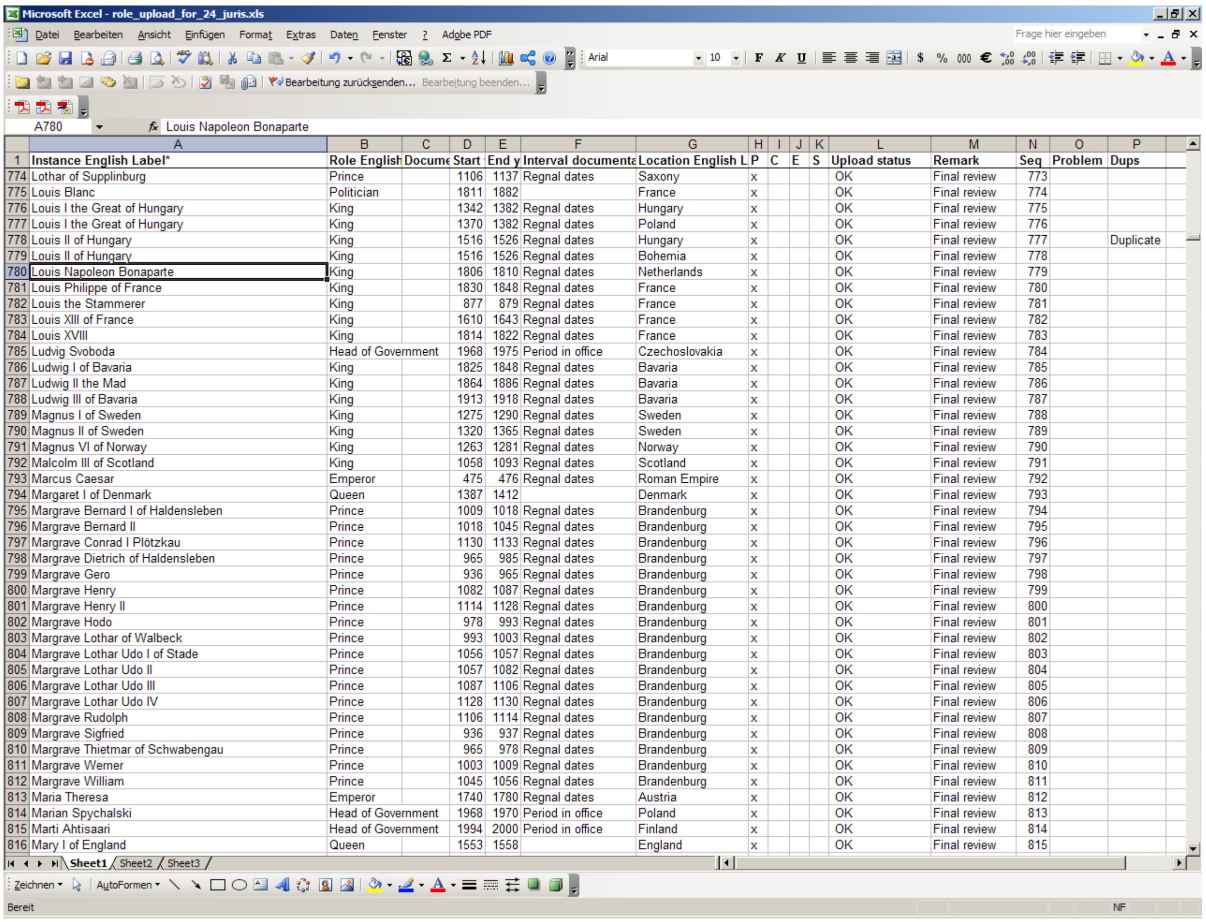
Figure 10.3: Timed properties in a KAON OIModel

Spreadsheet programs for instance editing

Spreadsheet programs, such as MS Excel or OpenOffice Calc are ideal for editing huge number of entities. They are therefore also suitable for editing lots of ontology instances and relation instances. Using the fast and powerful search/replace and copy/paste features, it is relatively easy to make even extensive changes in the instance base — changes that would not be possible using visual editors, such as Protege. The easiest way to exploit these programs is to define a standard structure based on the intermediate model. E.g., in my case I had the following spreadsheet tables during my thesis work: Instances, Relations, TemporalRelations, TemporalAttributes and Lexical.

Using spreadsheets as intermediate models for the instance level, it is also very easy to define intermediate models which provide specialized constructs for cases that are common in the target domain. E.g., during the VICODI project, we had a special table in the sheet, which was customized for defining person roles. In a person role, a person was connected with a role specification (such as EMPEROR); with a location where the role was played, and with a time specification during which the role was played. In a "traditional", generic ontology formalism it would be quite cumbersome to express such relationships. In our case, it was possible to simply enter the needed information to one row of an Excel sheet ((see Figure 10.4).

This flexibility of defining customized sheets for complicated modeling constructs is perhaps the biggest merit of the spreadsheet approach: it is much easier to define new tables, or new columns in a spreadsheet, than to extend a visual editor when the required modeling idioms are missing from the intermediate model.



Microsoft Excel - role_upload_for_24_juris.xls

A780 Louis Napoleon Bonaparte

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Instance English Label*	Role English	Docume	Start	End y	Interval documenta	Location English L	P	C	E	S	Upload status	Remark	Seq	Problem	Dups
774	Lothar of Supplinburg	Prince		1106	1137	Regnal dates	Saxony	x				OK	Final review	773		
775	Louis Blanc	Politician		1811	1882		France	x				OK	Final review	774		
776	Louis I the Great of Hungary	King		1342	1382	Regnal dates	Hungary	x				OK	Final review	775		
777	Louis I the Great of Hungary	King		1370	1382	Regnal dates	Poland	x				OK	Final review	776		
778	Louis II of Hungary	King		1516	1526	Regnal dates	Hungary	x				OK	Final review	777		Duplicate
779	Louis II of Hungary	King		1516	1526	Regnal dates	Bohemia	x				OK	Final review	778		
780	Louis Napoleon Bonaparte	King		1806	1810	Regnal dates	Netherlands	x				OK	Final review	779		
781	Louis Philippe of France	King		1830	1848	Regnal dates	France	x				OK	Final review	780		
782	Louis the Stammerer	King		877	879	Regnal dates	France	x				OK	Final review	781		
783	Louis XIII of France	King		1610	1643	Regnal dates	France	x				OK	Final review	782		
784	Louis XVIII	King		1814	1822	Regnal dates	France	x				OK	Final review	783		
785	Ludvig Svoboda	Head of Government		1968	1975	Period in office	Czechoslovakia	x				OK	Final review	784		
786	Ludwig I of Bavaria	King		1825	1848	Regnal dates	Bavaria	x				OK	Final review	785		
787	Ludwig II the Mad	King		1864	1886	Regnal dates	Bavaria	x				OK	Final review	786		
788	Ludwig III of Bavaria	King		1913	1918	Regnal dates	Bavaria	x				OK	Final review	787		
789	Magnus I of Sweden	King		1275	1290	Regnal dates	Sweden	x				OK	Final review	788		
790	Magnus II of Sweden	King		1320	1365	Regnal dates	Sweden	x				OK	Final review	789		
791	Magnus VI of Norway	King		1263	1281	Regnal dates	Norway	x				OK	Final review	790		
792	Malcolm III of Scotland	King		1058	1093	Regnal dates	Scotland	x				OK	Final review	791		
793	Marcus Caesar	Emperor		475	476	Regnal dates	Roman Empire	x				OK	Final review	792		
794	Margaret I of Denmark	Queen		1387	1412		Denmark	x				OK	Final review	793		
795	Margrave Bernard I of Haldensleben	Prince		1009	1018	Regnal dates	Brandenburg	x				OK	Final review	794		
796	Margrave Bernard II	Prince		1018	1045	Regnal dates	Brandenburg	x				OK	Final review	795		
797	Margrave Conrad I Plötzkau	Prince		1130	1133	Regnal dates	Brandenburg	x				OK	Final review	796		
798	Margrave Dietrich of Haldensleben	Prince		965	985	Regnal dates	Brandenburg	x				OK	Final review	797		
799	Margrave Gero	Prince		936	965	Regnal dates	Brandenburg	x				OK	Final review	798		
800	Margrave Henry	Prince		1082	1087	Regnal dates	Brandenburg	x				OK	Final review	799		
801	Margrave Henry II	Prince		1114	1128	Regnal dates	Brandenburg	x				OK	Final review	800		
802	Margrave Hodo	Prince		978	993	Regnal dates	Brandenburg	x				OK	Final review	801		
803	Margrave Lothar of Walbeck	Prince		993	1003	Regnal dates	Brandenburg	x				OK	Final review	802		
804	Margrave Lothar Udo I of Stade	Prince		1056	1057	Regnal dates	Brandenburg	x				OK	Final review	803		
805	Margrave Lothar Udo II	Prince		1057	1082	Regnal dates	Brandenburg	x				OK	Final review	804		
806	Margrave Lothar Udo III	Prince		1087	1106	Regnal dates	Brandenburg	x				OK	Final review	805		
807	Margrave Lothar Udo IV	Prince		1128	1130	Regnal dates	Brandenburg	x				OK	Final review	806		
808	Margrave Rudolph	Prince		1106	1114	Regnal dates	Brandenburg	x				OK	Final review	807		
809	Margrave Sigfried	Prince		936	937	Regnal dates	Brandenburg	x				OK	Final review	808		
810	Margrave Thietmar of Schwabengau	Prince		965	978	Regnal dates	Brandenburg	x				OK	Final review	809		
811	Margrave Werner	Prince		1003	1009	Regnal dates	Brandenburg	x				OK	Final review	810		
812	Margrave William	Prince		1045	1056	Regnal dates	Brandenburg	x				OK	Final review	811		
813	Maria Theresa	Emperor		1740	1780	Regnal dates	Austria	x				OK	Final review	812		
814	Marian Spychalski	Head of Government		1968	1970	Period in office	Poland	x				OK	Final review	813		
815	Marti Ahtisaari	Head of Government		1994	2000	Period in office	Finland	x				OK	Final review	814		
816	Mary I of England	Queen		1553	1558		England	x				OK	Final review	815		

Figure 10.4: Role specifications in the VICODI spreadsheet

The drawback of the approach is that only minimal sanity checking[14] can be done during instance editing time[15]. E.g., it is possible to refer to non-existing instances and concepts, or violate domain and range constraints in the ontology. Those modeling errors would not be possible in a visual ontology editor which does not allow the user to refer to non-existing entities, and can check simple ontological constraints, such as property domain and range. In the "spreadsheet approach", sanity checking is deferred until "upload time" when the intermediate model is transformed into the final ontology formalism.

Transformation of the intermediate models

At the very end, the two parts of the intermediate model — the one defining the concepts and properties, and the other defining the instances and their relations or attributes — are transformed into the implementation ontology formalism, which can be used for reasoning. This step must happen fully automatically, of course. Technically it is not a problem because

[14]The definition of sanity checking according to the The New Hacker's Dictionary: "The act of checking a piece of code (or anything else, e.g., a Usenet posting) for completely stupid mistakes".

[15]Although some checks are possible by using macros implemented in the scripting language of the spreadsheet program, such as Visual Basic.

there are APIs to access both the ontology formalisms supported by visual editors and the various spreadsheet formats. E.g., during VICODI, and also later during my thesis work I used the KAON API to access the KAON OIModel, and the “Java Excel API”[16] to access the Excel spreadsheets.

The transformation was not a problem conceptually, either, because the two parts of the intermediate model defined all the information that was needed by the target formalism[17].

During VICODI, the KAON OIModel also served as the implementation formalism. During my thesis work, I used the implementation formalism that was discussed in Chapter 7.

10.3 The VICODI methodology for building IR ontologies

Based on our previously described experiences during the VICODI project we were able to construct an ontology building methodology that was customized for the development of IR ontologies. This methodology, termed the VICODI methodology, was successfully used during the VICODI project to build the ontology of European history. The same methodology was then used in this thesis work to build the ontology for evaluating my ontology-based IR system.

The methodology follows the *bottom-up strategy* for conceptualization, with an initial brainstorming activity, which is characteristic for the beginning of the *middle-out strategy*. It is inspired and based on METHONTOLOGY, therefore I will discuss here only the differences between the two methodologies grouped by the specific ontology development activities.

10.3.1 Specification

The specification in the VICODI methodology is informal, and document collection oriented. It consists mainly of identifying documents in an existing document collection, which will serve as the basis of the new ontology[18]. The specification also includes an initial set of the metadata generation rules, which should be supported by the ontology.

Knowing the knowledge sources and the rules guides the process of ontology creation very effectively but also ties the ontology closely to the document collection. The goal of the ontology is to support the metadata generation process on the one hand, and effective information retrieval on the other hand. These tasks are clearly dependent on the document collection. Moreover, it is not the goal of the ontology to provide a complex, deep and full axiomatization of any application domain that would be independent of any actual document collection. Therefore, the strong relation between the ontology and the document collection is not a drawback in this application scenario.

[16] `http://www.jexcelapi.org`
[17] The spreadsheet part even defined elements that were not transformed into the implementation formalism, such as information on the imperfection of time specifications.
[18] or of a new extension to an existing ontology

10.3.2 Conceptualization and knowledge acquisition

The conceptualization process is *instance- and rule-oriented.* Instance-oriented means that the process starts first by collecting promising instances from the documents, and the concept and property structure is refined only later, when most of the new instances are already known. Rule-oriented means that the already identified heuristic rules guide the process of creating new concepts, relations and attributes for the ontology. In line with the goals of the ontology development stated in Section 10.2.1, only such changes are made in the concept and property structure of the ontology that have the potential to increase metadata generation and/or information retrieval effectiveness. This all defines a bottom-up strategy for conceptualization.

It is usually helpful to conduct a short brain-storming at the beginning of the conceptualization activity, to identify the most important concepts of the domain. A list of the most important concepts can ease the task of finding new instances in the documents. Therefore, such a brain-storming phase is also encouraged by the VICODI methodology. Although the brain-storming activity is normally characteristic for a middle-out strategy, it does not mean that the VICODI methodology follows this strategy. On the contrary, after the initial brain-storming activity, the VICODI methodology follows the bottom-up strategy to strictly scope the conceptualization and knowledge acquisition processes.

Another distinguishing feature of the conceptualization activity in this methodology is the heavy use of intermediate models. As was explained, we had very good experiences with intermediate models during the VICODI project. We could clearly see that it helped domain experts to participate in the ontology building process because it was possible to fine-tune the modeling constructs to the application domain. It was also easier to add additional documentation about design decisions, or about information which could not been represented in the target ontology formalism. E.g., in the VICODI model only classical time intervals were used, and the temporal imperfection issues were added only as informal comments. Finally, intermediate models can be edited much more comfortably using well-known tools than ontologies with existing visual ontology editors.

10.3.3 Formalization and Implementation

The formalization and implementation steps are merged, and the intermediate models are transformed directly to the ontology formalism described in Chapter 7.

10.3.4 Evaluation and ontology adjustment

The evaluation metric of the ontology is the quality of the generated semantic metadata and the quality of information retrieval results. Therefore, the methodology proposes a *technology-focused evaluation* of the ontology, where information retrieval and metadata generation results are analyzed, and potential deficiencies are identified.

In addition, the evaluation can also be made user-centric, by concentrating on queries that most user execute, and on documents that most users examine. As the result of the evaluation, a list of documents may be identified that are not covered properly by the ontology. Although it is not strictly part of the ontology evaluation, during this phase also a list of new ontology-based rules can be identified, that can be applied to improve semantic metadata quality. At the same time, rules that cannot be exploited during metadata generation (i.e., never executed), can be removed.

If deficiencies (missing ontology elements) were identified during evaluation, the ontology has to be adjusted. This means that the conceptualization activity should be started again. In other words, the VICODI methodology defines a cyclic process, similarly to many other methodologies, including METHONTOLOGY and OTK.

10.4 Summary

In this chapter, I first reviewed the state of the art of ontology development. Later, I reported on our experiences during the VICODI project where a large ontology of European history was developed for an ontology-based information portal. These experiences showed that (1) ontologies for IR are different from other types of ontologies, (2) existing methodologies are not suitable for developing this kind of ontologies without change and (3) there was no adequate tool support for the development of such ontologies.

Based on these experiences, a new methodology for the development of IR ontologies emerged during the project. This methodology is termed the VICODI methodology, and it was described in the remaining part of the chapter. The VICODI methodology defines an instance and rule oriented methodology which follows the bottom-up strategy for conceptualization. An important characteristic of the methodology is the use of intermediate models during the conceptualization step that allows for a greater involvement of domain experts in the ontology development process than is possible in the usual case.

Chapter 11

User interface and implementation

The ideas and algorithms previously presented in Chapters 6 to 10 were implemented in the IRCON prototype. The IRCON prototype is a classical three-tier web application from a technological point of view. Its user interface allows non-expert users to interact with the system and explore the capabilities of an ontology-supported information system.

This chapter first describes the IRCON user interface which extensively exploits the ontology and the generated semantic metadata to make the search process more user friendly. Later, some aspects of the implementation are addressed that realized the flexible architecture of IRCON. Next, the system is described from the point of view of the system administrator. Finally, a discussion of some performance issues with ontology reasoning that were discovered and solved during the course of implementation concludes this chapter.

11.1 User interface and user workflow

In this section, the user interface of the IRCON prototype is reviewed and the workflow of the end-user is described.

11.1.1 Requirements for the IRCON user interface

The goal of the IRCON user interface is to provide a possibility for end users to experiment with ontology-based IR. I expect that the typical end-user is not an IR-expert but has some experience with traditional Internet search engines, such as Google. This defines the following requirements towards the user interface:

- It should be possible to specify a query in natural language by simply typing it in a text field because users are accustomed to that possibility and they expect it. See also the requirement NATURAL LANGUAGE QUERY.
- The user interface should visualize semantic metadata in some form. This feature allows users to further navigate in the repository without manually submitting new queries. This is especially important for users who are not IR experts. Moreover, by explicitly displaying semantic metadata we stress the fact that the system can do something more than a traditional full-text IR system. I.e., we encourage users to explore the semantic possibilities provided by the system.

- Browsing of the ontology should be supported. This is a trivial feature in any ontology-based information system. Developing ontologies costs time and money, and the resulting artifact is interesting per se. By browsing the ontology, users can explore non-trivial relations among information entities. Moreover, they can also get some impression about which topics are addressed by the document collection.

- Ontologies also provide the possibility for semantic disambiguation. Users can exactly specify which "John Smith" or "George Bush" they exactly mean. In doing so, they open the possibility for the system to provide better search results that better fit their information need. This semantic disambiguation should be easy and intuitive for the end user.

11.1.2 Related work

In this subsection, I briefly review the works that inspired the user interface of the IRCON prototype. This list of solutions represents by no means a comprehensive list of efforts in the area of ontology-based user interfaces, as this area was not the main focus of this thesis.

Most ontology-based information systems, including the ones that were reviewed in Chapter 4, allow users to browse the ontology. This can be very helpful to formulate new queries to the system, or just simply to explore what kind of topics are covered in the document repository. The IRCON prototype also provides this feature.

Apart from that trivial way of exploiting an ontology, there are also other possibilities to make use of the knowledge stored in ontologies in the user interface. These possibilities make use of the fact that information retrieval is rather a process than an independent list of query–answer interactions with the system, as was explained in Section 2.1. Clearly, the queries in this process are semantically related and usually a query is a simple semantic transformation of the previous one. Some examples of these transformations are: including synonyms of existing search terms into the query, excluding synonyms, adding new, semantically related search terms etc. [Bat90]. This semantic transformation requires complex cognitive processing and lots of experience from the end user. Therefore, especially beginners often have problems to formulate good queries based on the results of previous ones. This query-reformulation process can be effectively supported using the background knowledge stored in ontologies or using the semantic metadata that is associated to documents.

The system of Stojanovic [Sto05] automatically proposes new queries based on the previous query, using a simple linguistic ontology. Users just have to click on one of the proposals to start a new query. I.e., the system completely eliminates the need to reformulate the query because users only have to decide which reformulation is the closest to their information needs, which is an easier task. The solution was successfully evaluated in a small-scale evaluation where PhD students found the suggestions of the system relevant to their information need.

This approach clearly shows that advantages of automatic query suggestions to guide the unexperienced user in the query reformulation process, and motivated the need for such a feature in the IRCON prototype. However, the presented solution of directly generating suggestions based on user queries was overly complex for my purposes, and therefore was not directly exploited. Instead, IRCON uses a simpler method, based on the metadata of the currently viewed document, to provide navigation hints for the users.

The Semantic Search system, proposed by Guha and his colleagues [GMM03] uses semantic metadata of documents and ontological knowledge to provide background information of the query on the one hand, and to improve the results of full-text search on the other hand. To achieve these goals, the denotation of the query has to be clarified first. E.g., if the user searches for “John Smith”, the proper John Smith has to be selected first from the ontology. They propose a manual approach for that, where the possible denotations of query terms are proposed to the user who can simply select the right meaning of the query term. To start, the system always picks one of the denotations automatically, so in many cases no selection is needed by the user. In some cases, such as in the example of John Smith above, there are simply too many possible denotations of a term. In such cases, the system provides a “this denotation” link for the search results. I.e., in a query about “John Smith”, there will be a link “this John Smith” to each query result. That way, the user still gets the opportunity to select the right denotation, even if there are many hundreds or thousands of possibilities.

After the right denotation of a term has been selected, the information is used in the Semantic Search system in two ways. First, the relevant portions of the ontology are shown to the user together with the query results. E.g., the user sees the profession, the birth date, working place, living place etc. of his or her selected John Smith. This allows the user to navigate to semantically related entities (i.e., to semantically transform the query) simply by clicking on an entity displayed. E.g., the user could click on the workplace of John Smith to execute a new query about that entity.

Further, knowing the right denotation of a search term can be used to improve the results of full-text search by exploiting background information in the ontology. E.g., the query can be extended by the email address of the person, or if the person is a musician, music related documents that contain the search term can get a higher ranking score. Unfortunately, the authors of the paper present only some vague ideas about this approach, and no concrete details.

The Semantic Search system introduces many interesting ideas that inspired the IRCON GUI. First, the feature to semantically disambiguate query terms just by a simple click was also implemented in IRCON, albeit in a different (simpler) manner. Second, the idea of showing semantically relevant information to the user to facilitate navigation (i.e., automatic query transformation) was also implemented in IRCON. In contrast to Semantic Search, however, IRCON uses the metadata of the document to display related entities of the ontology for a specific information need.

The VICODI system [NDO05] that was already briefly introduced in Section 3.1 visualizes the semantic metadata of the documents. In VICODI, all document terms that were linked with a metadata element are highlighted. Metadata elements that do not have linked terms in the document text (i.e., indirectly relevant concepts) are listed below the document text. Finally, metadata elements that represent locations are visualized on a historical map that is selected based on the temporal part of the metadata. E.g., in a document about the Battle of Trafalgar, United Kingdom, France and Spain are highlighted on a historical map of the first decade of the 19. century[1] (see Figure 3.1).

[1]VICODI contained a map for each decade. The Battle of Trafalgar happened in 1805.

Similarly to the other discussed systems, clicking on the highlighted elements initiates a new semantic query. In addition, VICODI implemented the idea to initiate the semantic query "in context" by considering all of the metadata elements of the actual document, and not only the element that was actually clicked on. Of course, the element that was clicked on gets the most weight in the query. E.g., if the user clicks on France in the Battle of Trafalgar document, she or he gets documents about France in the context of Battle of Trafalgar, and not simply documents generally about France.

The VICODI system strongly inspired the IRCON GUI. IRCON basically implements the same ideas as VICODI on the user interface, with some differences:

- IRCON strictly separates the metadata elements from the document content. I.e., metadata elements are not highlighted in the document text, even if the textual representation of a metadata element appears in the text. This has the advantage that indirectly and directly relevant entities can be treated in a uniform manner, which allows a consequent visualization of all metadata elements.
- IRCON provides some advanced metadata visualization features, such as the explicit visualization of the importance of the metadata element and that metadata elements are ordered based on their relevance.
- IRCON provides the possibility to directly browse the definition of a metadata element in the ontology, and also to initiate a contextual search based on that element just by one click. In contrast, VICODI allowed only the initiation of new queries, which made the access to the ontology complicated.
- IRCON allows users to select more than one element as relevant or irrelevant for the next contextual query. This provides more flexibility than VICODI.
- Finally, IRCON does not implement the visualization of maps that was a major feature of VICODI. Although this is a very nice and user-friendly feature, it is very challenging to implement it, and it was not considered crucial for the IRCON prototype. It is, however, a possible future extension of the IRCON system.

11.1.3 The end-user workflow

The end-user interacts with the IRCON system following the workflow that is shown in Figure 11.1. The user has two options to start a new query. First, she or he can specify a textual query on the home page of the application, similar to Google (see Figure 11.2). Experienced users can also directly build a semantic query by specifying the elements of the conceptual, textual and temporal parts one-by-one (see Figure 11.3). They can browse the ontology to pick elements of the conceptual part, and use the tool for constructing fuzzy temporal intervals (see Section 6.6.3) for the temporal part.

Our experience in VICODI showed that domain experts and end-users cannot reliably determine the exact weight of a metadata or query element. I.e., they cannot really say whether the exact weight of an element is 0.56 or 0.62. Therefore, instead of requiring users to specify the exact weight of a query element, they can chose the weight from a weight scale from zero

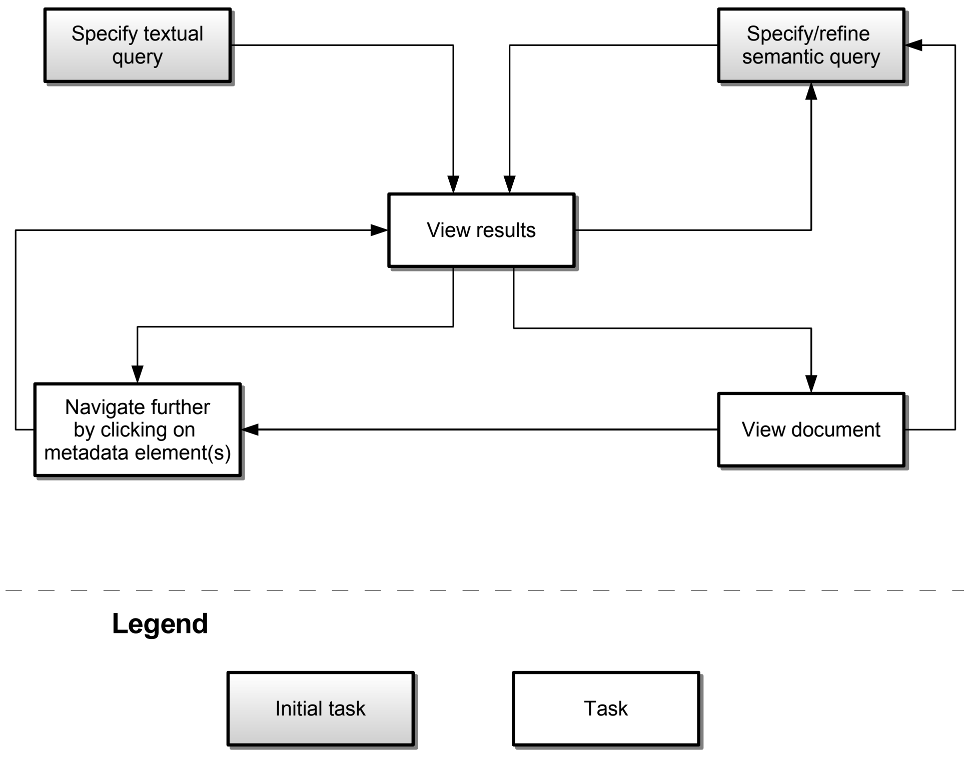

Specify textual query
Specify/refine semantic query
View results
Navigate further by clicking on metadata element(s)
View document
Legend
Initial task
Task

Figure 11.1: The workflow of the end-user search and navigation process

to five[2]. These weights are then mapped to the weights 0.0, 0.2 etc., respectively. A similar mapping is executed for display purposes in the opposite direction. E.g., all values between 0.8 and 1.0 are mapped to the display weight five. This is needed because many of the weights are generated by the automatic semantic metadata generation algorithm which naturally produces weights such as 0.56 or 0.62.

Semantic queries can be executed directly by the system. Textual queries are first automatically parsed into a semantic query which is then processed identically to a manually defined semantic query.

The results of a semantic query are shown together with the query itself (see Figure 11.4). This is especially important when the user specified the query as natural language text because the results of query parsing can be controlled, and semantic disambiguation can be done when necessary. In addition, this solution also allows some simple forms of query reformulation.

The query is displayed on the left side of the GUI. Elements of the query are ordered by their weights. Elements of the conceptual part are linked with the ontology, i.e., when the user clicks on a WOI, she or he can browse the ontology beginning with the OI. Clicking on the intervals results in a string that can be directly imported and thus displayed in the fuzzy interval construction tool. Elements of the textual part are displayed as simple text.

Each metadata element contains a search icon (see Figure 11.5 that shows the visualization of the #GEORGE_W._BUSH metadata element). By clicking on this icon, the user can initiate a new semantic search, where the entity that was clicked on is emphasized (has the weight 1.0),

[2]from one to five for a query

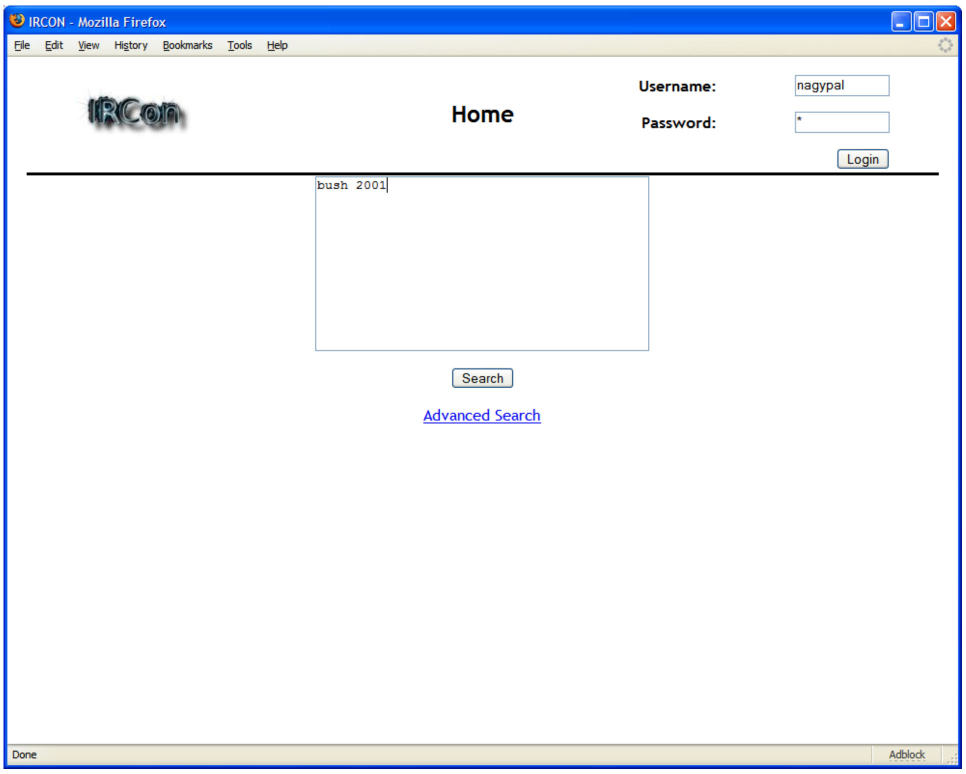


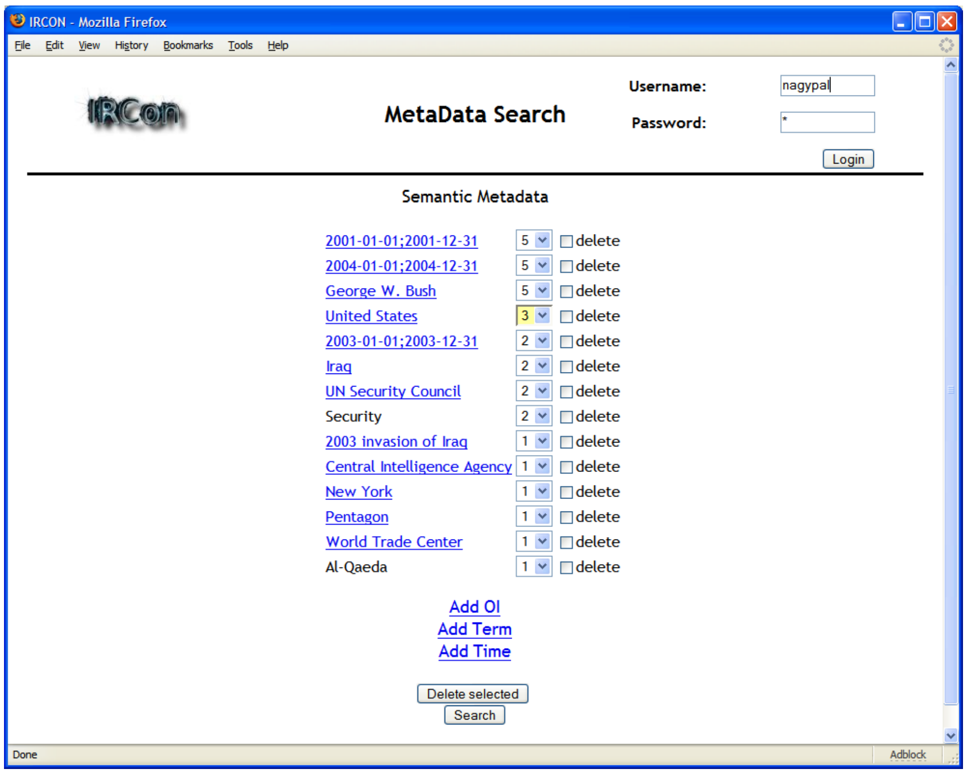
Figure 11.2: The start page of the IRCON prototype

all of the other metadata elements are weakened, and the elements that are below a threshold are removed from the query. This practically implements the "contextual search" idea of the VICODI system. In addition, the user can also select and deselect items of the query, and can initiate a new search. Elements that were selected get the maximal weight, elements that were deselected are removed from the query. All other elements are weakened as described above.

This mechanism practically allows the user to disambiguate the query simply by clicking on the right alternative(s) and optionally deselecting the wrong alternative(s). E.g., when the user specified "Bush" as the textual query, the system will create a semantic query containing both the OIs #GEORGE_H._W._BUSH and #GEORGE_W._BUSH. The user can easily emphasize the right Bush simply by clicking on it. This method can be viewed as an alternative implementation of the first idea for semantic disambiguation that was described for the Semantic Search system. In addition, the user can also emphasize elements for other purposes, e.g., when she or he gets especially interested in some parts of the query based on the result list.

Expert users additionally have the option to manually refine the semantic query on the semantic query editing page, which is the same page that is used for defining new semantic queries. In this page, they can freely add and remove elements to any parts of the query, and they can freely change the weights of existing elements.

Instead of initiating a new query using one of the described options, the user can also select a document from the result list. In this case, the document is displayed together with its semantic metadata (see Figure 11.6). The semantic metadata of the document is visualized using the same method that was described before for the semantic query. This is not surprising because



MetaData Search

Username: nagypal

Password: *

Semantic Metadata

2001-01-01;2001-12-31	5	delete
2004-01-01;2004-12-31	5	delete
George W. Bush	5	delete
United States	3	delete
2003-01-01;2003-12-31	2	delete
Iraq	2	delete
UN Security Council	2	delete
Security	2	delete
2003 invasion of Iraq	1	delete
Central Intelligence Agency	1	delete
New York	1	delete
Pentagon	1	delete
World Trade Center	1	delete
Al-Qaeda	1	delete

Figure 11.3: Specifying the semantic query in IRCON

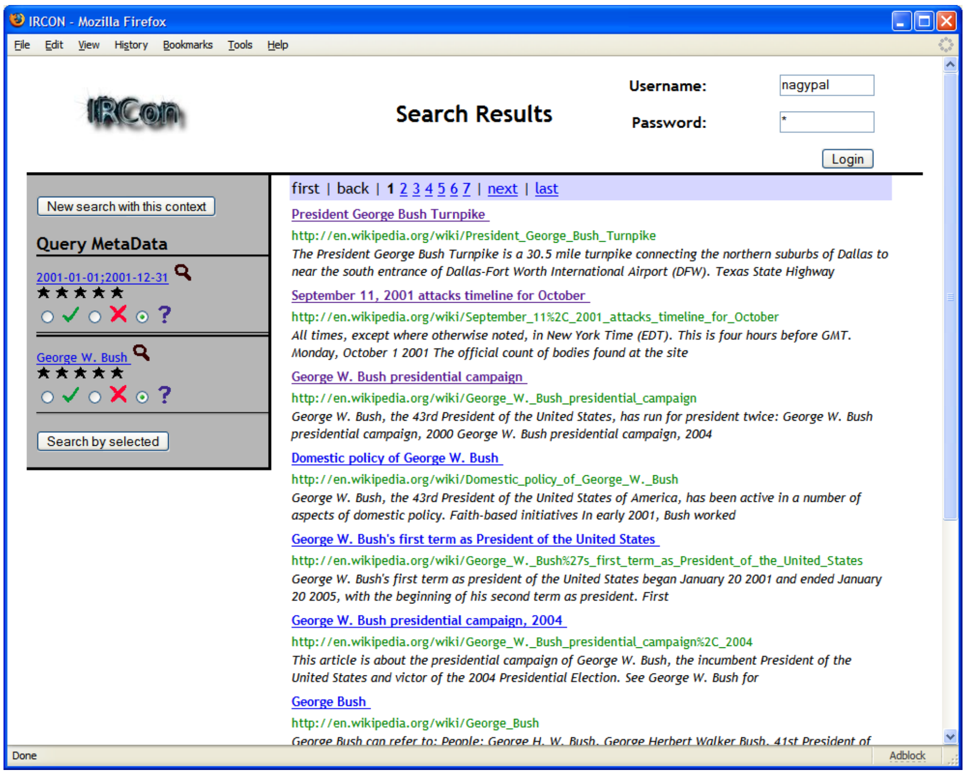
the semantic metadata and the query use the same information model. Consequently, the user has the same options to start a new query as she or he had on the result list page:

1. She or he can directly start a new query by emphasizing one element.
2. She or he can select and deselect some elements ans start a new query.
3. She or he can manually edit the semantic query starting from the actual document metadata.

To summarize the discussion about the search process, we can say that non-expert users have the option to control the IR process simply by clicking on various elements of the user interface, i.e., there is no need to actively reformulate any queries. From the user point of view, the IR process feels like browsing, the fact that in the background new semantic queries are constructed and executed is transparent to him or her. The only query that a non-expert user has to pose is a simple textual one that is common in current Internet search engines, and therefore familiar to the users. Expert users, however, have the freedom to fine-tune queries to get more precise results.

Besides supporting the IR process, the IRCON prototype also has the functionality to edit descriptive and semantic metadata. To use this functionality, users have to log in to the system. Descriptive metadata includes the title and abstract of a document.

Semantic metadata use the same information model as semantic queries and therefore the query editing page can also be used to edit metadata. There are two differences, however. First, users have the option to validate metadata elements. Because metadata elements are generated fully



Search Results

Username: nagypal

Password: *

first | back | 1 2 3 4 5 6 7 | next | last

President George Bush Turnpike
http://en.wikipedia.org/wiki/President_George_Bush_Turnpike
The President George Bush Turnpike is a 30.5 mile turnpike connecting the northern suburbs of Dallas to near the south entrance of Dallas-Fort Worth International Airport (DFW). Texas State Highway

September 11, 2001 attacks timeline for October
http://en.wikipedia.org/wiki/September_11%2C_2001_attacks_timeline_for_October
All times, except where otherwise noted, in New York Time (EDT). This is four hours before GMT. Monday, October 1 2001 The official count of bodies found at the site

George W. Bush presidential campaign
http://en.wikipedia.org/wiki/George_W._Bush_presidential_campaign
George W. Bush, the 43rd President of the United States, has run for president twice: George W. Bush presidential campaign, 2000 George W. Bush presidential campaign, 2004

Domestic policy of George W. Bush
http://en.wikipedia.org/wiki/Domestic_policy_of_George_W._Bush
George W. Bush, the 43rd President of the United States of America, has been active in a number of aspects of domestic policy. Faith-based initiatives In early 2001, Bush worked

George W. Bush's first term as President of the United States
http://en.wikipedia.org/wiki/George_W._Bush%27s_first_term_as_President_of_the_United_States
George W. Bush's first term as president of the United States began January 20 2001 and ended January 20 2005, with the beginning of his second term as president. First

George W. Bush presidential campaign, 2004
http://en.wikipedia.org/wiki/George_W._Bush_presidential_campaign%2C_2004
This article is about the presidential campaign of George W. Bush, the incumbent President of the United States and victor of the 2004 Presidential Election. See George W. Bush for

George Bush
http://en.wikipedia.org/wiki/George_Bush
George Bush can refer to: People: George H. W. Bush, George Herbert Walker Bush, 41st President of

Figure 11.4: Results list in IRCON

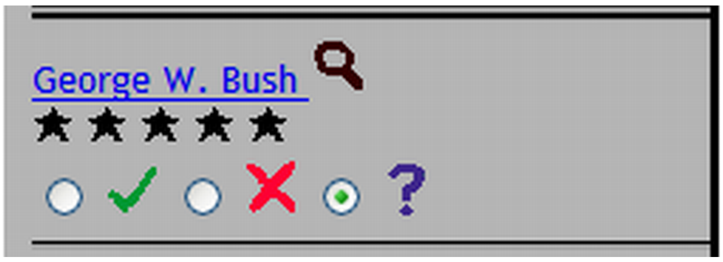
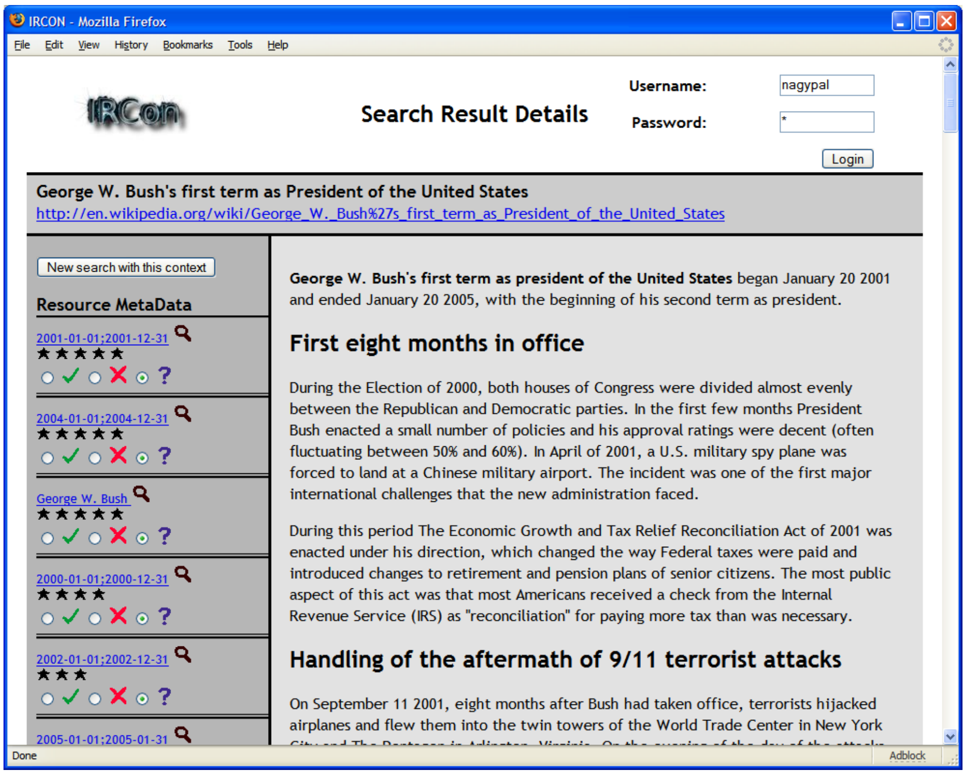


Figure 11.5: Visualization of one metadata element

automatically, it is a very valuable information whether human experts agree with the decisions of the algorithm. This information can be used to train and fine-tune heuristic rules. Second, in contrast to queries, also elements with the weight zero are retained. This is because it is a very valuable information that some elements that were considered relevant by the algorithm were later deleted by a domain expert. Users have the option, however, to hide elements with weight zero.

It is important to stress that in the current implementation this data is used only to guarantee that no manually validated metadata element gets overwritten by the results of the automatic metadata generation process. Because validated data is never deleted, it can positively affect the result of the metadata generation already in the current system because the heuristic rules work with high-quality, validated elements, and consequently they can make better decisions.


IRCON - Mozilla Firefox
File Edit View History Bookmarks Tools Help
IRCon
Login
New search with this context
Done
Adblock

Search Result Details

Username: nagypal

Password: *

George W. Bush's first term as President of the United States

http://en.wikipedia.org/wiki/George_W._Bush%27s_first_term_as_President_of_the_United_States

Resource MetaData

2001-01-01;2001-12-31 ★★★★★

2004-01-01;2004-12-31 ★★★★★

George W. Bush ★★★★★

2000-01-01;2000-12-31 ★★★★

2002-01-01;2002-12-31 ★★★

2005-01-01;2005-01-31

George W. Bush's first term as president of the United States began January 20 2001 and ended January 20 2005, with the beginning of his second term as president.

First eight months in office

During the Election of 2000, both houses of Congress were divided almost evenly between the Republican and Democratic parties. In the first few months President Bush enacted a small number of policies and his approval ratings were decent (often fluctuating between 50% and 60%). In April of 2001, a U.S. military spy plane was forced to land at a Chinese military airport. The incident was one of the first major international challenges that the new administration faced.

During this period The Economic Growth and Tax Relief Reconciliation Act of 2001 was enacted under his direction, which changed the way Federal taxes were paid and introduced changes to retirement and pension plans of senior citizens. The most public aspect of this act was that most Americans received a check from the Internal Revenue Service (IRS) as "reconciliation" for paying more tax than was necessary.

Handling of the aftermath of 9/11 terrorist attacks

On September 11 2001, eight months after Bush had taken office, terrorists hijacked airplanes and flew them into the twin towers of the World Trade Center in New York

Figure 11.6: Document view in IRCON

The information could be also used, however, to record the discrepancies between the results of the metadata generation process and the decisions of domain experts. This could provide a good basis to automatically fine-tune heuristic rules with using machine learning solutions. This possibility, however, has not been implemented in the current system.

In addition to the option to manually edit and validate metadata, users that are logged in have the possibility to edit the content of the documents, and to initiate the automatic (re-)generation of semantic metadata.

11.2 Flexible architecture

11.2.1 Architecture overview

The IRCON application is a classical three-tier web application, implemented in Java. The high-level infrastructure of the application together with the technologies and frameworks were already discussed in Chapter 5 and displayed in Figure 5.2. Figure 11.7 shows the high-level architecture again for convenience.

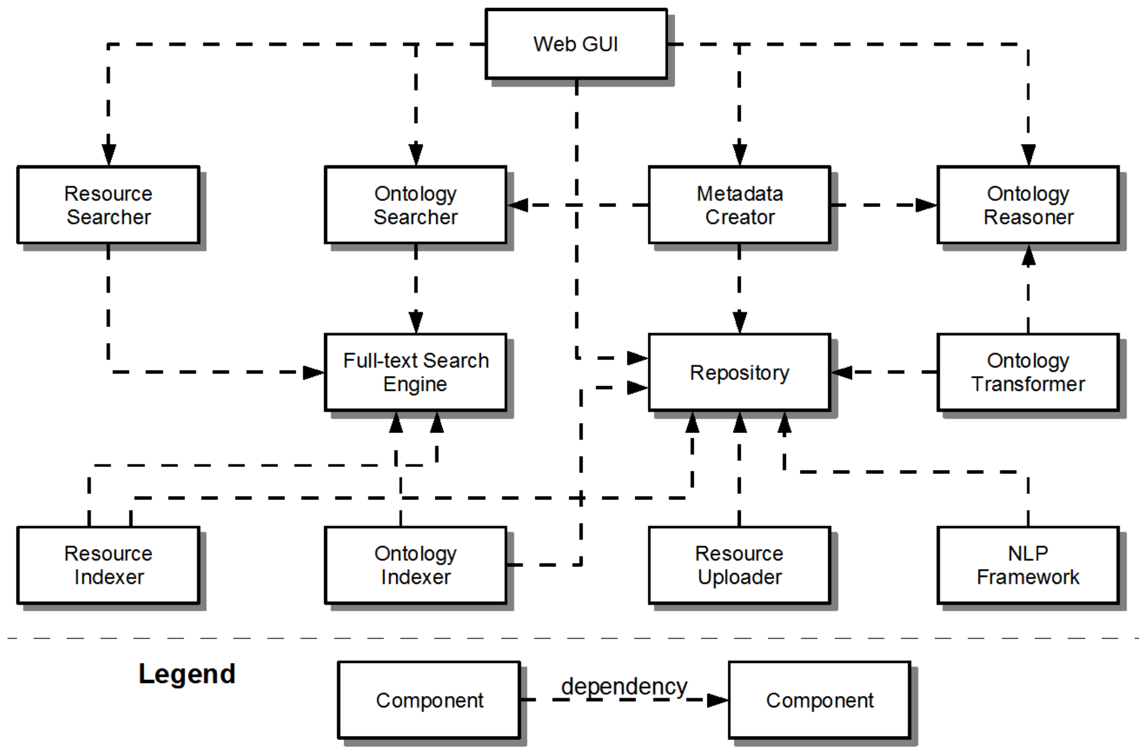


Figure 11.7: Architecture of IRCON (identical with Figure 5.2)

11.2.2 Requirements

The basic requirement for the implementation of this architecture was flexibility. To execute the evaluation of the system (which is described in the next chapter), different algorithms had to be tested and compared with each other. I.e., it should be easy to replace the implementations of the individual modules.

In addition, the IRCON application is very complex, integrating many different technologies. The functionality of a given module is dependent on many other modules that use different, complex technologies. E.g., the metadata creator module uses both the ontology reasoner, the ontology searcher (i.e., indirectly the full-text search engine), and also accesses the document repository (i.e., the relational database). All these dependencies make it difficult to develop and test such a module. A common solution for this problem is the use of so called mock objects or stubs [HT04] during testing that simulate the functionality of other modules a specific module depends on. To use this technique, it is again required that the implementation of the modules can be easily replaced by the dummy implementations during testing.

11.2.3 Inversion of control and the Spring framework

To implement the required flexibility, the "inversion of control" principle [Fow04] was used throughout the implementation. The basic idea of this principle is that dependencies of an object are injected into the object, and are not set or created by the object itself. That way, code dependencies just appear automatically in the object, and can be used without knowing how the dependencies were set. If the principle of inversion of control is combined with the rule that dependencies should be referenced only by their interfaces and never by their implementation classes, the basis for the required flexible architecture is set.

To implement the inversion of control principle, I used the Spring framework[3]. This framework implements an inversion of control container that can inject dependencies into objects based on external configuration, usually specified in XML configuration files. Spring also provides several other useful features that made the implementation of IRCON much easier. These features include the possibility to automatically start and commit database transactions on the invocation of specific methods of specific interfaces (a kind of aspect orientation) and several utility classes that support the JDBC-based interaction with databases.

11.2.4 Implementing the semantic search process

I demonstrate the presented ideas of inversion of control and Java interfaces on the example of the semantic search process implementation. Flexibility is achieved in other parts of the architecture similarly.

The semantic search process is an important part of the Resource Searcher module. It takes as input a semantic query (represented by the `MetaData` class), and returns a list of `ResourceSearchResult` objects that contain the necessary information for the GUI to represent the result (such as the URI and title of the resource, and its ranking score). As was described in Chapter 9, the search process needs to generate full-text queries against the content index and the metadata index, execute the queries, and combine the results using one of the combination algorithms that were described in Section 9.2.6. For evaluation purposes, various combination algorithms had to be experimented with. Moreover, during the course of the development, many possibilities had to be tested to find the optimal algorithm for the transformation of the semantic query to the content and metadata full-text queries. Finally, the parts of the search process are quite complex. Therefore, they were implemented and tested separately, using the testing principle of mock objects.

The UML class diagram of the search process implementation is shown in Figure 11.8. As can be seen, the search process itself is represented by the `SearchProcess` interface, so it can easily be plugged in to the Resource Searcher module. The interface is implemented by the `SimpleSearchProcess` class that references a list of activity layers. An activity layer consists of a list of activities, represented by the `Activity` interface. The `SimpleSearchProcess` simply executes each activity in each activity layer by invoking the `enact()` method of these activities.

The `Activity` interface has several subinterfaces. `MetaDataProviders` produce a `MetaData` object. These are needed by `QueryExecutors` that are also `ResourceResultCreators`, i.e., they create a list of `ResourceSearchResult` objects. `RankMergers` are also `ResourceResultCreators` and need other `ResourceResultCreators` as input. Finally, `MetaDataConsumers` can receive a `MetaData` object. This hierarchy of interfaces provides the flexibility to model many

[3]`http://www.springframework.org`

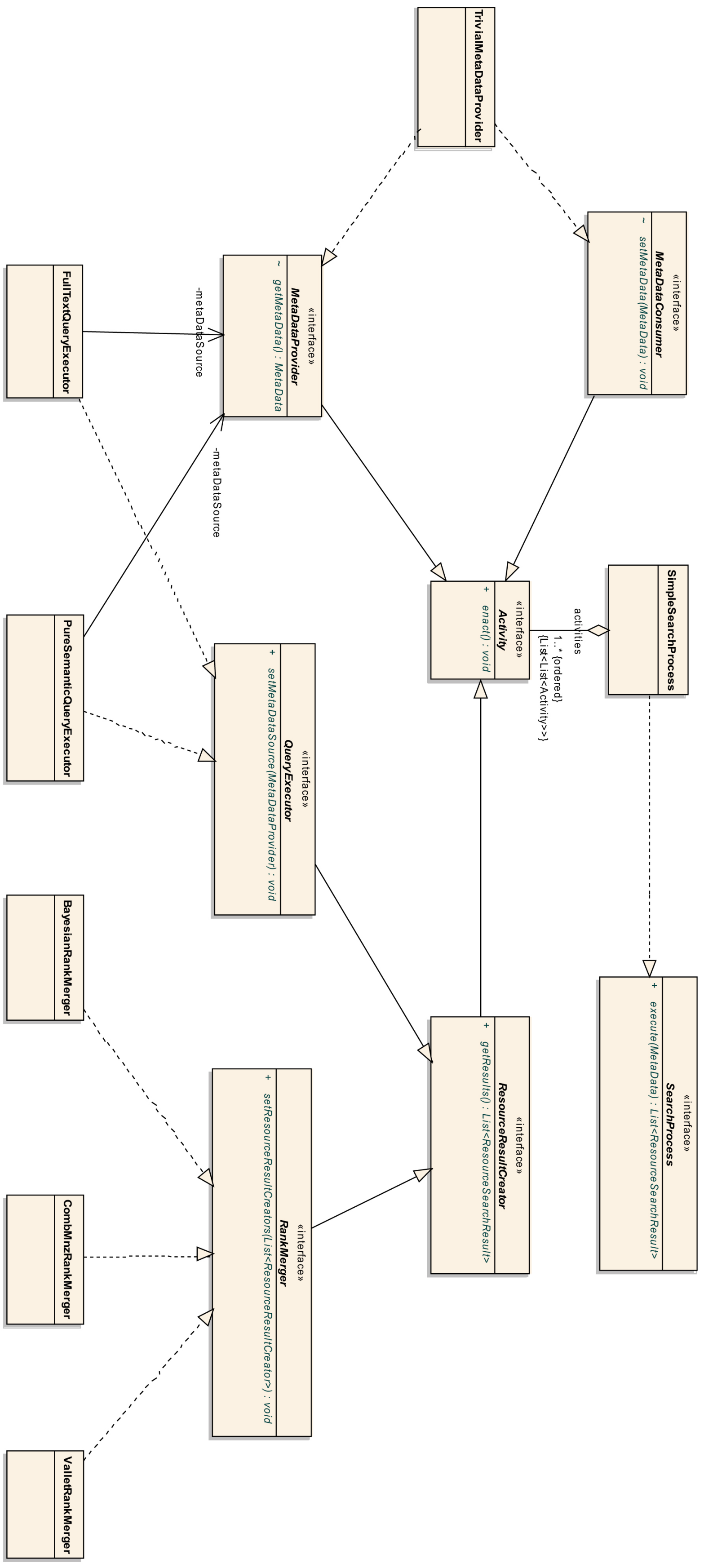


Figure 11.8: Classes and interfaces of the search process implementation

kinds of search processes. E.g., implementations of `MetaDataProvider` may implement query rewriting algorithms that change the semantic query representation, implementations of `RankMerger` usually implement score combination algorithms, etc. In the current implementation some theoretical possibilities of this structure are not used, e.g., as was described in Chapter 5, query expansion of the semantic query is not executed.

The `SimpleSearchProcess` makes the assumption that the first activity layer consists of one `MetaDataConsumer`, and injects its input `MetaData` to this activity. Further, it assumes that the last activity layer consists of one `ResourceResultCreator` activity, and returns the output of this activity as its result. By plugging in different implementations of the `Activity` interface (and its subinterfaces) into this search process implementation, it is possible to implement a very wide range of search processes.

The implementations of the interfaces refer to their dependencies only by their interfaces, and these dependencies are injected by the Spring framework. E.g., the `BayesianRankMerger` class that implements the Bayesian algorithm for combining search results refers only to a list of `ResourceResultCreator` interfaces. Which are the actual implementations of this interface, and how many of them are actually passed to this implementation, are unknown. The actual implementations of the interfaces are defined as Spring beans (components), and injected to other Spring beans as defined in the Spring XML configuration file. E.g., the definition of a search process running a metadata search and a content search, and combining the search results via the Bayesian algorithm is shown in Listing 11.1.

By combining the beans in a different way, a search process can be easily configured that executes only the metadata search. This configuration is shown in Listing 11.2.

Listing 11.1: Spring definition of the Bayesian combined search process

```
<bean id="simpleBayesianSearchProcess"
 class="de.fzi.ipe.ircon.search.process.SimpleSearchProcess">
 <property name="activities">
  <list>
   <list>
    <ref bean="trivialMetaDataProvider" />
   </list>
   <list>
    <ref bean="metaDataQueryExecutor" />
    <ref bean="contentQueryExecutor" />
   </list>
   <list>
    <ref bean="bayesianRankMerger" />
   </list>
  </list>
 </property>
</bean>
```

In addition to the flexibility in designing new processes, this solution also allows the separate testing of search process activities. E.g., the `BayesianRankMerger` class can be implemented and tested without providing any implementations of the

Listing 11.2: Spring definition of the metadata only search process

```
<bean id="metaDataSearchProcess"
 class="de.fzi.ipe.ircon.search.process.SimpleSearchProcess">
 <property name="activities">
  <list>
   <list>
    <ref bean="trivialMetaDataProvider" />
   </list>
   <list>
    <ref bean="metaDataQueryExecutor" />
   </list>
  </list>
 </property>
</bean>
```

`ResourceResultCreator` interface. For testing purposes any mock object framework may be used. During the IRCON implementation we used the jMock[4] framework.

Using mock objects, it is possible to simulate the results of a method invocation of an interface without actually providing a real implementation of that interface. E.g., in the case of the `BayesianRankMerger` class the results returned by the `getResults()` methods of the input `ResourceResultCreator` interfaces can be exactly specified. In doing so, it can be tested that the `BayesianRankMerger` class merges the results exactly as expected without actually implementing or executing the querying part of the search process.

11.2.5 Flexible rule definitions

Besides the flexibility to define different implementations of a specific interface, flexibility is also needed in another place. The heuristic rules are dependent on the application scenario and on the ontology, and it should be possible to define them very easily without modifying the IRCON implementation. The IRCON application allows the definition of heuristic rules in an XML file, i.e., no programming is needed to specify new rules. The XML Schema describing the XML configuration file of the heuristic rules is listed in Appendix C.1. As an example, the XML definition of the of the subevent rule that was discussed in Chapter 8 and shown in Figure 8.7 is displayed in Listing 11.3.

The full listing of the XML configuration file that was used during evaluation is shown in Appendix C.2.

[4] `http://www.jmock.org/`

Listing 11.3: XML definition of the subevent rule

```
<rule enable="yes" amplification_factor="0.7" name="1␣Subevent">
  <condition weight_threshold="0.2">
    <source_concept>urn:ircon:ontology#Event</source_concept>
    <cardinality>1</cardinality>
    <relation>urn:ircon:ontology#isSubEventOf</relation>
  </condition>
  <temporal_attribute>
  urn:ircon:ontology#happensDuring
  </temporal_attribute>
  <target_concept>urn:ircon:ontology#Event</target_concept>
</rule>
```

11.3 Administration workflow

Before the system is ready to provide support for the end-users, the necessary artifacts that are needed by the system have to be generated. This is the task of the system administrator. The tasks of the system administrator and the generated artifacts are shown in Figure 11.9. The administration workflow has two major parts: the tasks to generate and index the ontology, and the tasks to generate and index document metadata that use this ontology.

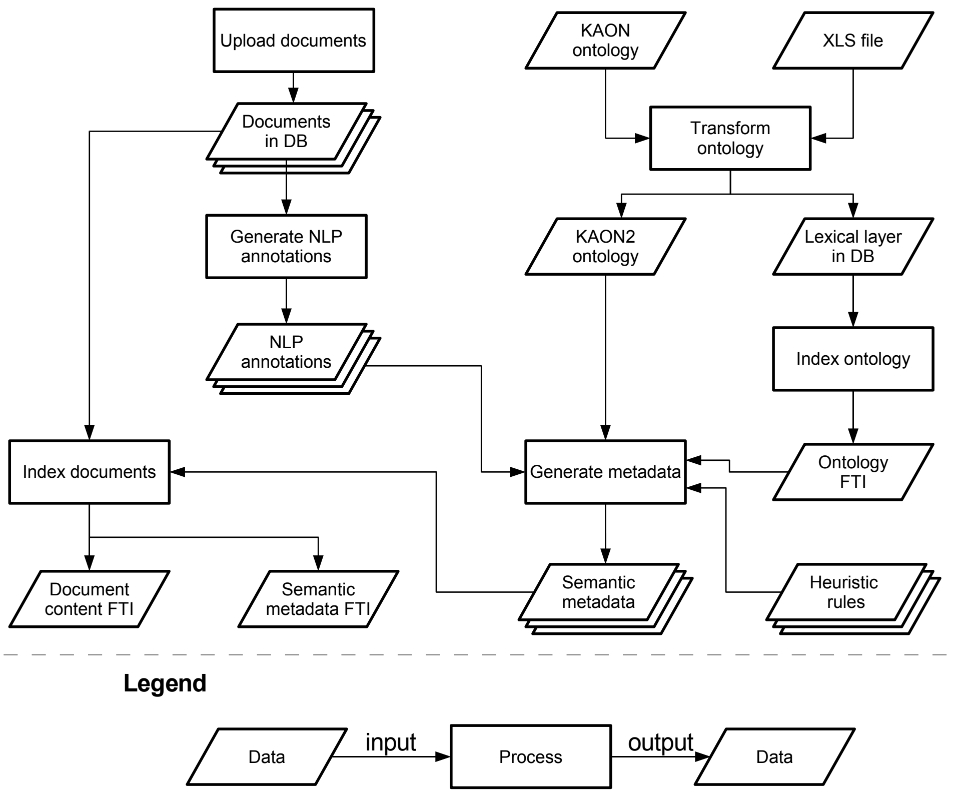


Figure 11.9: The administration workflow

To generate the ontology, the administrator first has to create the implementation model of the ontology out of the intermediate model representation. As was described in Chapter 10, the intermediate model in our case consists of two files: a KAON1 model describing concepts and properties, and an Excel file containing the instances together with their relations and attributes. The implementation formalism is described in Chapter 7, and is implemented using the KAON2 system. As was described before, for performance reasons the lexical layer of the ontology is not stored in the implementation formalism but it is stored in a classical relational database. In the next step, this lexical layer is indexed with a traditional full-text search engine to allow efficient search in the ontology lexical layer.

On the metadata generation side of the workflow, the first task is to upload the documents into the database. The IRCON system is capable to upload any document in HTML format that were downloaded using any crawler tool, such as wget[5]. During upload, the textual content has to be extracted from the HTML representation for indexing purposes. Moreover, also a simplified HTML representation has to be created that is used by the IRCON user interface to display the document content. This simplified representation is needed because different websites use different layouts and design elements, and they also often contain advertisements and menu elements that are not relevant for IR purposes. Moreover, often documents downloaded directly from the Internet do not even contain valid HTML content.

These facts make the extraction of the textual content a challenging task, too. The IRCON prototype automatically corrects erroneous HTML content by using NekoHTML[6] that provides a valid XHTML (and thus XML) document as its output. The prototype also allows administrators to specify XPath statements for specific Internet domains to extract useful content from the HTML documents. Only parts of the XML tree that match the specified XPath statement(s) are extracted and simplified. For the extraction of textual content and for the simplification of HTML, NekoHTML is used again. This tool can remove arbitrary HTML elements from the XHTML tree. For a simplified HTML presentation some HTML elements are left intact; for the text extraction, simply all HTML elements are removed.

Another solution was needed for the purposes of the evaluation where the whole Wikipedia had to be uploaded. Wikipedia does not allow crawling of its whole site because of obvious performance reasons. Instead, an XML dump of all Wikipedia articles is provided for download. The IRCON prototype supports the uploading of these Wikipedia dumps to its resource repository.

After the documents are uploaded to the repository, NLP annotations have to be generated. As was already described in the previous chapters, the GATE framework is used for this purpose.

In the next step, semantic metadata is generated. The NLP annotations, the ontology, the ontology full-text index and the heuristic rules are all needed to accomplish this task. The role of these artifacts is described in Chapter 8 in detail, with the exception of the ontology index. The ontology full-text index is used for the lookup of OI candidates for a specific text snippet, i.e., during the generation of the initial metadata.

[5]http://www.gnu.org/software/wget/
[6]http://people.apache.org/~andyc/neko/doc/html/

Finally, the documents are indexed. During this step, two full-text indexes are generated: one index over the textual content of the documents, and another index based on the semantic metadata of the documents. This process was already described in Chapter 9 in detail.

After the initial installation, there are two common situations when administrative tasks have to be executed. First, the ontology can change. In this case, practically all of the administrative steps have to be executed on all documents, with the exception of document upload and NLP annotation steps[7]. Second, new documents can be added to the system[8]. In this case, the document upload, NLP annotation, metadata generation and document indexing steps have to be executed on the new documents.

11.4 Performance issues

Although most of the performance issues are solved by using the scalable full-text search engine Lucene for semantic search, there are also other parts of the IRCON application where performance matters. Although metadata generation does not have to be an interactive process, it is clear that when thousands of documents have to be indexed (which is usually the case), every second matters. Although the upper time limit for indexing a given number of documents is highly dependent on the actual application case, an approach that requires days just to index several documents is very likely not feasible for any application scenarios. Another area where performance matters a lot is the GUI: users should be able to interactively browse the ontology and view documents (together with their metadata), in addition to searching.

As was already discussed in Chapter 3, ontology reasoning is highly complex. Therefore there are usually performance issues. During the development of the VICODI and IRCON applications, we had performance issues when we tried to implement a specific functionality using ontology reasoning, or using the features provided by the KAON1 and KAON2 frameworks, in the following cases:

Full-text search in the lexical layer: Although the KAON1 framework provides the functionality to execute full-text search on ontology labels, this feature had scalability problems on bigger ontologies, such as the VICODI ontology. The KAON2 system does not provide such search features any more, partly motivated by our negative feedback based on the VICODI experiences.

Determine the number of instances for a concept: On the GUI during ontology browsing, we had the problem that some concepts had so many instances that it was not useful to display all of them on the GUI. Therefore, we defined an upper threshold on the number of instances for a concept. If the number of instances exceeded the threshold, only a search field was shown for a specific concept, where the user could execute a full-text search on the instance labels of the specific concept. To implement this functionality, it was necessary to determine the number of (direct and indirect) instances of a

[7]Although in some cases it is useful to update the GATE gazetteers for named entity recognition based on the ontology lexical layer, and rerun the NLP step to get better results for named entity recognition and coreference resolution.

[8]Of course, documents can be also deleted from the system but this does not cause any problems.

concept. It turned out, however, that this operation cannot be executed interactively for bigger ontologies.

Navigation in the ontology structure: During metadata generation, most of the heuristic rule operations require traversing the ontology graph. Although one traversing step did not take really long using the KAON2 system, many such steps had to be executed, and the sum of the required time was too high for practical scenarios (e.g., for executing the evaluation of the system).

The generic idea to solve the first two issues can be summarized in the following principle: “use the ontology directly only when really necessary, generate artifacts out of the ontology whenever possible”. By following this principle, it is always possible to use the most efficient technology available for a specific task. E.g., in our case the lexical layer of the ontology was first extracted into a relational database, and was later indexed by Lucene. Into the Lucene index also the URIs of the parent concepts of the instance were stored. This allowed to use full-text search also to implement the full-text search feature for instances of a specific concept, which was needed for ontology browsing. This index is also used during ontology lookup in initial metadata generation part of the semantic metadata generation step. Finally, the number of instances for a specific concept were pre-calculated, and the results were stored in the relational database. The relation between the ontology and the generated artifacts is shown in Figure 11.10

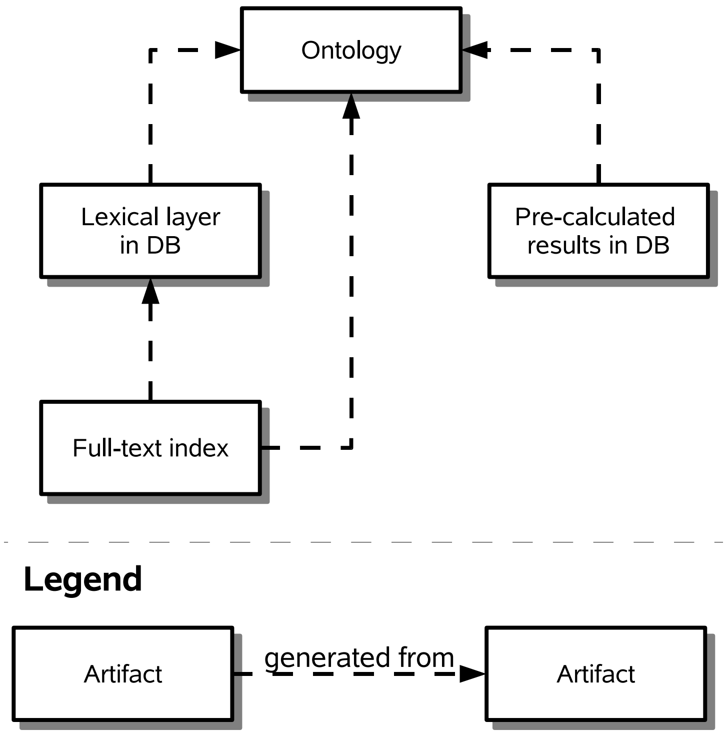


Figure 11.10: Artifacts generated from the ontology

The idea presented here is similar to the idea of the model driven architecture (MDA)[9], which recently gained some popularity in the area of software development. In MDA terms, the

[9]http://www.omg.org/mda/

ontology is the model and the "code" that is generated are the artifacts (database tables, full-text indexes). The major difference between MDA and our solution is the motivation to apply the technology: while in the case of MDA the goal is to speed-up the implementation process of an application, in our case the goal was to increase the performance of the application.

To solve the performance issue with ontology navigation, caching was used. During metadata expansion, often the same ontology reasoning steps had to be made. Clearly, in such a situation caching can be very effective. The importance of caching was already recognized by the developers of KAON2, and KAON2 provides a built-in caching functionality in its latest version. However, in the version of KAON2 which was available when the IRCON application was developed, this caching function was not yet implemented. Therefore, in the IRCON application an application-level cache was used. By applying caching, the metadata expansion step could be sped up by a factor of 10. As a result, the execution time of the metadata generation step (including metadata expansion) is now comparable (and in many cases even better) than the time needed by the NLP step.

11.5 Summary

In this chapter, the implementation details of the IRCON prototype application were discussed. The IRCON prototype is a comprehensive web application that allows even non-expert users to experiment with ontology-based IR. It was a non-trivial task to achieve this simplicity despite of the advanced and complex technologies that underly the system. Both the experiences with the VICODI prototype and the lessons learned from related work were exploited to reach this goal.

First, the system was described from the point of view of the end user. Next, I showed how the required flexibility was achieved in the IRCON architecture by applying the principle of "inversion of control", and using the features of the Spring framework. Later, I described the application from the perspective of the system administrator. Finally, a discussion of some performance issues related to ontology reasoning and their solutions closed this chapter.

Chapter 12

Evaluation

12.1 Introduction

As was discussed in Chapter 3, it cannot be stated for sure that using background knowledge in the IR process automatically increases the retrieval performance of an information system, as one would expect intuitively. On the contrary, sometimes it is even possible that retrieval performance decreases if we use hand-crafted background knowledge instead of simple statistical methods, as experience with thesauri-supported information systems shows. Because ontologies store hand-crafted background knowledge as well, the same problem can occur also in ontology-based information systems. Therefore, a thorough evaluation of retrieval performance is of paramount importance.

This chapter introduces and discusses the results of a large-scale evaluation using Wikipedia as test collection. The evaluation was conducted with the IRCON system that was implemented based on the ideas introduced in this thesis.

12.2 Hypotheses to evaluate

The main question of this thesis is whether possibly imperfect ontologies can help improve IR effectiveness. Therefore, the main hypothesis to evaluate is the following:

Hypothesis (Main)**.** *Possibly imperfect ontologies improve the retrieval performance of an information system.*

Checking this hypothesis effectively means comparing the results of the new system with the results of a legacy full-text search engine.

Besides checking this main hypothesis, however, there are some other hypotheses that were identified during the course of this thesis. Those should be evaluated, as well. These hypothesis are the following:

Hypothesis (H1)**.** *Metadata search alone provides good results in areas that are covered by an ontology, even if the ontology is imperfect.*

As was discussed in Chapter 3, it is impossible in practice to construct an ontology that contains all entities appearing in a generic document collection, together with all of the interesting relations between those entities. Therefore, even in areas that are in principle covered by the ontology, the ontology will be imperfect in the sense that it will not contain all possible information. Still, it is expected that in such areas the positive effect of using an ontology will outweigh the negative effect of the imperfection.

Hypothesis (H2)**.** *Combining content and metadata query results diminishes the effects of ontology imperfection.*

As was mentioned above, there will usually be topic areas of a document collection that are not covered at all or not covered adequately by an ontology. Clearly, if a user query refers to such parts, the results of the metadata search alone will not be adequate. The intuitively compelling solution to eliminate this negative effect is to combine the results of classical full-text search engines (in our case the content search) and the results of the semantic search (in our case the metadata search). It should be evaluated here whether this strategy really helps to improve the search results. It is also important to see that there are many different algorithms to combine search results, as was shown in Chapter 9. Therefore, the evaluation should also show which strategy is the best for combining content and metadata search results.

Hypothesis (H3)**.** *The higher quality the ontology has, the better the retrieval performance of the system is.*

As was seen in Chapter 4, most of the state of the art systems use only linguistic information and sometimes the ontology concept taxonomy during metadata generation and information retrieval. The intuitively compelling hypothesis, which was explained in this thesis, is that using more advanced ontology constructs can further increase retrieval performance. More specifically, I was interested whether adding non-linguistic connections and (fuzzy) temporal information to the ontology has positive effects on IR effectiveness.

Hypothesis (H4)**.** *It is possible to use advanced ontology constructs and provide an interactive system (according to the* SCALABILITY *requirement) at the same time.*

One possible cause why most state of the art systems do not exploit complex ontology constructs is that ontology reasoning is highly complex and therefore time intensive. Extensive use of ontology reasoning during query time would prohibit building an interactive system, which would violate the SCALABILITY requirement. As was described in Chapter 5 and Chapter 9, the idea used in this thesis is to exploit existing and mature full-text search engines during the query time and use complex ontology reasoning only during metadata generation. It should be evaluated whether the performance of such a hybrid system meets the criteria of the SCALABILITY requirement.

12.3 Retrieval performance evaluation basics

The performance of an information system can be easily evaluated by measuring response times. Evaluating retrieval effectiveness, however, is a highly complex issue. In this section I review the basics of retrieval performance evaluation that are needed to understand the results presented in this chapter. More details on the issue can be found in any IR textbook (e.g., [BR99] or [Fer03]).

12.3.1 Need for laboratory evaluation

First of all, because IR is an iterative and subjective process (see also Section 2.1), real evaluation would need real users who try to satisfy their information needs to solve various tasks. The ultimate value of an information system could then be measured by checking user satisfaction. There are many problems with this approach, however. First, it is extremely expensive to conduct such an evaluation. Second, the results of the evaluation are dependent on the evaluating user and also on the evaluation task. Therefore, it is practically impossible to repeat and reuse such evaluation results in other contexts.

Knowing the problems and limitations of the "real-world" evaluation, in most of the IR publications the so-called "laboratory evaluation" is used. Here the results of an information system are compared to some "golden standard" that describes the perfect results for a specific information need. Although this approach ignores many of the characteristics of a real IR process [1], it has the advantage that the results can be reproduced and therefore various retrieval algorithms can be compared with each other. This is also the approach which is used in the well-known Text REtrieval Conference (TREC)[2] series.

12.3.2 Performance measurement in laboratory evaluation

The two main measures that are checked in laboratory experiments are *precision* and *recall*. Let R_C denote all relevant documents (for a specific information need, specified as a user query) in the document collection. Let A denote the documents the system returns for the user query. Let R_A denote the relevant documents in the response set. Let $|S|$ denote the size of a set S.

Precision is defined as the portion of relevant documents in the response set, i.e.,

$$\text{Precision} = \frac{|R_A|}{|A|}$$

Recall is defined as the portion of relevant documents that were returned by the system, i.e.,

$$\text{Recall} = \frac{|R_A|}{|R_C|}$$

It is important to see that precision and recall are normally in a reciprocal relation. I.e., the higher the precision is, the lower the recall is. E.g., if a system dumps the whole document collection on each request, its recall will be maximal but the precision will be very low.

[1] e.g., subjectivity and the fact that it is an iterative process with continuously changing information need

[2] `http://trec.nist.gov/`

Most IR techniques that increase one of the two measures usually also decrease the other measure. E.g., thesauri are normally viewed as a technique that increases recall because the user query can be extended with linguistically related words. At the same time, thesauri usually also decrease precision because some of the newly inserted query words are not appropriate in the query context and therefore introduce false results into the result set.

An IR algorithm is "better" in an absolute sense than an other IR algorithm only if it provides better precision *and* better recall at the same time.

Because of the strong connection between precision and recall, normally precision and recall values are measured together. The standard technique is to take the ranked result list of an information system and calculate precision and recall based on the top n results. As we increase n, recall values will usually increase while precision values will usually decrease.

12.3.3 Precision-recall diagrams

The analysis of all precision-recall pairs would be impractical. Thus, in most studies these values are given only for some standard recall levels. The most common is to use the 11 recall levels from 0.0, 0.1, $0.2 \ldots$ up to 1.0. Of course, usually the exact precision value for a specific standard recall level cannot be calculated[3]. In this case, the precision value for a standard recall level is interpolated by taking the maximum precision value for any recall levels that are greater or equal than the given standard recall level.

Using these standard precision-recall pairs, a so-called *precision-recall diagram* (PR diagram) can be drawn that shows the precision values at the standard recall levels. Based on the criterion above, an algorithm is better in an absolute sense than an other algorithm if its curve lies always above the other curve in the PR diagram.

12.3.4 Precision@20 diagrams

While PR diagrams are a good instrument to reason about the quality of a retrieval algorithm, most users are rather interested in the quality of the highest ranked results of an information system. Studies show [SJ04, JS03] that in 70% of the time users on the Web view only the top 10 results of a search engine. On the average, users view 2.35 pages (one result page contains 10 results) whereas 50% of the users do not advance to the second page and 75% of the users do not view more than 2 pages. Finally, users check only 5 documents in the result set before either their information need is satisfied or they give up and change to a different search engine.

These results show that real users are rather interested in getting good results early than in high recall values. Therefore, in addition to PR diagrams I also used another common evaluation technique, namely measuring the precision at a specific document position. Motivated by the web study results above, I measured precision up to the position 20. I termed the diagrams displaying these position-precision pairs Precision@20 (P@20) diagrams.

[3]E.g., the recall at position x is below 0.1, at position $x + 1$ is already above 0.1.

12.4 Evaluation methodology

12.4.1 Selecting the test collection

The biggest problem during retrieval performance evaluation is how to get the "golden standard" which specifies the set of relevant documents for some selected user queries. There are some standard test collections available that provide such golden standards. E.g., test collections produced by the TREC conferences are well-known and widely used. Unfortunately, these test collections are made for classical (statistics-based) information retrieval and they do not contain an ontology. Therefore, either an ontology should be developed for one of these test collections or a new test collection must be designed.

As I was not a domain expert in any of the TREC domains, I decided to take the English Wikipedia web encyclopedia[4] as the basis for my test collection. It contains domains where I was able to develop a test ontology[5]. Additional motivation for choosing Wikipedia was the fact that Wikipedia had the appropriate size to test system performance. Moreover, Wikipedia is also very well-known on the World Wide Web, therefore achieving better search results on this collection can have a significant impact on the state of the art of web search.

When developing a new test collection, usually the major question is how one can determine the set of all relevant documents (needed for calculating recall). It is normally impossible to scan all documents of a big collection for relevant documents. E.g., the Wikipedia dump I used contained more then one million documents. Fortunately, Wikipedia has the concept of "categories" which contain all documents that belong to an abstract topic. This category concept made it possible to identify relevant documents for a specific topic without scanning any Wikipedia documents. Moreover, the approach has the advantage that the inherent subjectivity of selecting the relevant documents for a specific topic was not an issue here. A Wikipedia category contains a consolidated set of relevant documents for a topic. This selection of relevant documents was agreed by all Wikipedia contributors.

12.4.2 Selecting the user queries to evaluate

After the decision about the test collection was made, I had to decide about the information needs (queries) to test. Based on the problem analysis (see Chapter 3), I could see that one of the weakest points of traditional full-text search (FTS) is the handling of abstract concepts (topics) where the name of the concept does not necessary appear in relevant documents. One of the main features of my system is the use of ontology-based heuristic rules that exploit non-linguistic background information stored in the ontology to explore such indirectly relevant concepts. Therefore, I chose to evaluate my approach by running queries about two abstract concepts: "War on Terrorism" (denoted as WOT) and "Gulf War" (denoted as Gulf).

My motivations to choose exactly those two concepts were the following: First, the topics are well-known, therefore I felt myself a domain expert and could easily design an ontology

[4]`http://en.wikipedia.org`, snapshots of Wikipedia are available at `http://download.wikimedia.org` as XML files.

[5]Because Wikipedia changes constantly, it is important to note that I used the 2006-01-25 dump of Wikipedia.

for these two topics. Second, there were many entities in the ontology that were relevant for both topics, e.g., Iraq, USA, or Saddam Hussein. Third, there was also a very characteristic case for disambiguation: two US-presidents with the same name (Bush) were involved in the two different conflicts. Finally, because of the overlapping, there was a potential of wrong inferences. E.g., based on a set of ontology instances, such as Iraq, USA, and Saddam Hussein, it is not easy to decide whether "War on Terrorism" or "Gulf War" is relevant. While it seemed to be relatively easy to improve search results in domains where the good inferences are clear, overlapping domains seem to be especially problematic for semantic technologies. Moreover, in real life overlapping domains seem to be rather the norm than the exception. Therefore, it was interesting to see how my system behaves in such a challenging situation. Getting good results in this test would be a good indicator that the approach can be useful also in the "real world", outside the laboratory.

12.4.3 Developing the evaluation ontology and rules

As the next step, an ontology had to be designed for the selected domain. I used the ontology development approach that was presented in Chapter 10 and created an ontology that described the two selected WOT and Gulf topics. The ontology itself was already discussed in detail in Section 8.6.2. A full list of ontology instances, relations, attributes, and the ontology lexical layer can be found in Appendix B.

Parallel to the ontology development, I also developed a list of heuristic rules to improve the generated metadata. Some examples of these rules were already discussed in Section 8.6.3. The XML dump of the rule definition file together with its XML Schema can be found in Appendix C.

12.4.4 Preparing the test collection for the evaluation

Finally, the test collection had to be prepared for the evaluation. First, the Wikipedia dump had to be processed and the textual and HTML content of the articles had to be extracted and stored in the repository of the prototype system. It is important to note that Wikipedia also contains lots of pages that were not relevant for my experiment. These included special pages (Wikipedia help, user talk protocols, images) and also pages that were only redirections (aliases) for "real" Wikipedia pages. Finally, I discarded also index pages that contained only a list of event or person names. Finally, 945843 unique documents remained after the cleaning process.

After executing the upload process, relevant documents for the two selected topics had to be determined by exploiting the already mentioned Wikipedia categories. To determine all relevant documents for a topic, first I selected all categories that contained the text of the topic in its title (e.g., the categories "Gulf War" and "Gulf War movies"). Next, I transitively extracted all subcategories of these core categories. Finally, I determined all Wikipedia articles that belonged to the extracted categories. I selected these articles as the relevant ones for the topic. With this approach, I got 38 relevant documents for the Gulf topic and 559 relevant documents for the WOT topic. Two of the Gulf documents did not contain the phrase "Gulf

War" and 469 of the WOT documents did not contain the phrase "War on Terrorism". I.e., especially in the area of WOT there was a great potential to improve search results.

Ideally, metadata for all Wikipedia documents should have been generated as the next step. Unfortunately, it was impossible. The GATE annotation process alone took 30 seconds on the average for one document. The generation of the semantic metadata based on the GATE annotation (including the metadata expansion step) took 10 seconds per document on the average. Based on these numbers the whole metadata generation process would have taken approx. 438 days, i.e., more than a year. Moreover, I needed to regenerate the metadata not only once but many times to experiment with various algorithms and ontology constructs. Therefore, generating metadata for all documents was clearly impossible.

As an alternative approach I did the following: I generated metadata for all documents that were selected as relevant for any of the two topics. This gave the chance for the semantic metadata generation process to assign the proper ontology instances to those documents. In addition, I also added some "noise" to the set of documents. My motivation was to simulate the situation where an irrelevant document contains many instances that appear in many relevant documents for an abstract concept. In such cases, there is a danger that the heuristic rules incorrectly infer that the abstract concept is relevant for the document. To simulate this situation, I ran the full-text query ***"Iraq AND Bush"*** with my Lucene-based full-text search engine (returned 972 documents) and included all result documents to the list of the documents to be annotated. With this approach, I got 1457 documents to annotate[6]. I.e., after an initial GATE annotation step of approx. 12 hours, I could regenerate the semantic metadata of the documents in approx. 4 hours.

12.4.5 Evaluation strategy

The final, but at the same time the most important, remaining question is how to evaluate the hypotheses that were discussed in Section 12.2.

Determining the queries to execute

To evaluate the main hypothesis, results of a classical full-text search engine (the baseline) have to be compared with results of the new system. As the baseline system I took my Lucene-based search engine. It is not trivial, however, to determine which full-text search query should be taken as the baseline. Lucene supports not only the vector space model (VSM, see also Section 2.3) but among others also (almost) arbitrary boolean queries and proximity queries. All Lucene queries are ranked using a variation of the usual TF-IDF measure. Therefore, first the effectiveness of the various FTS queries had to be evaluated. I selected four types of FTS queries for both topics, based on my personal experience with Lucene and other full-text search engines. I introduce the queries and their semantics on the example of the Gulf topic:

[6]Naturally, many documents in the "Iraq AND Bush" result set were also relevant for one of the selected topics.

- Query: ***Gulf War***. Query code: **FTS_OR**. This query returns all documents that contain any of the words “Gulf” or “War”[7]. This query gives the same results than a VSM algorithm would do.
- Query: ***Gulf AND War***. Query code: **FTS_AND**. This query returns all documents that contain both the “Gulf” and the “War” words. This query returns fewer results than the previous version.
- Query: ***"Gulf War"^3***. Query code: **FTS_proximity**. This query returns all documents that contain both the “Gulf” and the “War” words within three words distance (a so called “proximity” query). This variation returns even fewer results than the previous version.
- Query: ***"Gulf War"***. Query code: **FTS_quoted**. This query returns all documents that contain the exact phrase “Gulf War”. This variation returns the fewest results from all of the FTS queries.

Naturally, if there is a clear winner among these queries (based on the PR diagrams), the best should be taken as the baseline.

For the further experiments, metadata searches[8] and combined semantic searches[9] had to be executed. To meet the NATURAL LANGUAGE QUERY requirement, I used the phrases “Gulf War” and “War on Terrorism” as inputs for my system. Of course, these phrases were automatically parsed and transformed into the proper semantic query representation by the system, as was described in Chapter 9.

Query combination algorithms

To evaluate the effect of combining metadata and content query results to diminish the negative effects of ontology imperfection[10], the following strategy is used. The results of the metadata query are compared with the combined search results, following the various query combination strategies introduced in Chapter 9. These are the Bayesian inference strategy (Bayes), the CombMNZ strategy, and finally the strategy described by Vallet et al. The ontology used for the evaluation was developed with low effort and contains a relatively small number of instances and relations. Hence, it can be viewed as a typical imperfect ontology. Therefore, the expected results of the metadata search will not be optimal. Thus, I can check whether the combined search results are better than the pure metadata search results and whether one of the combination algorithms delivers significantly better results than the others.

Ontology versions for evaluation

To check the impact of ontology quality on the search results[11], the experiments are executed with different ontology versions. First, experiments are run with the full-featured version

[7]All Lucene queries are case insensitive.
[8]to evaluate **H1** hypothesis
[9]that combine the results of metadata and content queries, for evaluating the main hypothesis
[10]**H2** hypothesis
[11]**H3** hypothesis

of the ontology, containing the lexical layer (with labels and synonyms), the non-linguistic relations, normal and fuzzy time information (ontology code: *fuzzy time*). Second, the fuzzy time information in the ontology is replaced with a classical time interval. At the same time, the parsing of fuzzy time intervals during metadata generation is also switched off (ontology code: *normal time*). Next, all time information is removed from the ontology (ontology code: *no time*). Finally, also the non-linguistic relations are removed. So, the remaining ontology becomes a kind of thesaurus (ontology code: *no connection*).

The results of different experiments are compared with each other. To validate the hypothesis, an increase of the effectiveness is expected when more and more ontological features are used.

Checking the interactivity hypothesis

Finally, to check the interactivity hypothesis[12], the response time is measured for both topics for the most complex queries (combined query with all ontology features switched on).

As was discussed before, "comparing results" mean in all cases to produce and analyze PR diagrams and Precision@20 diagrams.

12.5 Evaluation results

12.5.1 Selecting the FTS baseline

The results of running the four types of FTS queries are shown in Figures 12.2 to 12.5 and in Tables 12.1 to 12.4. In the tables, the precision values that are maximal for a given recall level or for a given position are printed in bold face to help identify the best algorithm.

As we move from FTS_OR toward FTS_quoted, the queries become stricter and stricter. It can be expected that they will return fewer and fewer irrelevant results. On the other hand, a stricter query may miss some relevant documents. E.g., the FTS_quoted query will miss any relevant WOT documents that do not contain the exact "War on Terrorism" phrase. I.e., what intuitively can be expected is that the stricter queries will have higher precision values at low recall levels but the precision-recall curve will drop to zero sooner because they will find fewer relevant results.

This expectation is completely valid in the case of the WOT scenario, as it can be seen in Figure 12.2 and in Table 12.1. I.e., in an absolute sense none of the strategies is superior to the other, with the exception of FTS_proximity and FTS_quoted searches. These two searches drop to zero precision at the same recall level, thus FTS_quoted is better in an absolute sense than FTS_proximity.

In the case of the Gulf scenario, all algorithms drop to zero precision at the same recall level, similarly to the FTS_proximity and FTS_quoted search case in the WOT scenario (see Figure 12.4 and Table 12.3). This means that the FTS_proximity and FTS_quoted strategies are

[12]**H4** hypothesis

superior to the FTS_OR and FTS_AND strategies in an absolute sense. It is somewhat surprising that the FTS_proximity strategy performs better at some recall levels than the FTS_quoted strategy.

In the Precision@20 diagrams, however, there are no significant differences among the strategies for any of the scenarios. I.e., the end-user will not notice any change in quality if we change the FTS query strategy. It is really interesting that the FTS_OR algorithm, which was the worst in the PR diagram, provides the best results in the Precision@20 diagram in the Gulf scenario (see Figure 12.5 and Table 12.4).

As a conclusion, **it is not possible to identify a clear baseline for FTS**. The results of the new system should be compared with all of the discussed FTS strategies that are not worse than other FTS strategies in an absolute sense. Based on the discussion, the FTS strategies that will be taken as the baseline for the specific scenarios and diagram types are shown in Figure 12.1.

WOT PR diagram: FTS_OR, FTS_AND, FTS_quoted

WOT P@20 diagram: FTS_quoted

Gulf PR diagram: FTS_proximity, FTS_quoted

Gulf P@20 diagram: FTS_OR

Figure 12.1: FTS baselines for evaluation

Table 12.1: WOT full-text search precision-recall values

		Precision			
		OR	*AND*	*proximity*	*quoted*
Recall	*0.0*	**1.0**	**1.0**	**1.0**	**1.0**
	0.1	0.008	0.0694	0.0954	**0.1054**
	0.2	0.008	**0.0532**	0.0	0.0
	0.3	0.008	**0.0429**	0.0	0.0
	0.4	**0.006**	0.0	0.0	0.0
	0.5	**0.0035**	0.0	0.0	0.0
	0.6	0.0	0.0	0.0	0.0
	0.7	0.0	0.0	0.0	0.0
	0.8	0.0	0.0	0.0	0.0
	0.9	0.0	0.0	0.0	0.0
	1.0	0.0	0.0	0.0	0.0

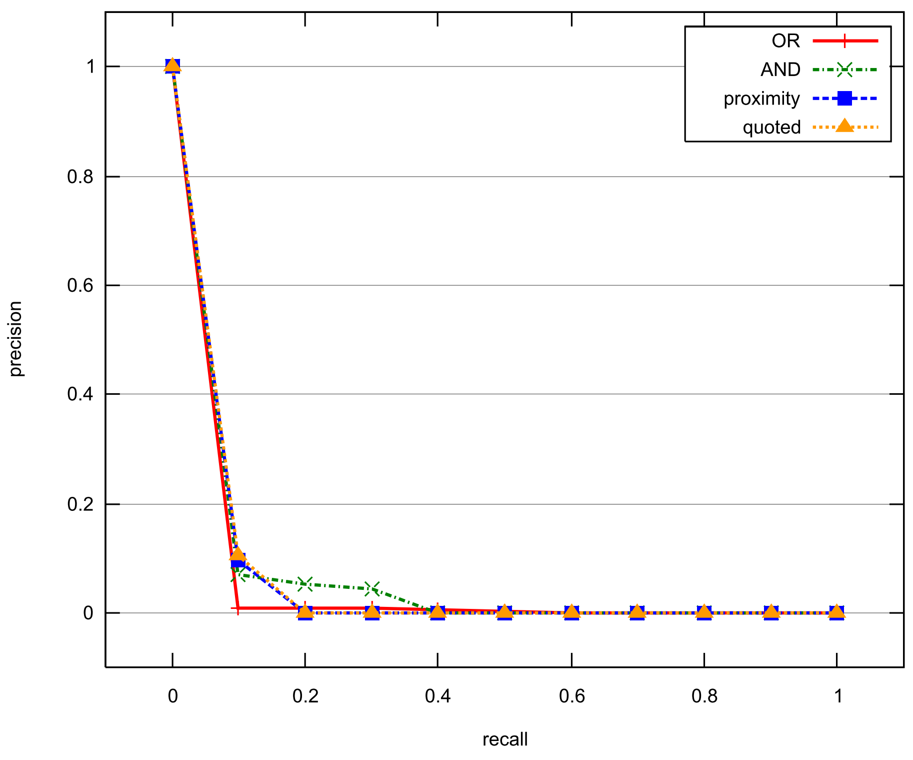


Figure 12.2: WOT full-text search precision-recall diagram

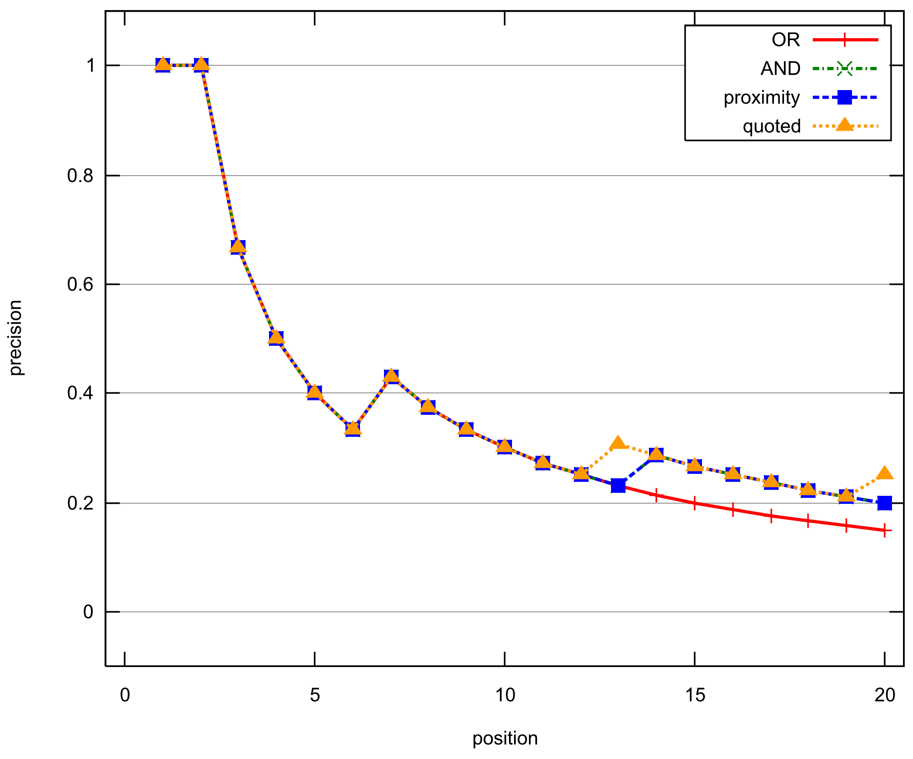


Figure 12.3: WOT full-text search precision@20 diagram

Table 12.2: WOT full-text search precision@20 values

		Precision			
		OR	*AND*	*proximity*	*quoted*
Position	*1*	**1.0**	**1.0**	**1.0**	**1.0**
	2	**1.0**	**1.0**	**1.0**	**1.0**
	3	**0.6667**	**0.6667**	**0.6667**	**0.6667**
	4	**0.5**	**0.5**	**0.5**	**0.5**
	5	**0.4**	**0.4**	**0.4**	**0.4**
	6	**0.3333**	**0.3333**	**0.3333**	**0.3333**
	7	**0.4286**	**0.4286**	**0.4286**	**0.4286**
	8	**0.375**	**0.375**	**0.375**	**0.375**
	9	**0.3333**	**0.3333**	**0.3333**	**0.3333**
	10	**0.3**	**0.3**	**0.3**	**0.3**
	11	**0.2727**	**0.2727**	**0.2727**	**0.2727**
	12	**0.25**	**0.25**	**0.25**	**0.25**
	13	0.2308	0.2308	0.2308	**0.3077**
	14	0.2143	**0.2857**	**0.2857**	**0.2857**
	15	0.2	**0.2667**	**0.2667**	**0.2667**
	16	0.1875	**0.25**	**0.25**	**0.25**
	17	0.1765	**0.2353**	**0.2353**	**0.2353**
	18	0.1667	**0.2222**	**0.2222**	**0.2222**
	19	0.1579	**0.2105**	**0.2105**	**0.2105**
	20	0.15	0.2	0.2	**0.25**

Table 12.3: Gulf full-text search precision-recall values

		Precision			
		OR	*AND*	*proximity*	*quoted*
Recall	*0.0*	**1.0**	**1.0**	**1.0**	**1.0**
	0.1	**0.625**	**0.625**	**0.625**	**0.625**
	0.2	0.0104	0.2258	0.2703	**0.2778**
	0.3	0.0104	0.2258	0.2571	**0.2609**
	0.4	0.0104	0.2258	0.2571	**0.2609**
	0.5	0.0104	0.2258	0.25	**0.2532**
	0.6	0.0022	0.1353	**0.2072**	0.097
	0.7	0.0016	0.0427	**0.0549**	0.0545
	0.8	0.0016	0.0209	**0.0473**	0.0436
	0.9	0.0016	0.0139	0.035	**0.0362**
	1.0	0.0	0.0	0.0	0.0

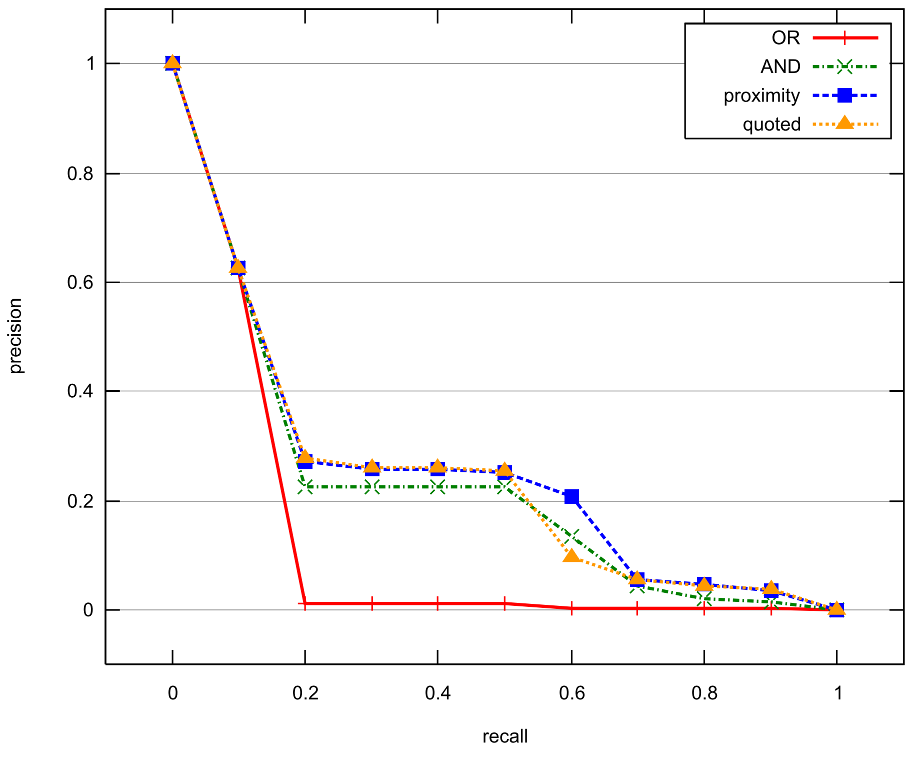


Figure 12.4: Gulf full-text search precision-recall diagram

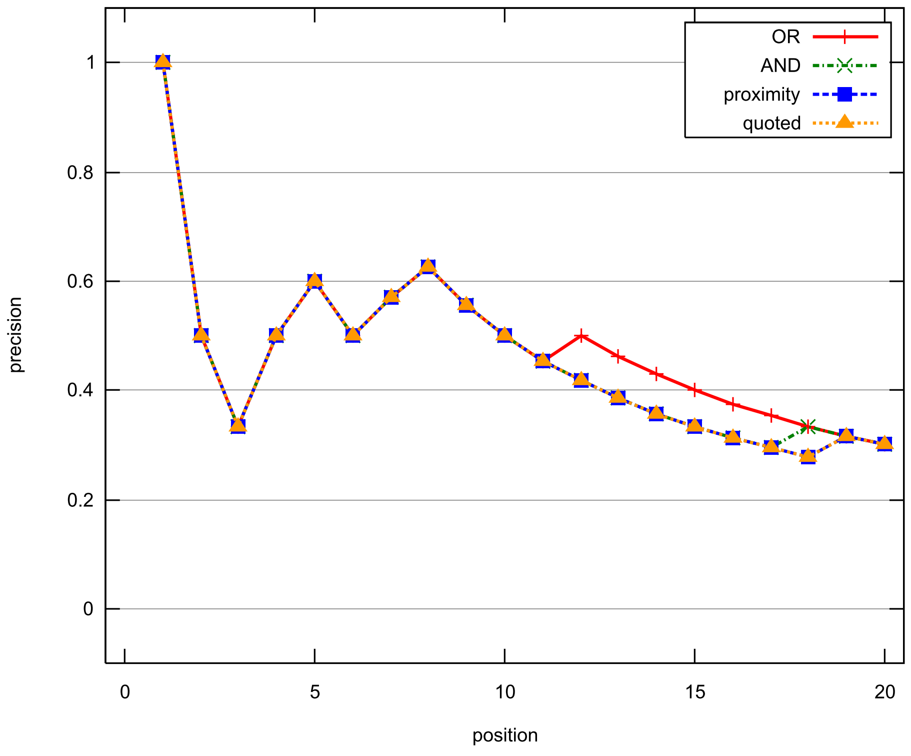


Figure 12.5: Gulf full-text search precision@20 diagram

Table 12.4: Gulf full-text search precision@20 values

		Precision			
		OR	*AND*	*proximity*	*quoted*
Position	*1*	**1.0**	**1.0**	**1.0**	**1.0**
	2	**0.5**	**0.5**	**0.5**	**0.5**
	3	**0.3333**	**0.3333**	**0.3333**	**0.3333**
	4	**0.5**	**0.5**	**0.5**	**0.5**
	5	**0.6**	**0.6**	**0.6**	**0.6**
	6	**0.5**	**0.5**	**0.5**	**0.5**
	7	**0.5714**	**0.5714**	**0.5714**	**0.5714**
	8	**0.625**	**0.625**	**0.625**	**0.625**
	9	**0.5556**	**0.5556**	**0.5556**	**0.5556**
	10	**0.5**	**0.5**	**0.5**	**0.5**
	11	**0.4545**	**0.4545**	**0.4545**	**0.4545**
	12	**0.5**	0.4167	0.4167	0.4167
	13	**0.4615**	0.3846	0.3846	0.3846
	14	**0.4286**	0.3571	0.3571	0.3571
	15	**0.4**	0.3333	0.3333	0.3333
	16	**0.375**	0.3125	0.3125	0.3125
	17	**0.3529**	0.2941	0.2941	0.2941
	18	**0.3333**	**0.3333**	0.2778	0.2778
	19	**0.3158**	**0.3158**	**0.3158**	**0.3158**
	20	**0.3**	**0.3**	**0.3**	**0.3**

12.5.2 Results of the metadata query

The **H1** hypothesis states that metadata search alone provides good results in areas that are covered by an ontology, even if the ontology is imperfect. To evaluate this hypothesis, I compared the results of the metadata query with the results of the FTS baseline. The results of this comparison are shown in Figures 12.6 to 12.9 and in Tables 12.5 to 12.8.

As the figures and tables show, **the metadata search is clearly superior to all FTS strategies in an absolute sense**[13]. This is a very interesting and encouraging result because (as was discussed in Section 8.6.2) the evaluation ontology was developed with low effort, and thus it can be viewed as a typical imperfect ontology that is developed in the real world. In particular, this ontology did not contain many of the entities (persons, organizations, events) that are relevant for the WOT and Gulf topics. It did not contain many of the interesting relations and time specifications, either. Still, even under such conditions, metadata search could clearly outperform FTS.

To summarize the discussion, the **H1** hypothesis could be successfully evaluated.

[13]With a small exception: in the WOT scenario the FTS_OR strategy has a marginally higher precision value at the recall level 0.5 (see Table 12.6).

Table 12.5: Comparison of WOT full-text and metadata search (precision-recall values)

		Precision			
		FTS OR	*FTS AND*	*FTS quoted*	*Metadata*
Recall	*0.0*	**1.0**	**1.0**	**1.0**	**1.0**
	0.1	0.0080	0.0694	0.1054	**0.725**
	0.2	0.0080	0.0532	0.0	**0.6505**
	0.3	0.0080	0.0429	0.0	**0.6145**
	0.4	0.0059	0.0	0.0	**0.549**
	0.5	**0.0035**	0.0	0.0	0.0
	0.6	**0.0**	**0.0**	**0.0**	**0.0**
	0.7	**0.0**	**0.0**	**0.0**	**0.0**
	0.8	**0.0**	**0.0**	**0.0**	**0.0**
	0.9	**0.0**	**0.0**	**0.0**	**0.0**
	1.0	**0.0**	**0.0**	**0.0**	**0.0**

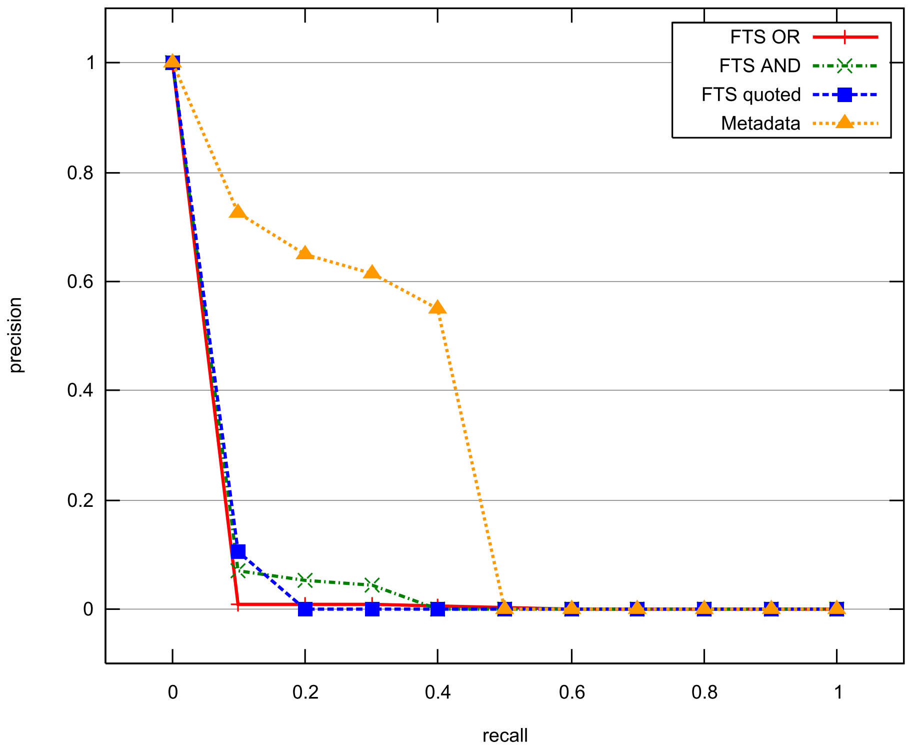


Figure 12.6: Comparison of WOT full-text and metadata search (precision-recall diagram)

Table 12.6: Comparison of WOT full-text and metadata search (precision@20 values)

		Precision	
		FTS quoted	*Metadata*
Position	*1*	**1.0**	**1.0**
	2	**1.0**	**1.0**
	3	0.6667	**1.0**
	4	0.5	**1.0**
	5	0.4	**1.0**
	6	0.3333	**1.0**
	7	0.4286	**0.8571**
	8	0.375	**0.875**
	9	0.3333	**0.7778**
	10	0.3	**0.7**
	11	0.2727	**0.7273**
	12	0.25	**0.75**
	13	0.3077	**0.7692**
	14	0.2857	**0.7857**
	15	0.2667	**0.8**
	16	0.25	**0.8125**
	17	0.2353	**0.7647**
	18	0.2222	**0.7778**
	19	0.2105	**0.7895**
	20	0.25	**0.8**

Table 12.7: Comparison of Gulf full-text and metadata search (precision-recall values)

		Precision		
		FTS proximity	*FTS quoted*	*Metadata*
Recall	*0.0*	**1.0**	**1.0**	**1.0**
	0.1	0.625	0.625	**1.0**
	0.2	0.2703	0.2778	**1.0**
	0.3	0.2571	0.2609	**0.75**
	0.4	0.2571	0.2609	**0.5862**
	0.5	0.25	0.2532	**0.5588**
	0.6	0.2072	0.097	**0.4182**
	0.7	0.0549	0.0545	**0.2813**
	0.8	0.0473	0.0436	**0.22**
	0.9	0.035	0.0362	**0.2**
	1.0	0.0	0.0	**0.1751**

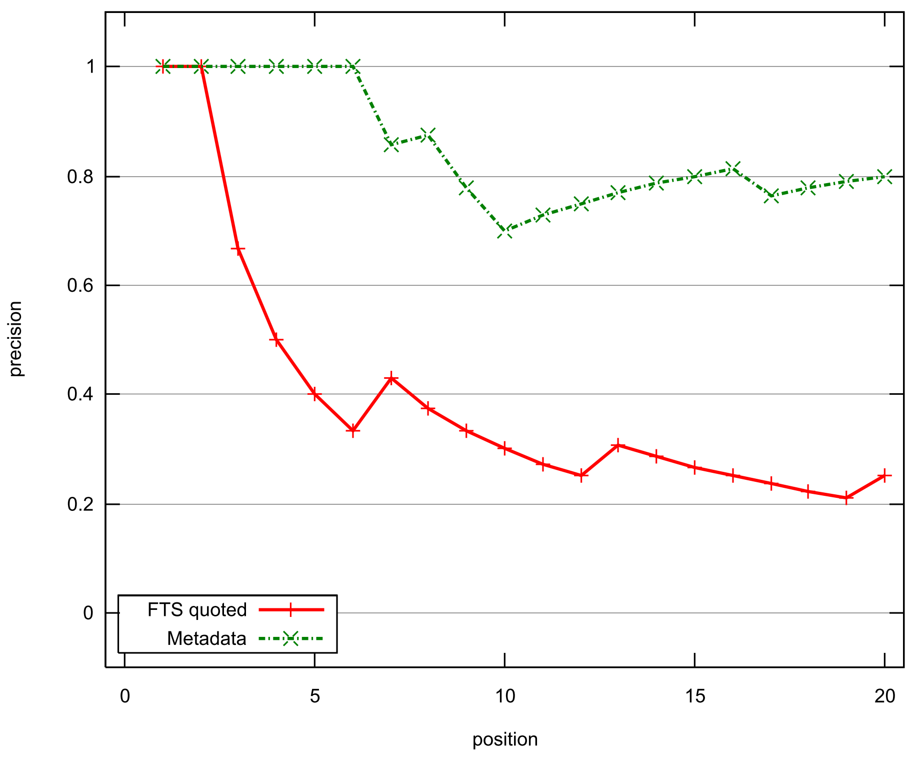


Figure 12.7: Comparison of WOT full-text and metadata search (precision@20 diagram)

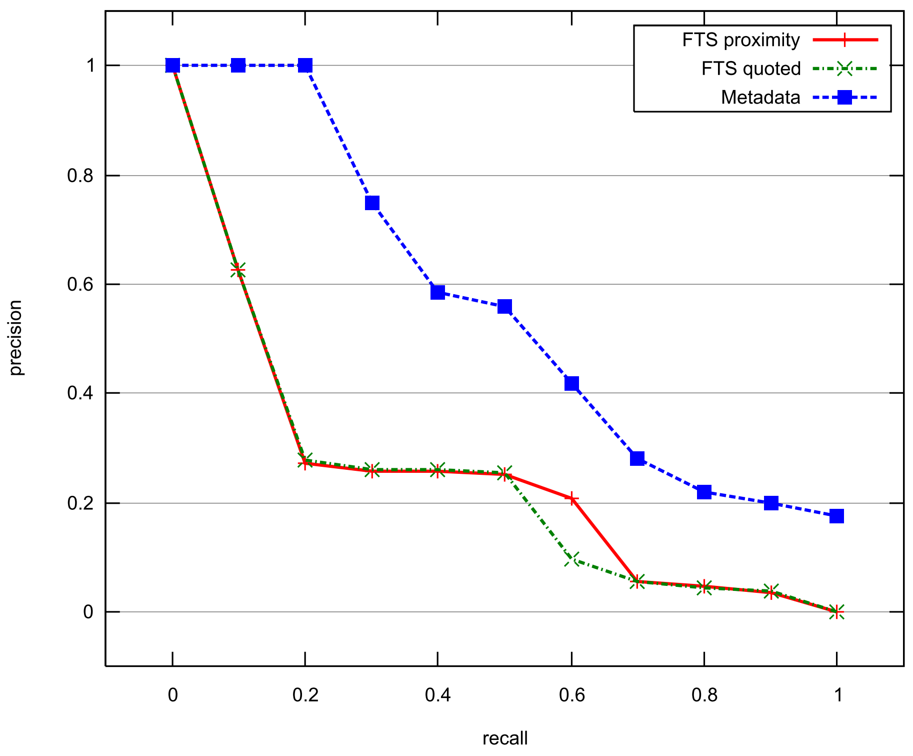


Figure 12.8: Comparison of Gulf full-text and metadata search (precision-recall diagram)

Table 12.8: Comparison of Gulf full-text and metadata search (precision@20 values)

		Precision	
		FTS OR	*Metadata*
Position	*1*	**1.0**	**1.0**
	2	0.5	**1.0**
	3	0.3333	**1.0**
	4	0.5	**1.0**
	5	0.6	**1.0**
	6	0.5	**1.0**
	7	0.5714	**1.0**
	8	0.625	**1.0**
	9	0.5556	**1.0**
	10	0.5	**1.0**
	11	0.4545	**0.9091**
	12	0.5	**0.8333**
	13	0.4615	**0.7692**
	14	0.4286	**0.7143**
	15	0.4	**0.7333**
	16	0.375	**0.75**
	17	0.3529	**0.7059**
	18	0.3333	**0.6667**
	19	0.3158	**0.6316**
	20	0.3	**0.6**

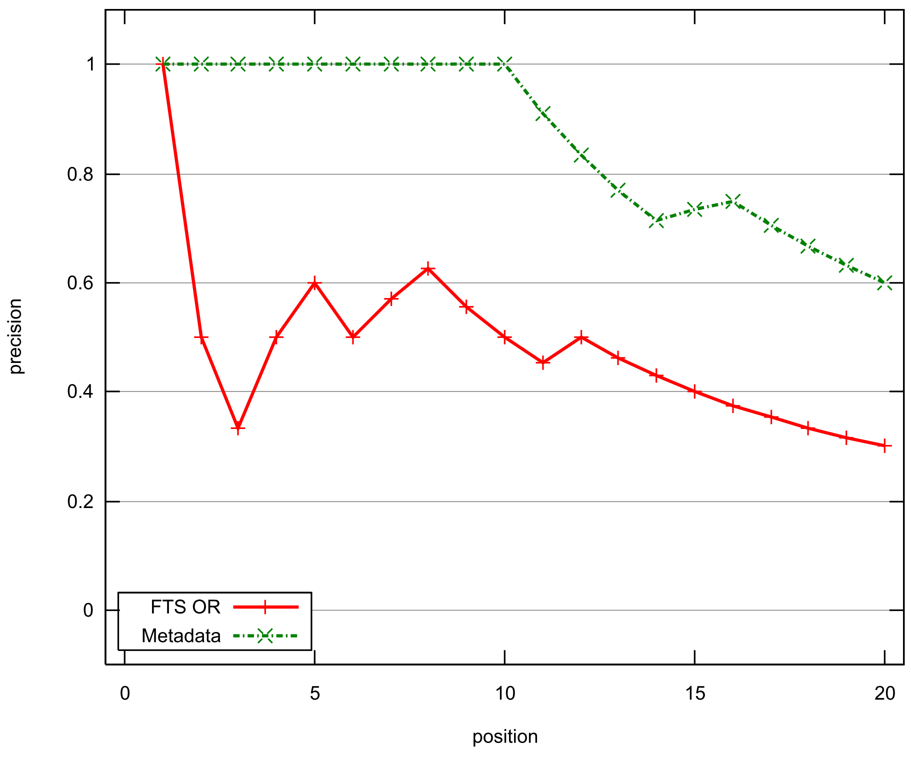


Figure 12.9: Comparison of Gulf full-text and metadata search (precision@20 diagram)

12.5.3 Results of query combination

The **H2** hypothesis states that combining content and metadata query results diminishes the effects of ontology imperfection. To evaluate this hypothesis, I analyzed how the combination of the content results with the metadata search results affects the retrieval performance. The experimental results are shown in Figures 12.10 to 12.13 and in Tables 12.9 to 12.12.

Naturally, the retrieval performance of the underlying metadata and content search algorithms affects the efficiency of the combined algorithm. Based on the experiences from metasearch research, an increase in retrieval performance is expected if all algorithms provide good results already by themselves. In addition, the sets of returned relevant documents should significantly overlap while the sets of returned non-relevant documents should not overlap significantly [Cro00].

There is no clear winner among the algorithms in any of the evaluated scenarios. This means that in many situations, the metadata search performs better than the combined search. I.e., the combination of the results even slightly decreases the retrieval performance in some cases. In other words, the **H2** hypothesis could not be validated in the chosen scenarios. The reason is most probably that the results of the metadata search are too good in comparison with the content search. I.e., the criteria for a successful result combination are not fulfilled because the content search results are of too low quality.

What is probably more surprising is that none of the combination algorithms could perform better than the others. E.g., while the CombMNZ algorithm was clearly the best in some scenarios (see e.g., Figure 12.12), it was the worst in some other scenarios (see e.g., Figure 12.10 and Figure 12.11). It seems that the effectiveness of the combination is highly domain-dependent. However, it is impossible to predict which algorithm will provide the best performance in a specific case.

It seems based on these results that the strategy proposed by Vallet et al. [VFC05] has the weakest retrieval performance on the average, although of course there are scenarios where it provides quite a good results (see e.g., Figure 12.10).

Table 12.9: WOT metadata and combined semantic search (fuzzy) precision-recall values

		Precision			
		Metadata	*Bayes*	*CombMNZ*	*Vallet*
Recall	*0.0*	**1.0**	**1.0**	**1.0**	**1.0**
	0.1	0.725	**0.7273**	0.7053	0.7143
	0.2	**0.6505**	0.6348	0.6278	0.645
	0.3	**0.6146**	0.5965	0.5819	0.6036
	0.4	**0.549**	0.5271	0.4817	0.5463
	0.5	0.0	**0.4409**	**0.4409**	**0.4409**
	0.6	0.0	0.0	0.0	0.0
	0.7	0.0	0.0	0.0	0.0
	0.8	0.0	0.0	0.0	0.0
	0.9	0.0	0.0	0.0	0.0
	1.0	0.0	0.0	0.0	0.0

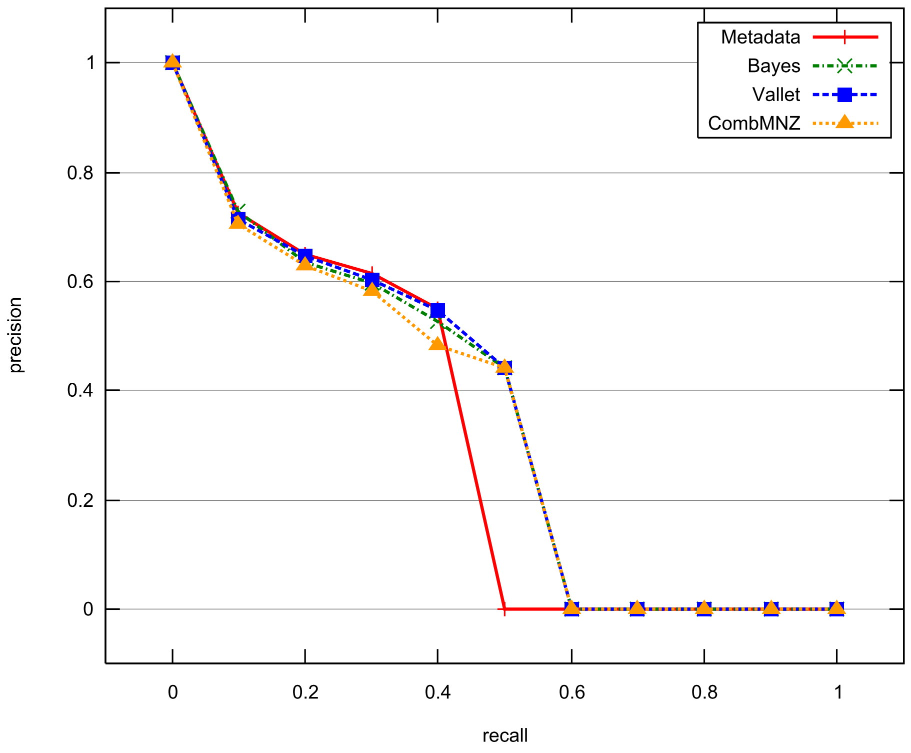


Figure 12.10: WOT metadata and combined semantic search (fuzzy) precision-recall diagram

Table 12.10: WOT metadata and combined semantic search (fuzzy) precision@20 values

		Precision			
		Metadata	*Bayes*	*Vallet*	*CombMNZ*
Position	*1*	**1.0**	**1.0**	**1.0**	**1.0**
	2	**1.0**	**1.0**	**1.0**	**1.0**
	3	**1.0**	**1.0**	**1.0**	0.6667
	4	**1.0**	**1.0**	0.75	0.5
	5	**1.0**	**1.0**	0.8	0.4
	6	**1.0**	0.8333	0.8333	0.5
	7	**0.8571**	**0.8571**	**0.8571**	0.5714
	8	**0.875**	**0.875**	**0.875**	0.625
	9	0.7778	**0.8889**	0.7778	0.6667
	10	0.7	**0.9**	0.8	0.7
	11	0.7273	**0.9091**	0.8182	0.6364
	12	0.75	**0.8333**	**0.8333**	0.6667
	13	0.7692	**0.8462**	**0.8462**	0.6923
	14	0.7857	**0.8571**	**0.8571**	0.7143
	15	0.8	**0.8667**	**0.8667**	0.7333
	16	0.8125	0.8125	**0.875**	0.6875
	17	0.7647	0.8235	**0.8824**	0.6471
	18	0.7778	0.8333	**0.8889**	0.6667
	19	0.7895	0.7895	**0.8421**	0.6842
	20	**0.8**	**0.8**	**0.8**	0.65

Table 12.11: Gulf metadata and combined semantic search (fuzzy) precision-recall values

		Precision			
		Metadata	*Bayes*	*CombMNZ*	*Vallet*
Recall	*0.0*	**1.0**	**1.0**	**1.0**	**1.0**
	0.1	**1.0**	0.9091	**1.0**	0.8571
	0.2	**1.0**	0.9091	**1.0**	0.75
	0.3	0.75	0.6842	**0.8**	0.4375
	0.4	0.5862	0.5938	**0.6296**	0.4103
	0.5	0.5588	0.5938	**0.6**	0.3571
	0.6	0.4182	0.4902	**0.6**	0.3571
	0.7	0.2813	0.2947	**0.4118**	0.2362
	0.8	0.22	0.2793	**0.3563**	0.2313
	0.9	0.2	0.2229	**0.2713**	0.159
	1.0	0.1751	0.2135	**0.2517**	0.1564

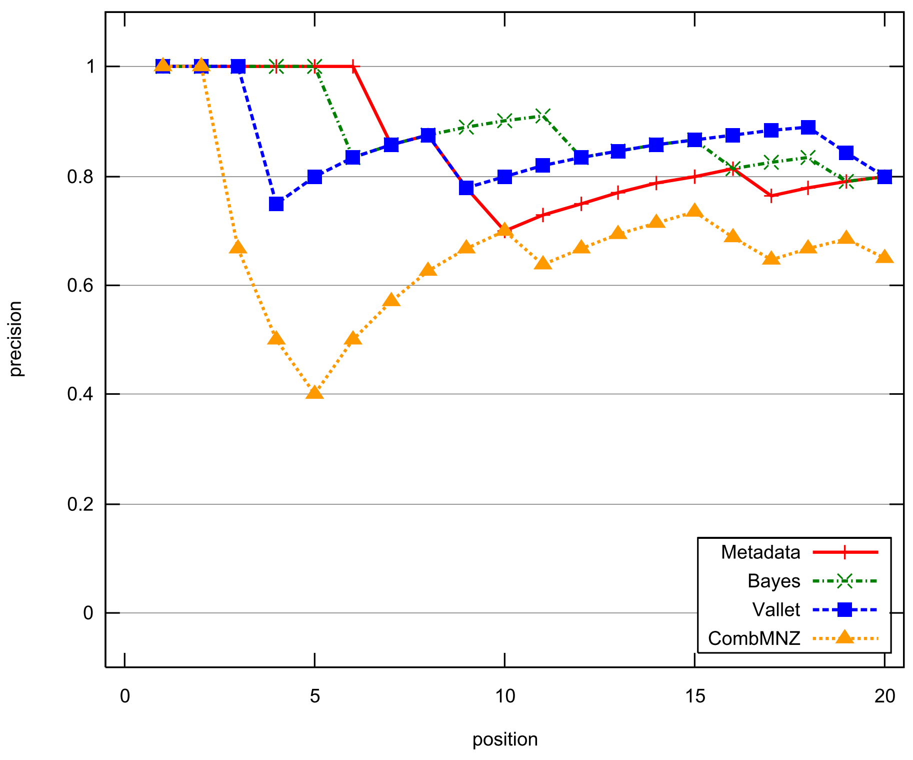


Figure 12.11: WOT metadata and combined semantic search (fuzzy) precision@20 diagram

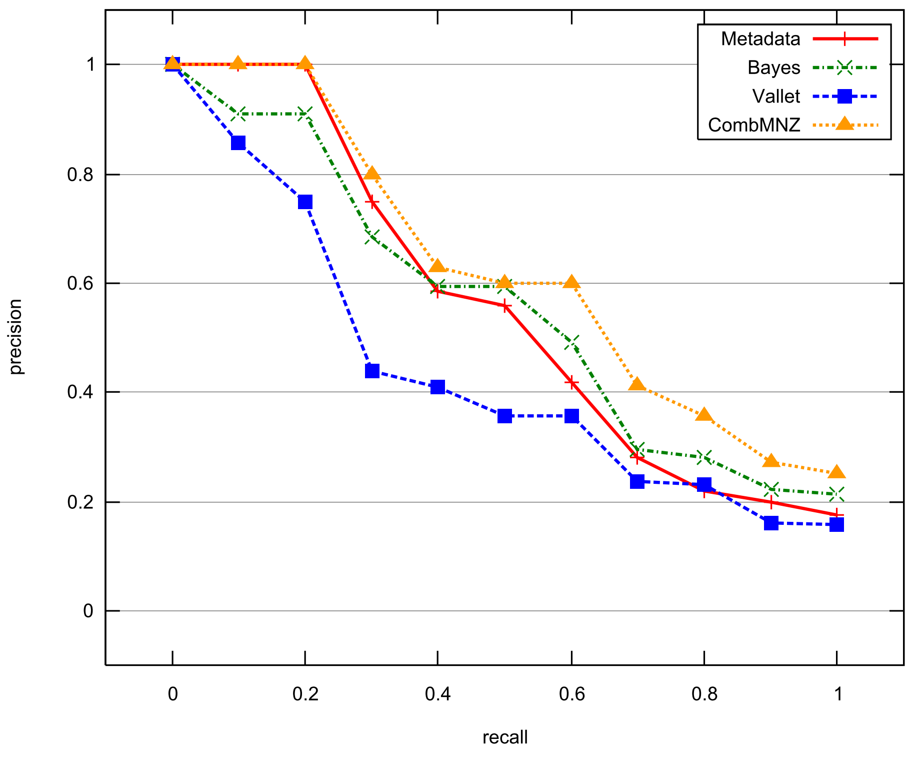


Figure 12.12: Gulf metadata and combined semantic search (fuzzy) precision-recall diagram

Table 12.12: Gulf metadata and combined semantic search (fuzzy) precision@20 values

		Precision			
		Metadata	*Bayes*	*Vallet*	*CombMNZ*
Position	*1*	**1.0**	**1.0**	**1.0**	**1.0**
	2	**1.0**	**1.0**	**1.0**	**1.0**
	3	**1.0**	0.6667	**1.0**	**1.0**
	4	**1.0**	0.75	0.75	**1.0**
	5	**1.0**	0.8	0.8	**1.0**
	6	**1.0**	0.8333	0.8333	**1.0**
	7	**1.0**	0.8571	0.8571	**1.0**
	8	**1.0**	0.875	0.75	**1.0**
	9	**1.0**	0.8889	0.6667	0.8889
	10	**1.0**	0.9	0.7	0.9
	11	**0.9091**	**0.9091**	0.7273	0.8182
	12	**0.8333**	**0.8333**	0.75	0.75
	13	**0.7692**	**0.7692**	0.6923	**0.7692**
	14	0.7143	0.7143	0.6429	**0.7857**
	15	0.7333	0.6667	0.6	**0.8**
	16	**0.75**	0.625	0.5625	**0.75**
	17	**0.7059**	0.6471	0.5294	**0.7059**
	18	0.6667	0.6667	0.5556	**0.7222**
	19	0.6316	**0.6842**	0.5263	**0.6842**
	20	0.6	**0.65**	0.5	**0.65**

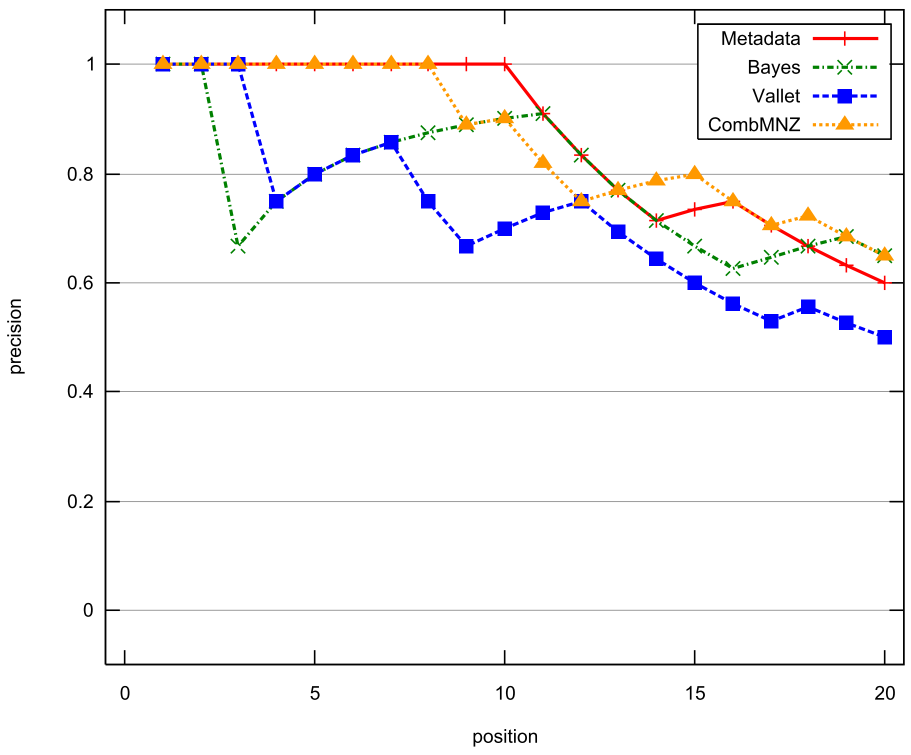


Figure 12.13: Gulf metadata and combined semantic search (fuzzy) precision@20 diagram

Based on these results, a possible conclusion could be that metadata query should be used alone instead of combining it with content search. However, it is important to note that there could be cases where metadata search will not deliver good results. Especially in cases where the ontology does not contain any entities that are relevant to a specific topic, the metadata search will not deliver any results on that topic. It is very difficult to estimate ontology coverage on a specific topic and thus it is difficult to determine whether it makes sense to use result combination for the actual query[14]. Therefore, in the general case, it is still better to use the combined results. Although it will slightly decrease retrieval performance if the ontology covers high-quality information on a specific topic but it will increase effectiveness on the average.

To show that the **H2** hypothesis holds where there is a negative effect of ontology imperfection that should be compensated, I did some another experiments using a different set of semantic metadata. This time, I did not generate semantic metadata for all 1457 documents but I chose only the documents from the list, whose URIs began with “A” to “D”. This selection yielded 364 documents (approx. 25% of the original), whereas from this set 129 documents were relevant for the WOT topic (approx. 23% of all relevants), and 15 documents were relevant for the Gulf topic (approx 40% of all relevants). The results of this experiments are shown in Figures 12.14 to 12.17 and in Tables 12.13 to 12.16.

The combined strategy now performs better than the pure metadata query. I.e., **when the metadata query results are of low quality, query combination can significantly increase the quality of the results**. Interestingly, on the P@20 diagrams, there are still no big differences. This is because the few results that were returned by the metadata search were still relevant. That is, the quality increase caused by the query combination is noticeable only when the metadata search “runs out” of the results. E.g., Figure 12.17 shows that the precision curve of the metadata search result drops steadily, while the combined strategies can also keep a steady precision rate for higher document positions. (Remember, for the Gulf scenario only 15 relevant documents were semantically annotated at all, i.e., the metadata search could return max. 15 documents.)

Interestingly, even in these new experiments there is no clear winner among the combination algorithms, as in the previous case.

To summarize: the **H2** hypothesis generally holds. However, the experiments also showed that if the metadata query provides good results (the ontology has a high quality) then query combination can slightly hurt retrieval performance.

For the following experiments, I used the Bayes strategy to make the scenarios and the discussion simpler. The Bayes strategy seemed to be “in the middle”: although it almost never was the best, it never was the worst.

[14]Although an approach that addresses this problem is a possible future research topic.

Table 12.13: WOT short metadata and combined semantic search precision-recall values

		Precision			
		Metadata	*Bayes*	*CombMNZ*	*Vallet*
Recall	*0.0*	**1.0**	**1.0**	**1.0**	**1.0**
	0.1	0.0	0.3758	**0.396**	0.3758
	0.2	0.0	**0.137**	**0.137**	**0.137**
	0.3	**0.0**	**0.0**	**0.0**	**0.0**
	0.4	**0.0**	**0.0**	**0.0**	**0.0**
	0.5	**0.0**	**0.0**	**0.0**	**0.0**
	0.6	**0.0**	**0.0**	**0.0**	**0.0**
	0.7	**0.0**	**0.0**	**0.0**	**0.0**
	0.8	**0.0**	**0.0**	**0.0**	**0.0**
	0.9	**0.0**	**0.0**	**0.0**	**0.0**
	1.0	**0.0**	**0.0**	**0.0**	**0.0**

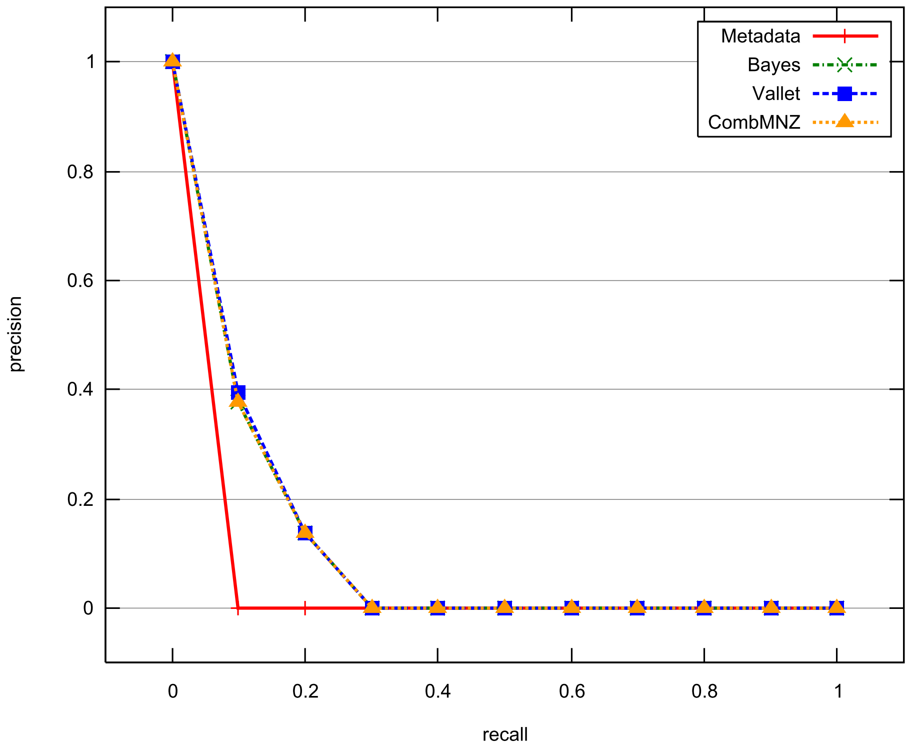


Figure 12.14: WOT short metadata and combined semantic search precision-recall diagram

Table 12.14: WOT short metadata and combined semantic search precision@20 values

		Precision			
		Metadata	*Bayes*	*Vallet*	*CombMNZ*
Position	*1*	**1.0**	**1.0**	**1.0**	**1.0**
	2	**1.0**	**1.0**	**1.0**	**1.0**
	3	**1.0**	**1.0**	**1.0**	**1.0**
	4	**1.0**	**1.0**	**1.0**	0.75
	5	**1.0**	**1.0**	**1.0**	0.8
	6	**1.0**	**1.0**	**1.0**	0.8333
	7	**1.0**	**1.0**	**1.0**	0.8571
	8	**1.0**	**1.0**	0.875	0.75
	9	0.8889	**1.0**	0.7778	0.7778
	10	0.8	**0.9**	0.7	0.8
	11	**0.8182**	**0.8182**	0.7273	**0.8182**
	12	0.75	0.75	0.75	**0.8333**
	13	0.7692	0.6923	0.7692	**0.8462**
	14	0.7143	0.6429	0.7143	**0.8571**
	15	0.7333	0.6	0.6667	**0.8667**
	16	0.75	0.625	0.625	**0.875**
	17	0.7647	0.5882	0.5882	**0.8235**
	18	**0.7778**	0.5556	0.6111	**0.7778**
	19	**0.7895**	0.5263	0.5789	0.7368
	20	**0.75**	0.55	0.55	0.7

Table 12.15: Gulf short metadata and combined semantic search precision-recall values

		Precision			
		Metadata	*Bayes*	*CombMNZ*	*Vallet*
Recall	*0.0*	**1.0**	**1.0**	0.5714	**1.0**
	0.1	0.7778	0.7	0.5714	**0.8333**
	0.2	0.3333	**0.6923**	0.4815	0.6316
	0.3	0.25	**0.6316**	0.4815	**0.6316**
	0.4	0.0	0.4444	0.3333	**0.5294**
	0.5	0.0	0.3231	0.2771	**0.4222**
	0.6	0.0	0.2	**0.2771**	0.2
	0.7	0.0	**0.1688**	**0.1688**	**0.1688**
	0.8	0.0	**0.1088**	**0.1088**	**0.1088**
	0.9	0.0	**0.0515**	**0.0515**	**0.0515**
	1.0	**0.0**	**0.0**	**0.0**	**0.0**

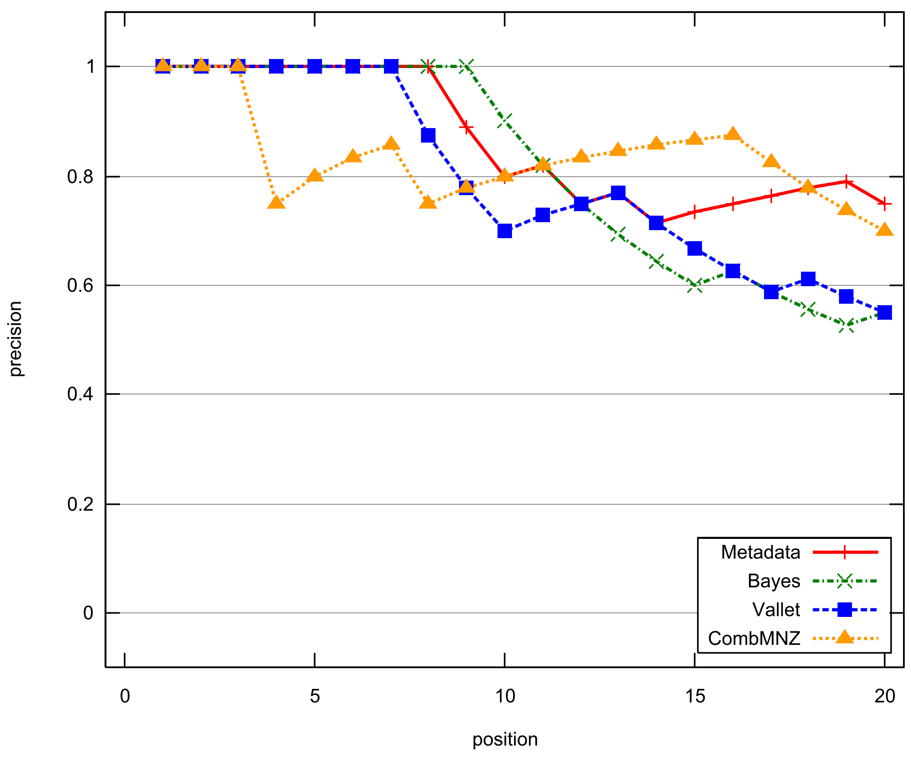


Figure 12.15: WOT short metadata and combined semantic search precision@20 diagram

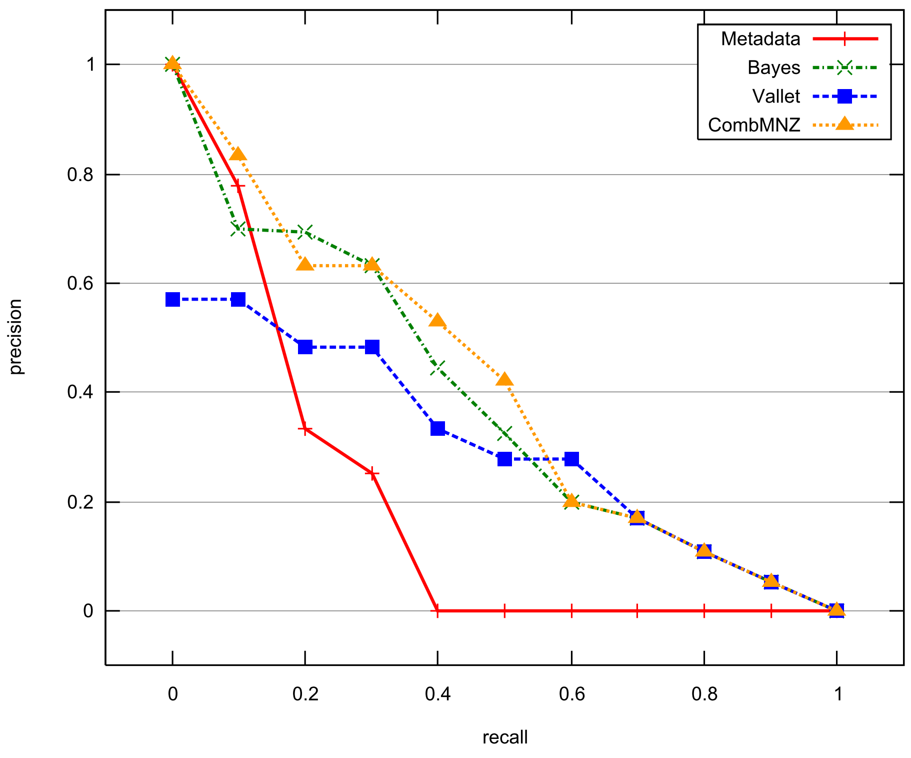


Figure 12.16: Gulf short metadata and combined semantic search precision-recall diagram

Table 12.16: Gulf short metadata and combined semantic search precision@20 values

		Precision			
		Metadata	*Bayes*	*Vallet*	*CombMNZ*
Position	*1*	**1.0**	**1.0**	0.0	**1.0**
	2	0.5	0.5	0.5	**1.0**
	3	**0.6667**	**0.6667**	0.3333	**0.6667**
	4	0.5	0.5	0.5	**0.75**
	5	0.6	0.6	0.4	**0.8**
	6	0.6667	0.5	0.5	**0.8333**
	7	**0.7143**	0.5714	0.5714	**0.7143**
	8	**0.75**	0.625	0.5	**0.75**
	9	**0.7778**	0.6667	0.4444	**0.7778**
	10	**0.7**	**0.7**	0.4	**0.7**
	11	**0.6364**	**0.6364**	0.3636	**0.6364**
	12	0.5833	**0.6667**	0.3333	0.5833
	13	0.5385	**0.6923**	0.3077	0.5385
	14	0.5	**0.6429**	0.2857	0.5714
	15	0.4667	**0.6**	0.3333	0.5333
	16	0.4375	**0.625**	0.375	0.5625
	17	0.4118	**0.6471**	0.3529	0.5882
	18	0.3889	**0.6111**	0.3889	**0.6111**
	19	0.3684	**0.6316**	0.4211	**0.6316**
	20	0.35	**0.6**	0.45	**0.6**

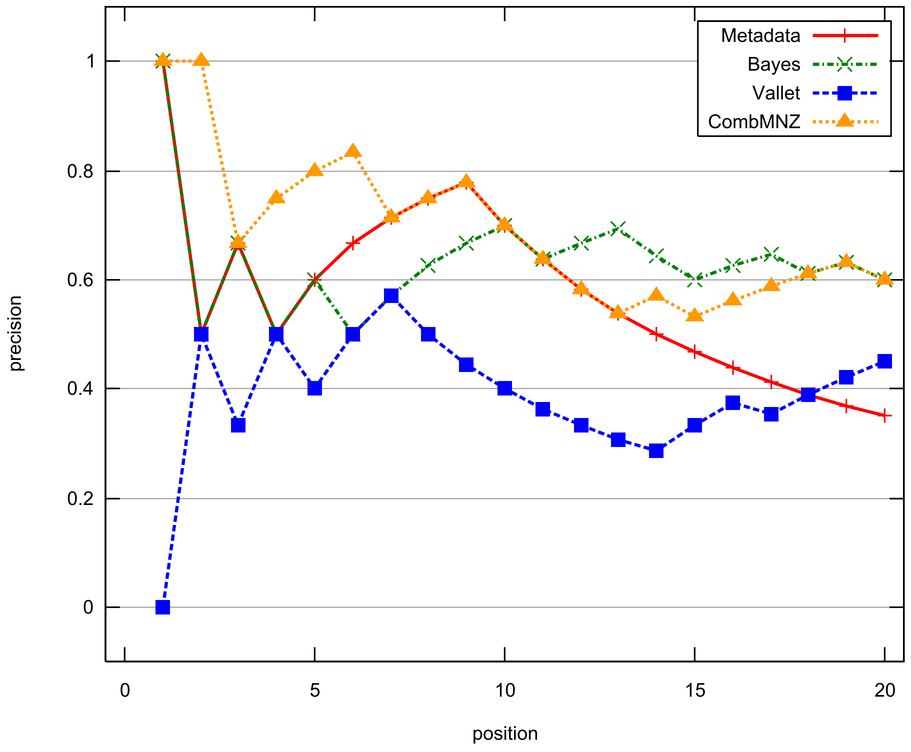


Figure 12.17: Gulf short metadata and combined semantic search precision@20 diagram

12.5.4 The effect of ontology quality

The **H3** hypothesis states that the higher quality the ontology has, the better the retrieval performance of the system is. To evaluate this hypothesis, I analyzed whether ontology quality has some effects on the retrieval performance of the system. To validate the positive effect of a high-quality ontology on search results, we should get better and better results when we switch on various ontology constructs.

The experimental results of comparing search results with different ontology features enabled are shown in Figures 12.18 to 12.25 and in Tables 12.17 to 12.24.

In the WOT scenario, the positive effect of a high-quality ontology can be clearly observed. There is an especially big quality increase when the relations between ontology instances are used. When time information is switched on, only a marginal quality increase can be observed whereas using fuzzy time made no practical difference.

The effect of ontology quality can also be observed in the Gulf scenario, although the results are not as positive as in the other scenario. When non-linguistic relations are switched on, retrieval performance actually drops drastically. This happens because on the one hand, there is practically no room for improvement for semantic search (only two out of 38 relevant documents are not found by traditional FTS); but on the other hand, there are many possible wrong inferences because of the overlap between the WOT and Gulf scenarios. Indeed, without time information, the “Gulf War” concept is wrongly inferred as relevant for many WOT articles. This causes the observed drop in retrieval efficiency.

This scenario also shows, however, that adding time information can solve this problem. After adding time information to the ontology, retrieval performance is restored and for some recall levels it is even increased. Switching on fuzzy time made no practical difference in this scenario, either.

The results clearly validate the hypothesis that ontology quality has a significant effect on retrieval performance. Semantic relations increase recall significantly in cases where FTS does not deliver adequate results. The potential negative effect of incorrectly applying these semantic relations, which could eventually decrease precision, can be eliminated by including temporal information into the ontology which guides the inference process.

To summarize: the **H3** hypothesis could be partially validated. Although there is a general correlation between ontology quality and retrieval results, there are some ontology features that actually have a negative effect in some scenarios. However, **an ontology using all of the features always provides the best results**, even if in some scenarios these best results can also be reached by using fewer features.

Table 12.17: Effect of ontology quality on WOT metadata search precision-recall values

Recall		**Precision**			
		no connection	*no time*	*normal time*	*fuzzy time*
	0.0	0.7	0.875	**1.0**	**1.0**
	0.1	0.3709	0.6867	**0.725**	**0.725**
	0.2	0.0	0.5989	**0.6505**	**0.6505**
	0.3	0.0	0.541	**0.6154**	0.6145
	0.4	0.0	0.5078	0.5398	**0.549**
	0.5	**0.0**	**0.0**	**0.0**	**0.0**
	0.6	**0.0**	**0.0**	**0.0**	**0.0**
	0.7	**0.0**	**0.0**	**0.0**	**0.0**
	0.8	**0.0**	**0.0**	**0.0**	**0.0**
	0.9	**0.0**	**0.0**	**0.0**	**0.0**
	1.0	**0.0**	**0.0**	**0.0**	**0.0**

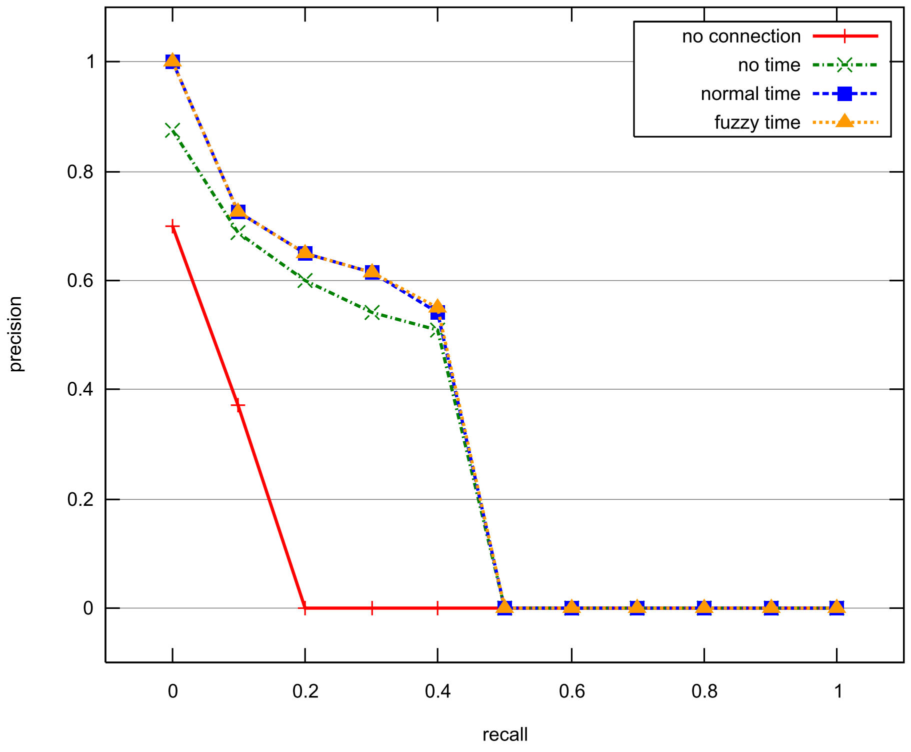


Figure 12.18: Effect of ontology quality on WOT metadata search precision-recall diagram

Table 12.18: Effect of ontology quality on WOT metadata search precision@20 values

		Precision			
		no connection	*no time*	*normal time*	*fuzzy time*
Position	*1*	0.0	0.0	**1.0**	**1.0**
	2	0.5	0.5	**1.0**	**1.0**
	3	0.6667	0.6667	**1.0**	**1.0**
	4	0.5	0.75	**1.0**	**1.0**
	5	0.6	0.8	**1.0**	**1.0**
	6	0.5	0.8333	**1.0**	**1.0**
	7	0.5714	**0.8571**	**0.8571**	**0.8571**
	8	0.625	**0.875**	**0.875**	**0.875**
	9	0.5556	**0.7778**	**0.7778**	**0.7778**
	10	0.6	**0.7**	**0.7**	**0.7**
	11	0.6364	0.6364	**0.7273**	**0.7273**
	12	0.6667	0.6667	**0.75**	**0.75**
	13	0.6923	0.6154	**0.7692**	**0.7692**
	14	0.6429	0.5714	**0.7857**	**0.7857**
	15	0.6667	0.6	**0.8**	**0.8**
	16	0.625	0.625	**0.8125**	**0.8125**
	17	0.6471	0.6471	**0.7647**	**0.7647**
	18	0.6667	0.6667	**0.7778**	**0.7778**
	19	0.6842	0.6842	**0.7895**	**0.7895**
	20	0.7	0.7	**0.8**	**0.8**

Table 12.19: Effect of ontology quality on WOT combined search precision-recall values

		Precision			
		no connection	*no time*	*normal time*	*fuzzy time*
Recall	*0.0*	**1.0**	**1.0**	**1.0**	**1.0**
	0.1	0.3353	0.7125	**0.7273**	**0.7273**
	0.2	0.0	0.5825	**0.6348**	**0.6348**
	0.3	0.0	0.5385	0.5951	**0.5965**
	0.4	0.0	0.5	0.5234	**0.5271**
	0.5	0.0	**0.4475**	0.302	0.4409
	0.6	**0.0**	**0.0**	**0.0**	**0.0**
	0.7	**0.0**	**0.0**	**0.0**	**0.0**
	0.8	**0.0**	**0.0**	**0.0**	**0.0**
	0.9	**0.0**	**0.0**	**0.0**	**0.0**
	1.0	**0.0**	**0.0**	**0.0**	**0.0**

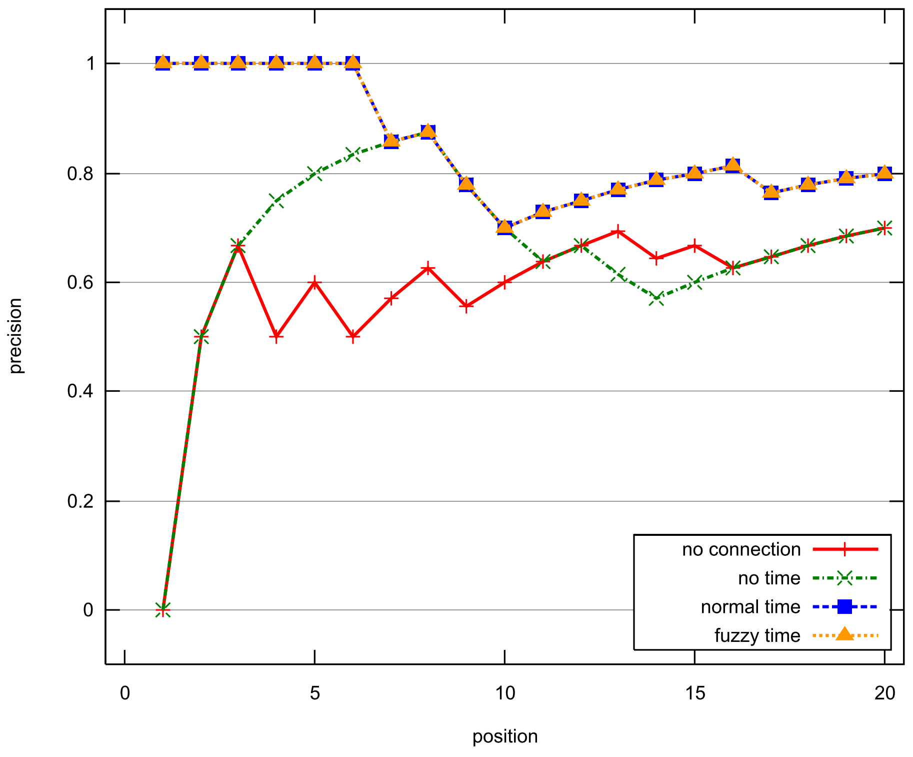


Figure 12.19: Effect of ontology quality on WOT metadata search precision@20 diagram

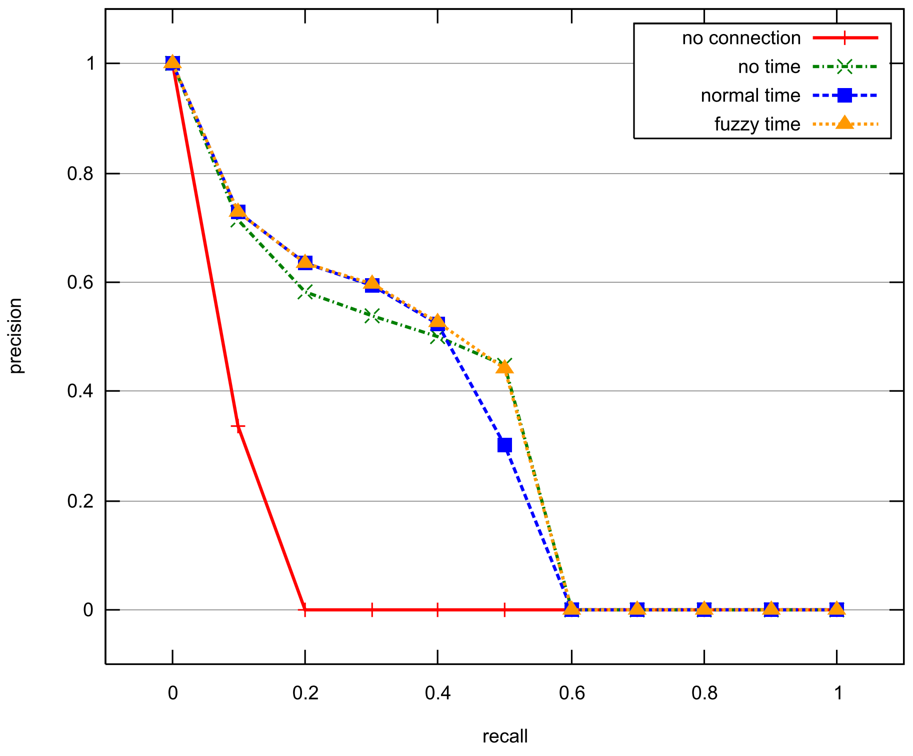


Figure 12.20: Effect of ontology quality on WOT combined search precision-recall diagram

Table 12.20: Effect of ontology quality on WOT combined search precision@20 values

		Precision			
		no connection	*no time*	*normal time*	*fuzzy time*
Position	*1*	**1.0**	**1.0**	**1.0**	**1.0**
	2	0.5	**1.0**	**1.0**	**1.0**
	3	0.6667	**1.0**	**1.0**	**1.0**
	4	0.5	**1.0**	**1.0**	**1.0**
	5	0.6	**1.0**	**1.0**	**1.0**
	6	0.5	**0.8333**	**0.8333**	**0.8333**
	7	0.4286	**0.8571**	**0.8571**	**0.8571**
	8	0.375	**0.875**	**0.875**	**0.875**
	9	0.3333	0.7778	**0.8889**	**0.8889**
	10	0.3	0.8	**0.9**	**0.9**
	11	0.2727	0.8182	**0.9091**	**0.9091**
	12	0.25	**0.8333**	**0.8333**	**0.8333**
	13	0.2308	0.7692	**0.8462**	**0.8462**
	14	0.2143	0.7857	**0.8571**	**0.8571**
	15	0.2667	0.8	**0.8667**	**0.8667**
	16	0.3125	**0.8125**	**0.8125**	**0.8125**
	17	0.2941	0.7647	**0.8235**	**0.8235**
	18	0.3333	0.7778	**0.8333**	**0.8333**
	19	0.3684	**0.7895**	**0.7895**	**0.7895**
	20	0.4	0.75	**0.8**	**0.8**

Table 12.21: Effect of ontology quality on Gulf metadata search precision-recall values

		Recall			
		no connection	*no time*	*normal time*	*fuzzy time*
Precision	*0.0*	**1.0**	0.8889	**1.0**	**1.0**
	0.1	**1.0**	0.8889	**1.0**	**1.0**
	0.2	0.8	0.8889	**1.0**	**1.0**
	0.3	0.5909	0.3	**0.75**	**0.75**
	0.4	0.5517	0.2208	**0.5862**	**0.5862**
	0.5	0.5263	0.1979	**0.5588**	**0.5588**
	0.6	0.36	0.1264	0.4107	**0.4182**
	0.7	**0.36**	0.0925	0.2872	0.2813
	0.8	**0.2981**	0.0891	0.2185	0.22
	0.9	**0.2035**	0.0891	0.2	0.2
	1.0	0.0	0.0878	**0.1751**	**0.1751**

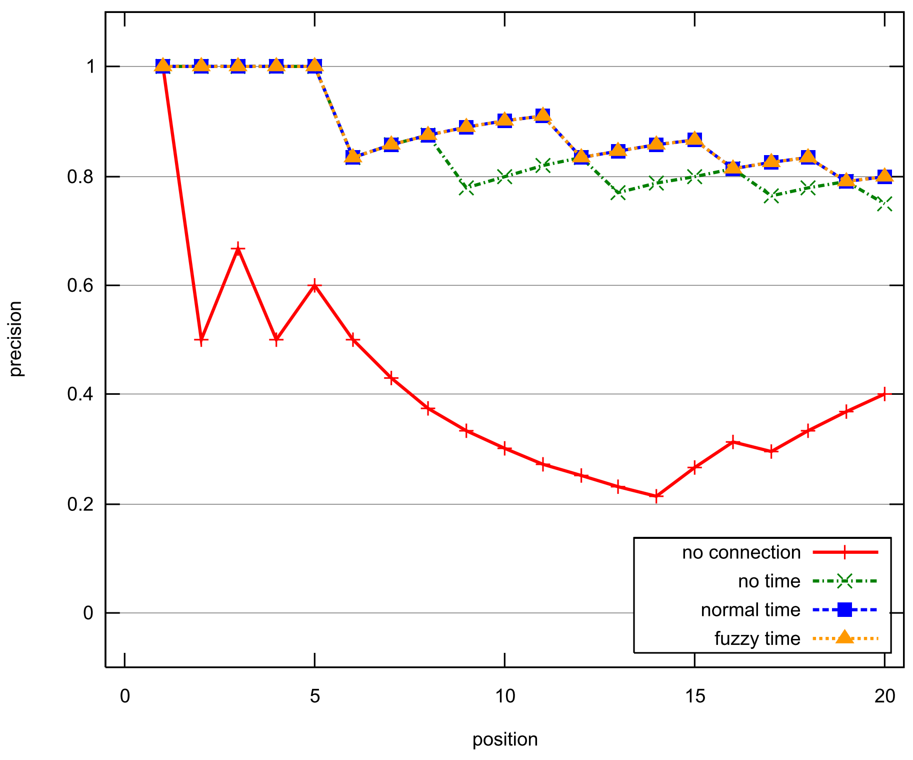


Figure 12.21: Effect of ontology quality on WOT combined search precision@20 diagram

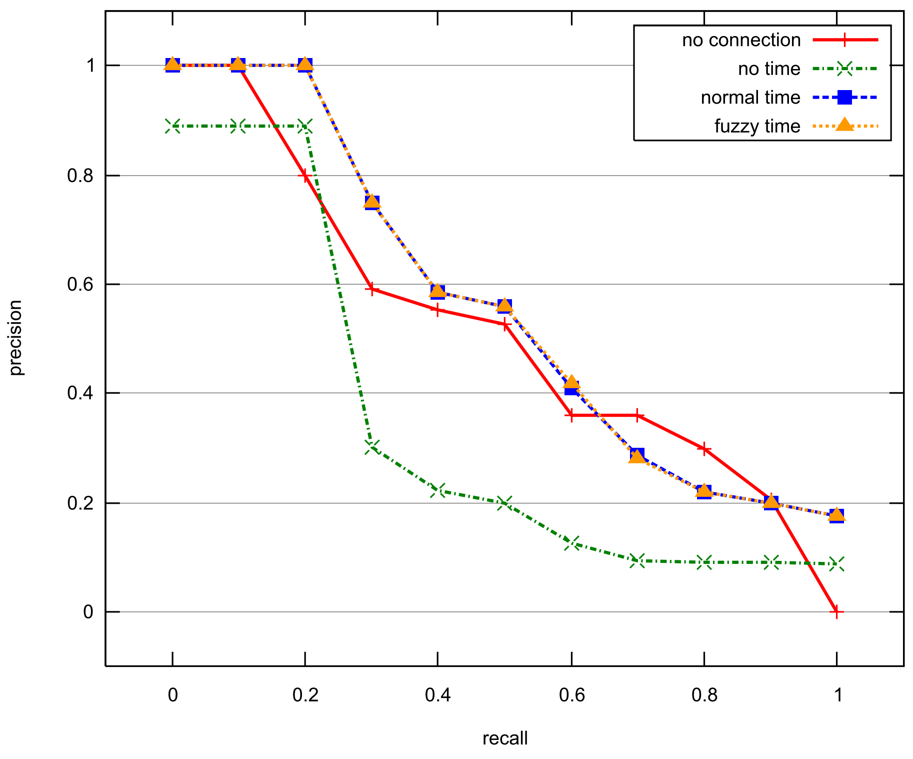


Figure 12.22: Effect of ontology quality on Gulf metadata search precision-recall diagram

Table 12.22: Effect of ontology quality on Gulf metadata search precision@20 values

		Precision			
		no connection	*no time*	*normal time*	*fuzzy time*
Position	*1*	**1.0**	0.0	**1.0**	**1.0**
	2	**1.0**	0.5	**1.0**	**1.0**
	3	**1.0**	0.6667	**1.0**	**1.0**
	4	**1.0**	0.75	**1.0**	**1.0**
	5	**1.0**	0.8	**1.0**	**1.0**
	6	**1.0**	0.8333	**1.0**	**1.0**
	7	**1.0**	0.8571	**1.0**	**1.0**
	8	0.875	0.875	**1.0**	**1.0**
	9	0.7778	0.8889	**1.0**	**1.0**
	10	0.8	0.8	0.9	**1.0**
	11	0.7273	0.8182	0.8182	**0.9091**
	12	0.6667	**0.8333**	**0.8333**	**0.8333**
	13	0.6923	**0.7692**	**0.7692**	**0.7692**
	14	**0.7143**	**0.7143**	**0.7143**	**0.7143**
	15	**0.7333**	0.6667	**0.7333**	**0.7333**
	16	0.6875	0.625	**0.75**	**0.75**
	17	0.6471	0.5882	**0.7059**	**0.7059**
	18	0.6111	0.5556	**0.6667**	**0.6667**
	19	0.5789	0.5263	**0.6316**	**0.6316**
	20	0.55	0.5	**0.6**	**0.6**

Table 12.23: Effect of ontology quality on Gulf combined search precision-recall values

		Precision			
		no connection	*no time*	*normal time*	*fuzzy time*
Recall	*0.0*	**1.0**	**1.0**	**1.0**	**1.0**
	0.1	0.875	0.8333	**0.9091**	**0.9091**
	0.2	0.7273	0.8333	**0.9091**	**0.9091**
	0.3	0.5909	0.4	**0.6842**	**0.6842**
	0.4	0.4865	0.2651	**0.5938**	**0.5938**
	0.5	0.4545	0.2651	**0.5938**	**0.5938**
	0.6	0.4118	0.1912	**0.4902**	**0.4902**
	0.7	**0.4118**	0.125	0.3077	0.2947
	0.8	**0.3735**	0.1056	0.287	0.2793
	0.9	**0.3125**	0.0931	0.2229	0.2229
	1.0	0.0	0.0931	**0.2135**	**0.2135**

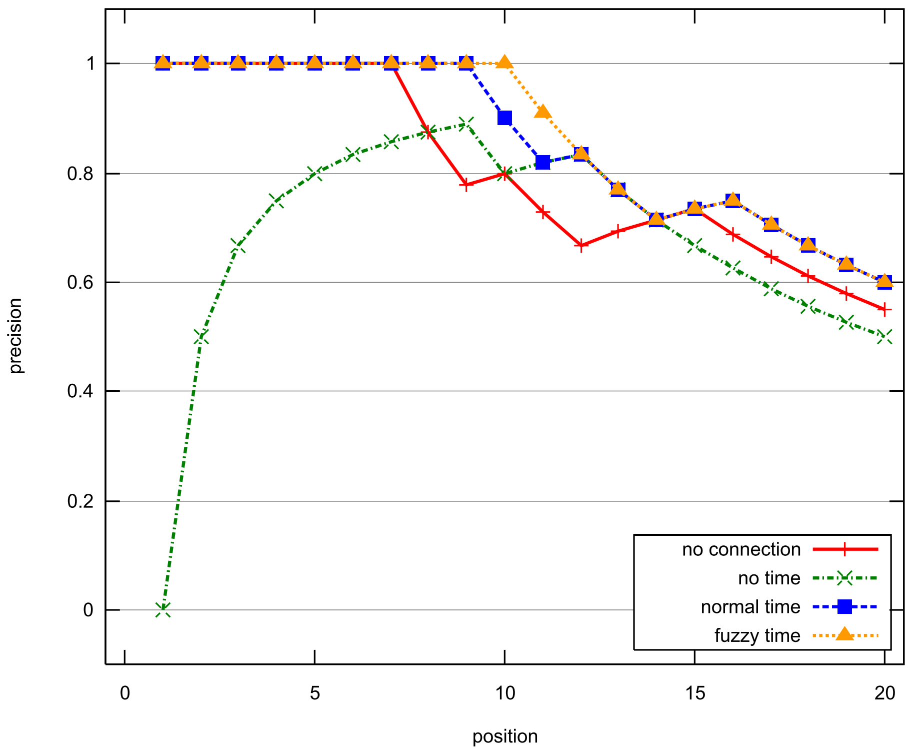


Figure 12.23: Effect of ontology quality on Gulf metadata search precision@20 diagram

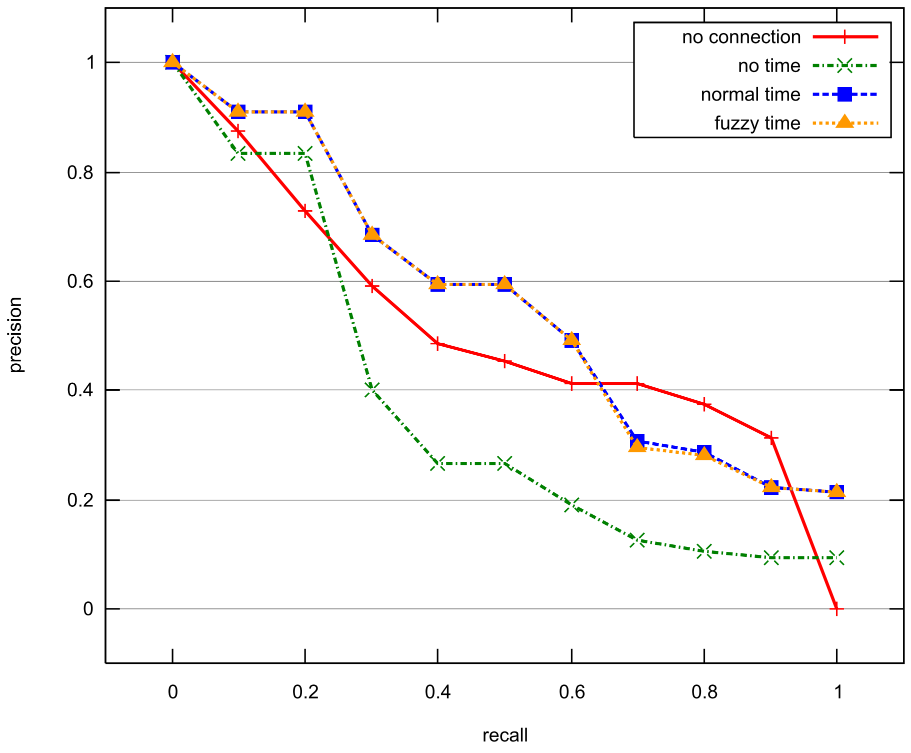


Figure 12.24: Effect of ontology quality on Gulf combined search precision-recall diagram

Table 12.24: Effect of ontology quality on Gulf combined search precision@20 values

		Precision			
		no connection	*no time*	*normal time*	*fuzzy time*
Position	*1*	**1.0**	**1.0**	**1.0**	**1.0**
	2	**1.0**	**1.0**	**1.0**	**1.0**
	3	**0.6667**	**0.6667**	**0.6667**	**0.6667**
	4	**0.75**	**0.75**	**0.75**	**0.75**
	5	**0.8**	**0.8**	**0.8**	**0.8**
	6	**0.8333**	**0.8333**	**0.8333**	**0.8333**
	7	**0.8571**	0.7143	**0.8571**	**0.8571**
	8	**0.875**	0.75	**0.875**	**0.875**
	9	0.7778	0.7778	**0.8889**	**0.8889**
	10	0.7	0.8	**0.9**	**0.9**
	11	0.7273	0.8182	**0.9091**	**0.9091**
	12	0.6667	**0.8333**	**0.8333**	**0.8333**
	13	0.6923	**0.7692**	**0.7692**	**0.7692**
	14	0.6429	**0.7143**	**0.7143**	**0.7143**
	15	**0.6667**	**0.6667**	**0.6667**	**0.6667**
	16	**0.625**	**0.625**	**0.625**	**0.625**
	17	0.5882	0.5882	**0.6471**	**0.6471**
	18	0.5556	0.5556	**0.6667**	**0.6667**
	19	0.5263	0.5263	**0.6842**	**0.6842**
	20	0.55	0.5	**0.65**	**0.65**

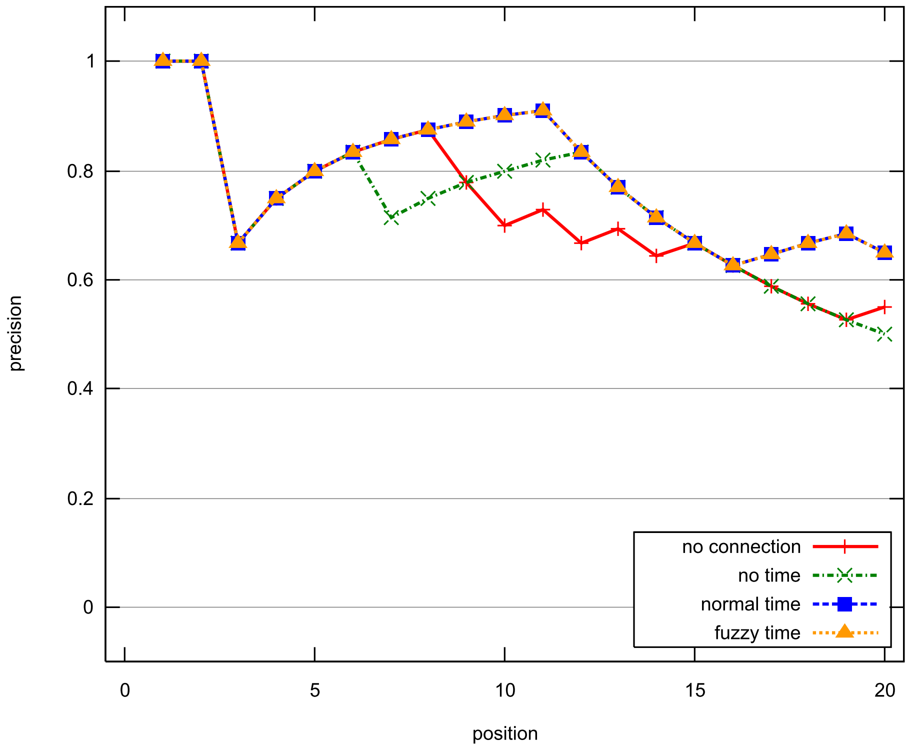


Figure 12.25: Effect of ontology quality on Gulf combined search precision@20 diagram

12.5.5 Comparing combined search with the baseline

To evaluate my **Main** hypothesis, i.e., that possibly imperfect ontologies improve the retrieval performance of an information system, I checked whether the results of my system are superior to the results of traditional full-text search. That is, I compared the combined search results with the baseline FTS strategies. The experimental results are shown in Figures 12.26 to 12.29 and in Tables 12.25 to 12.28.

I compared the worst-case and best-case performance (based on the ontology-quality experiments) of my system with the baseline. Based on the previous results, the worst case for the WOT scenario was using the ontology with no connections, whereas for the Gulf scenario, the worst case was using the ontology with connections but without time information.

The results show that in the best case, semantic search[15] is superior to any of the FTS strategies in any scenarios. Especially in the WOT scenario, the results are really impressive. Also in the Gulf scenario a very significant retrieval performance gain of 100% can be observed for some recall levels. The Precision@20 results are significantly better, too. I.e., the gain in efficiency is not only theoretical but it should be also noticeable for the non-expert end users.

The results also show that semantic search is not superior, or only slightly superior to traditional FTS in the worst case. As semantic search always requires significant investment for building the ontology and the heuristic rules, the problem that was observed in traditional IR for thesauri also exists for ontologies. Ontologies and semantic technologies do not automatically increase IR effectiveness, the quality of ontologies is crucial. Using all ontology features, however, guarantees good results in all scenarios (see results of the last section).

In other words: **my evaluation study showed that even casually developed, imperfect ontologies can be "good enough" to provide very significant improvements in IR performance.** Therefore the **main hypothesis could be successfully evaluated**. This positive result provides a strong motivation for the investment into the application of ontologies in information systems[16].

[15]As was discussed in Section 5.4, I term the combination of the metadata and content searches as semantic search.

[16]In addition to improving search results, ontologies and semantic metadata have other positive "side effects", such as the ability to make query refinement suggestions or navigation among ontology entities.

Table 12.25: Comparison of WOT full-text and semantic search (precision-recall values)

Recall	**Precision** *FTS OR*	*FTS AND*	*FTS quoted*	*Bayes (worst)*	*Bayes (best)*
0.0	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
0.1	0.0080	0.0694	0.1054	0.3353	**0.7273**
0.2	0.0080	0.0532	0.0	0.0	**0.6348**
0.3	0.0080	0.0429	0.0	0.0	**0.5965**
0.4	0.0059	0.0	0.0	0.0	**0.5271**
0.5	0.0035	0.0	0.0	0.0	**0.4409**
0.6	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
0.7	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
0.8	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
0.9	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
1.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

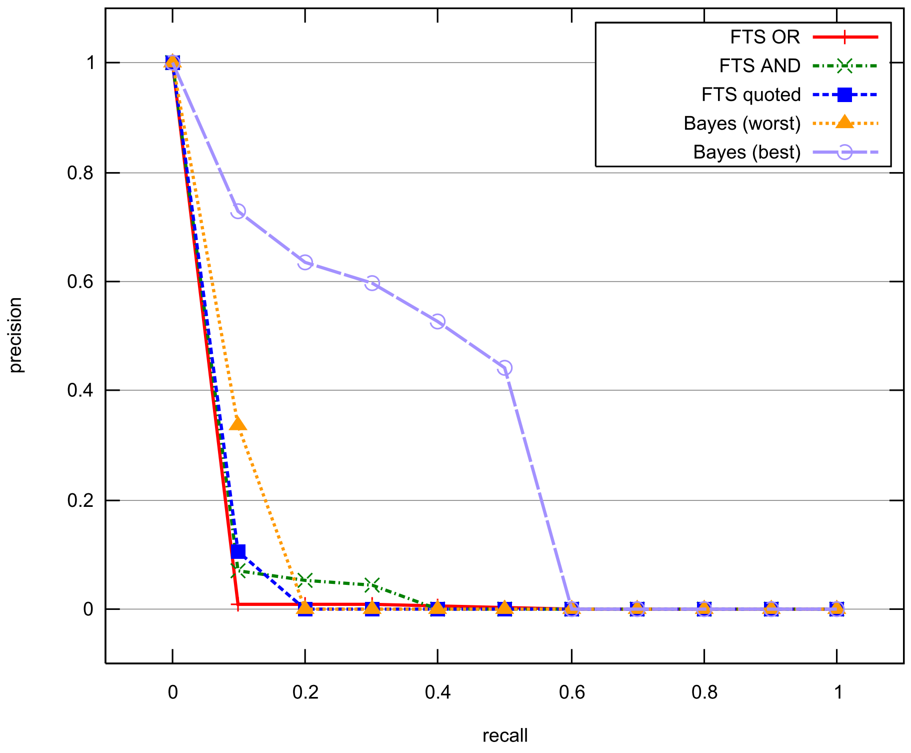


Figure 12.26: Comparison of WOT full-text and semantic search (precision-recall diagram)

Table 12.26: Comparison of WOT full-text and semantic search (precision@20 values)

		Precision		
		FTS quoted	*Bayes (worst)*	*Bayes (best)*
Position	*1*	**1.0**	**1.0**	**1.0**
	2	**1.0**	0.5	**1.0**
	3	0.6667	0.6667	**1.0**
	4	0.5	0.5	**1.0**
	5	0.4	0.6	**1.0**
	6	0.3333	0.5	**0.8333**
	7	0.4286	0.4286	**0.8571**
	8	0.375	0.375	**0.875**
	9	0.3333	0.3333	**0.8889**
	10	0.3	0.3	**0.9**
	11	0.2727	0.2727	**0.9091**
	12	0.25	0.25	**0.8333**
	13	0.3077	0.2308	**0.8462**
	14	0.2857	0.2143	**0.8571**
	15	0.2667	0.2667	**0.8667**
	16	0.25	0.3125	**0.8125**
	17	0.2353	0.2941	**0.8235**
	18	0.2222	0.3333	**0.8333**
	19	0.2105	0.3684	**0.7895**
	20	0.25	0.4	**0.8**

Table 12.27: Comparison of Gulf full-text and semantic search (precision-recall values)

		Precision			
		FTS proximity	*FTS quoted*	*Bayes (worst)*	*Bayes (best)*
Recall	*0.0*	**1.0**	**1.0**	**1.0**	**1.0**
	0.1	0.625	0.625	0.8333	**0.9091**
	0.2	0.2703	0.2778	0.8333	**0.9091**
	0.3	0.2571	0.2609	0.4	**0.6842**
	0.4	0.2571	0.2609	0.2651	**0.5938**
	0.5	0.25	0.2532	0.2651	**0.5938**
	0.6	0.2072	0.097	0.1912	**0.4902**
	0.7	0.0549	0.0545	0.125	**0.2947**
	0.8	0.0473	0.0436	0.1056	**0.2793**
	0.9	0.035	0.0362	0.0931	**0.2229**
	1.0	0.0	0.0	0.0931	**0.2135**

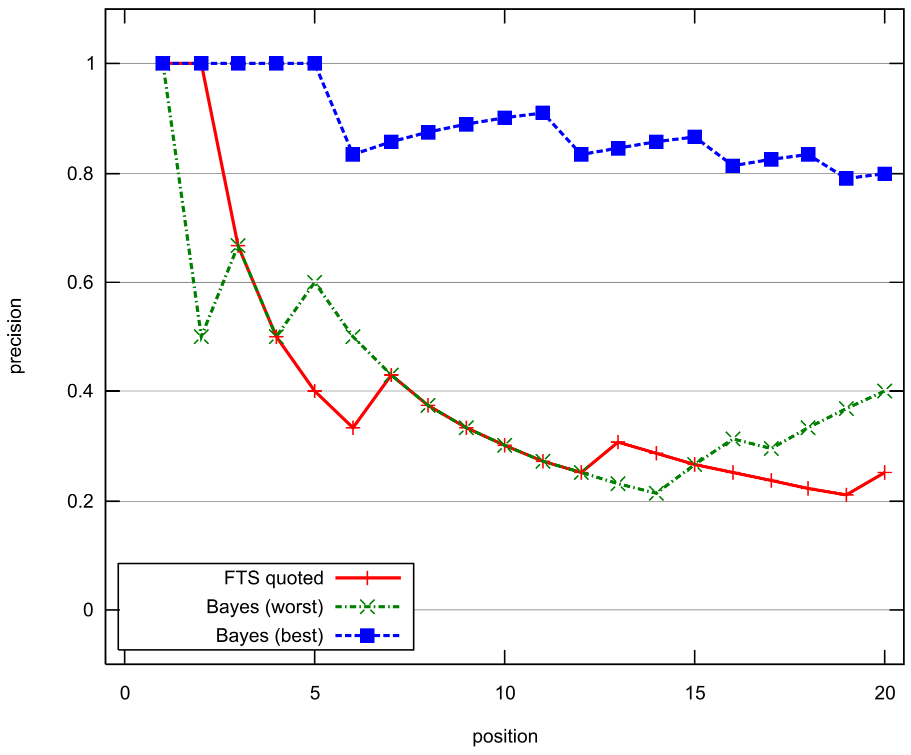


Figure 12.27: Comparison of WOT full-text and semantic search (precision@20 diagram)

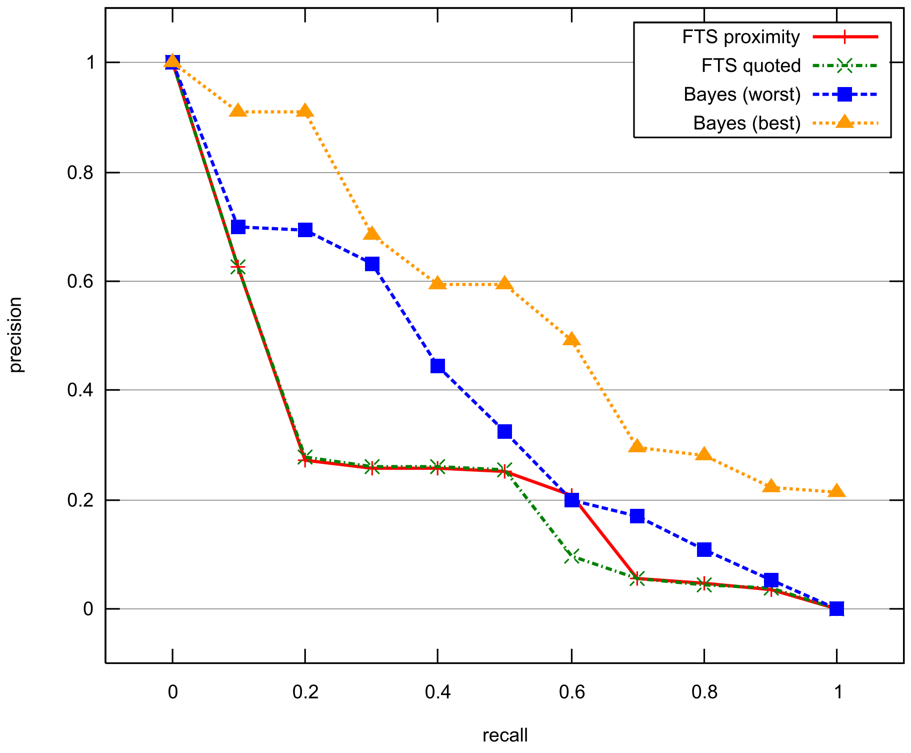


Figure 12.28: Comparison of Gulf full-text and semantic search (precision-recall diagram)

Table 12.28: Comparison of Gulf full-text and semantic search (precision@20 values)

		Precision		
		FTS OR	*Bayes (worst)*	*Bayes (best)*
Position	*1*	**1.0**	**1.0**	**1.0**
	2	0.5	**1.0**	**1.0**
	3	0.3333	**0.6667**	**0.6667**
	4	0.5	**0.75**	**0.75**
	5	0.6	**0.8**	**0.8**
	6	0.5	**0.8333**	**0.8333**
	7	0.5714	0.7143	**0.8571**
	8	0.625	0.75	**0.875**
	9	0.5556	0.7778	**0.8889**
	10	0.5	0.8	**0.9**
	11	0.4545	0.8182	**0.9091**
	12	0.5	**0.8333**	**0.8333**
	13	0.4615	**0.7692**	**0.7692**
	14	0.4286	**0.7143**	**0.7143**
	15	0.4	**0.6667**	**0.6667**
	16	0.375	**0.625**	**0.625**
	17	0.3529	0.5882	**0.6471**
	18	0.3333	0.5556	**0.6667**
	19	0.3158	0.5263	**0.6842**
	20	0.3	0.5	**0.65**

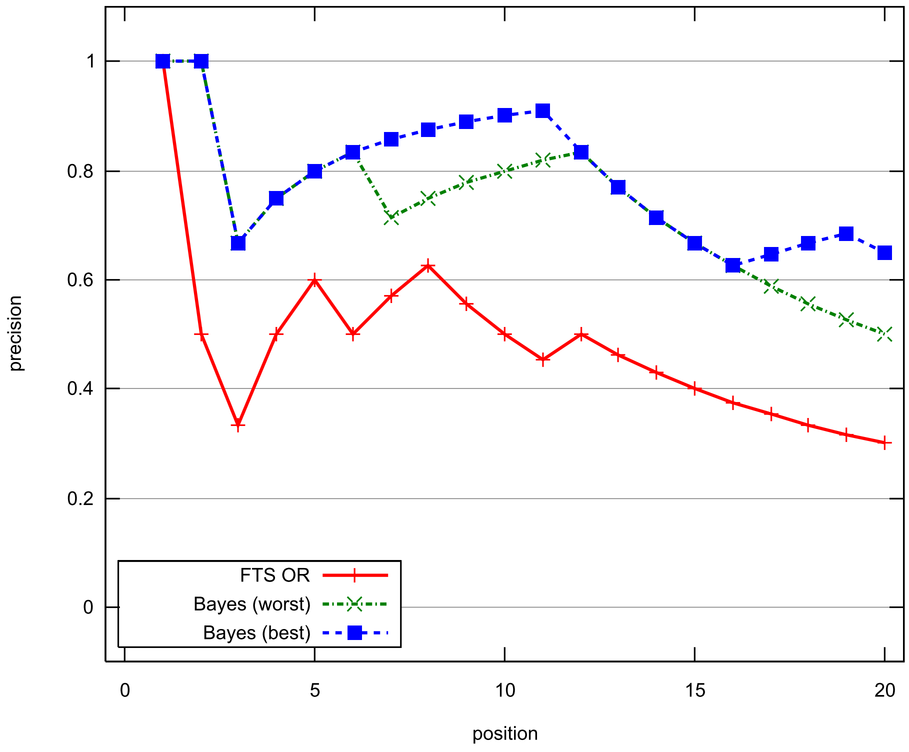


Figure 12.29: Comparison of Gulf full-text and semantic search (precision@20 diagram)

12.5.6 The use case for the fuzzy time model

The previous results showed that fuzzyfying the temporal model does not help increase retrieval effectiveness[17] if our goal is to find documents about an abstract concept. This does not mean, however, that the fuzzy time model cannot be used to improve other aspects of IR. Documents that contain fuzzy time definitions, or user queries that contain fuzzy time specifications can profit from the model. E.g., in the set of 1457 annotated documents there were 485 fuzzy time expressions that could be parsed during the semantic annotation.

The generated time information can be used to significantly improve the results of queries that refer to time. E.g., Table 12.29 and Figure 12.30 show the comparison of the results of the `Iraq the early 2000s` (FTS OR) and the `Iraq AND "the early 2000s"` (FTS AND) full-text queries, and of the `Iraq the early 2000s` semantic query, using the Bayes combination method (Bayes)[18]. It can be clearly seen that the semantic results are superior to the FTS results.

This improvement in retrieval quality was possible in the case of the semantic query because time information in the documents and also in the user query was parsed and explicitly represented. The FTS_AND full-text search could match only documents that explicitly contained the `"the early 2000s"`. As one could intuitively expect, only a small portion of relevant documents contained this exact phrase. Finally, the FTS_OR query returned many irrelevant results because it also returned documents that contained the terms “2000s” or “early” in isolation.

[17]But it does not decrease retrieval effectiveness, either.

[18]It is not possible to create a PR diagram because the set of all relevant documents is not known for such a complex query. The (probably partial) set of relevant documents were manually determined by examining the first 20 results of all of the three queries.

Table 12.29: Comparison of “Iraq the early 2000s” full-text and semantic search (precision@20 values)

		Precision		
		FTS OR	*FTS AND*	*Bayes*
Position	*1*	0.0	0.0	**1.0**
	2	0.5	0.5	**1.0**
	3	0.6667	0.3333	**1.0**
	4	0.75	0.25	**1.0**
	5	0.6	0.2	**0.8**
	6	0.6667	0.1667	**0.8333**
	7	0.5714	0.1429	**0.8571**
	8	0.5	0.125	**0.875**
	9	0.4444	0.1111	**0.8889**
	10	0.4	0.1	**0.9**
	11	0.3636	0.0909	**0.9091**
	12	0.3333	0.0833	**0.9167**
	13	0.3846	0.0769	**0.9231**
	14	0.3571	0.0714	**0.9286**
	15	0.4	0.0667	**0.9333**
	16	0.375	0.0625	**0.875**
	17	0.4118	0.0588	**0.8824**
	18	0.3889	0.0556	**0.8333**
	19	0.4211	0.0526	**0.8421**
	20	0.45	0.05	**0.85**

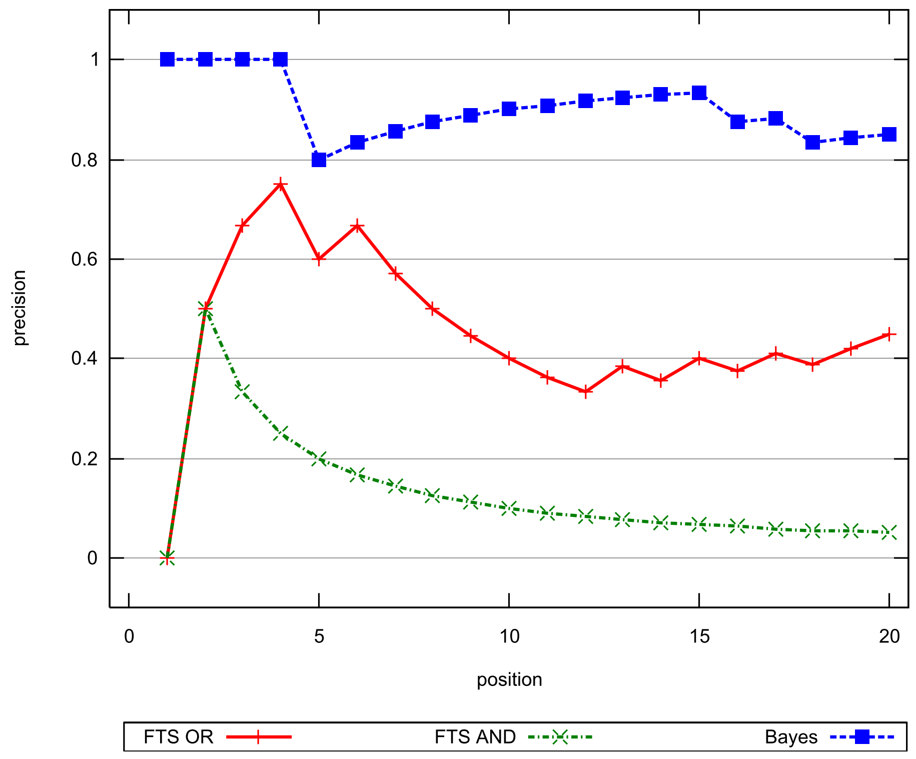


Figure 12.30: Comparison of “Iraq the early 2000s” full-text and semantic search (precision@20 diagram)

12.5.7 Evaluating response times

As the final experiment, I validated the **H4** hypothesis which states that it is possible to use advanced ontology constructs and provide an interactive system at the same time. In doing so, I measured the response times of the system in the WOT and Gulf scenarios. For the WOT scenario, the average response time was 0.9063 seconds, and for the Gulf scenario, the average response time was 2.3957 seconds[19]. **These results clearly show that the system meets the SCALABILITY requirement** because response times are far below 10 seconds. Therefore, the **H4** hypothesis could be successfully evaluated.

There is a significant difference, however, between the query times in the two scenarios. To check the causes, I measured the query time for the specific tasks during the query process. The results are shown in Table 12.30. The full semantic time includes the times for parsing the full-text query into its semantic representation, executing the metadata query, loading the metadata query results, and combining it with the content query results. The complete full-text time (full FTS) includes the time needed to execute the content query and to load the content results for query combination. The miscellaneous part include the processing overhead of calling various Java methods and also the generation of the metadata and content queries from the semantic query representation.

Table 12.30: Query response times (in seconds)

	WOT	**Gulf**
Query parsing	0.146	0.047
Metadata search	0.0103	0.0053
Loading metadata results	0.052	0.026
FTS (content search)	0.2136	0.4737
Loading FTS results	0.349	1.6927
Combining results	0.0053	0.037
Miscellaneous	0.13	0.114
Full query	**0.9063**	**2.3957**
Full semantic	0.2136	0.1153
Full FTS	0.5627	2.1664

The results show some interesting things. First, it is easy to see that the highest fraction of time is needed for executing the traditional full-text search over document content, and loading these search results. This means that the overhead caused by the semantic search part is acceptable in practice.

Second, in the Gulf scenario it takes much more time to execute the full-text search than in the WOT scenario. This is caused by two factors: first, the FTS query generated for the Gulf scenario is much more complicated than for the WOT scenario. The query contains more query clauses because the instance representing the Gulf War event in the ontology has dramatically

[19]Both values were calculated by averaging the results of three measurements using a P4 2.8 GHz PC with 1GB RAM running Windows XP Professional.

more synonyms than the War on Terrorism event. Naturally, the Lucene-based full-text search engine needs more time to execute a more complicated query.

The second factor that makes the Gulf War query slower is the number of results returned by the metadata and content queries. For the Gulf War case there were 1710 FTS results while for the War on Terrorism case there were only 1248 FTS results. Since all subquery results have to be processed during query combination, more subquery results clearly slow down the whole query process.

Knowing the causes for the additional query time needed for the Gulf scenario, one could easily imagine a scenario where the system would not meet the SCALABILITY requirement anymore. E.g., when the user navigates from a document to other similar documents and the query is generated automatically (as described in Chapter 11). In this case, the query contains many elements and it can be also expected that such generic queries will return many results.

The problem can be easily solved, however, by constraining the number of used semantic model elements in the query (e.g., taking only the most important elements), and by limiting the number of subquery results that are considered when calculating the final, combined query results. Limiting the number of subquery results is a very common technique in metasearch engines [Cro00]. Although these workarounds can theoretically decrease IR effectiveness, it is very unlikely that they have any effect on the top ranked documents. Therefore, end user experience is not affected negatively.

12.6 Summary

In this chapter, I evaluated the hypotheses of this work in a large-scale evaluation, using Wikipedia as the test collection. During evaluation, I executed queries in two scenarios (“War on Terrorism” and “Gulf War”) and compared the results with a self-made baseline full-text search engine, implemented using the well-known Lucene Java library.

First, I described the process of test collection creation and the experimental setup in detail. Further, I presented the results of the evaluation experiments and analyzed the results. The experiments showed that the major hypotheses of this work were valid. In particular, the main hypothesis of the work, namely that ontologies can significantly improve IR effectiveness, was clearly validated by the experimental results.

During evaluation, I experienced two surprising results. First, the fuzzy time model did not increase IR efficiency in the case of queries on abstract concepts. Second, combined metadata and content search did not perform better in the chosen scenarios than metadata search alone.

To conclude, the positive results of this evaluation should give enough motivation for decision makers to invest into the improvement of existing, full-text based information repositories by the application of ontologies.

Chapter 13

Conclusion and Outlook

The main question of my thesis was whether using ontologies in the IR process improves IR effectiveness. IR effectiveness was measured in this work using the classical IR measures — precision and recall. There were two major areas to analyze in order to answer this question. On the one hand, the issue of ontology imperfection had to be addressed. On the other hand, performance issues had to be solved.

Starting with the issue of ontology imperfection, it is important to mention that in a perfect world using a perfect ontology with perfect metadata annotations, the question of this thesis can trivially be answered with "yes". In practice, however, ontology development is costly and error-prone. Therefore most (if not all) ontologies are imperfect: There is missing information, they contain errors, or some important domain knowledge cannot be expressed because of the limitations of the ontology formalism. I termed these kinds of imperfection as *ontology imperfection*. Moreover, in some cases our knowledge about the domain is inherently imperfect, therefore we cannot encode perfect information into the ontology. I termed this kind of imperfection *domain imperfection*.

In the area of IR, most often thesauri are used for codifying background knowledge of a domain. Thesauri are similar to ontologies but the formalism is much more limited, language dependent and focuses strongly on linguistic relations among words. During the last decades of IR research, it turned out that using background domain knowledge in form of a thesaurus in the IR process does not automatically increase effectiveness. On the contrary, in many cases a decrease in retrieval performance was observed. The cause was the imperfection of thesauri.

Ontologies provide some more advanced constructs for representing domain knowledge than thesauri. These constructs include non-linguistic semantic relations and the possibility to specify temporal information in the ontology. Therefore, there was a hope at the beginning of this work that using these advanced constructs can compensate the negative effect of imperfection and may lead to more effective IR. Indeed, based on the results presented in Chapter 12, **it turned out that if both semantic relations and temporal information is exploited during IR, it consistently increases IR effectiveness in comparison with the baseline full-text search**. In other words, the initial hypothesis that the advantages of advanced ontology constructs outweigh the drawbacks of ontology imperfection could be successfully validated.

Interestingly however, explicitly representing domain imperfection turned to be useful only in some special scenarios. It did not hurt IR performance, though, to represent domain imperfection explicitly and having this information available in the ontology definitely increases the value of the ontology for domain experts.

The other big problem area that was addressed was the issue of performance. The advanced constructs of ontologies come with a price: Ontology reasoning in highly expressive ontology languages is highly complex, the algorithms are usually exponential, i.e., non-tractable. Therefore, it was a crucial question how to exploit ontologies and provide an IR system that is usable in practice. The solution was to limit ontology reasoning to the indexing step that can be processed offline and use a traditional full-text IR system for further steps of the IR process. Of course, this solution has the apparent drawback that the information model of classical full-text search engines (bag of words) can only represent semantic metadata with some information loss. Luckily, it turned out that even with this theoretical information loss the results were superior to that of traditional full-text search.

To conclude, **the main question of this thesis could be answered positively and at the same time it was possible to provide an ontology-based information system whose performance is comparable with traditional full-text search.**

On the way to find the answer to the main question, this work provided the following contributions:

- A **hybrid information model** was developed which is capable to represent both semantic metadata and the results of natural language processing in one framework. Using this model, it was possible to address the aspect of missing information in ontologies. The introduced model is a variation of the well-known vector space model. This makes it relatively easy to represent it in a form that is compatible with the “bag of words” model and thus use classical, mature, full-text search engines to access it.

- A **comprehensive methodology** is given how to build ontology-based information retrieval systems and particularly how to build the underlying ontology itself. The solution provided here does not make the usual assumption about the correctness of the ontology and the semantic metadata but it is **robust against all kinds of ontology imperfection**.

- A new paradigm for **automatically creating semantic metadata** was developed, which can add **indirectly relevant concepts** to the metadata that are not explicitly mentioned in the text.

- A new **fuzzy set based model** was proposed to **represent imperfect temporal information**, which is a kind of domain imperfection. Imperfection in the temporal dimension is common in many application domains, such as history, news, medical or criminal information systems. An ontology that strives to represent background knowledge in those domains should also be able to represent imperfect temporal information. It was also discussed in detail how to construct fuzzy temporal intervals (elements of the model) in a user friendly way.

  A model that does not provide for the common temporal relations, cannot be considered as a full-featured temporal model. This thesis therefore provides the fuzzification of the 13 classical temporal interval relations of Allen [All83]. This allows reasoning with imperfect temporal information.

- The theoretical and methodological results of this thesis work were tested by **implementing a comprehensive prototype of an ontology-based information retrieval system** (called IRCON). In addition to that, the prototype also provides some basic implementations of several innovative idea how ontologies could be exploited on the user interface level. In IRCON, users interact with the system using the well-known textual queries, or by simply clicking on elements of the user interface. I.e., no additional cognitive effort is required when compared with traditional search engines, in contrast to some other ontology-based solutions.

- Using the developed prototype, an extensive, **large scale evaluation** was conducted to validate the impact of ontologies on IR effectiveness in a "real world" scenario, using the popular Wikipedia encyclopedia as the document collection.

Although the ontology-based IR solution presented here is comprehensive and can be exploited for real-world tasks without any changes, there are of course many possibilities for extensions and for future research. In the following, I discuss some of these possibilities.

Metadata generation: The solution presented in this work concentrates on the issue of adding indirectly relevant concepts to the metadata to maximally support research types of questions. Therefore, it only provides simple solutions for the other tasks of semantic metadata generation, such as text-to-ontology matching and disambiguation. In these areas, ideas from advanced state of the art solutions could be integrated into the system.

Heuristic rules: The form of the heuristic rules for metadata expansion that are used in the current IRCON system is limited. Experiments with more advanced forms of rules would be definitely interesting. Another problem is the manual construction of rules in the current system, which makes installing this solution relatively expensive[1]. Applying approaches of machine learning to automatically learn new rules is definitely a very interesting area to make the solution more economic.

Ontology development: The solution highly depends on a relatively high-quality ontology[2], similarly to all other ontology-based solutions. Developing an ontology is an expensive, error-prone process. Although this work provided some guidelines for the effective development of ontologies specially for IR solutions, the process is still human-resource intensive. Any solution that makes ontology development cheaper is also advantageous for my system.

One possibility for improvement is using ontology learning [Mae02, CV05] to at least partially automatize the ontology building process. My personal experiences before and during this thesis work showed that ontology learning is not yet ready to effectively help ontology engineers in developing ontologies. There is, however, clearly a big potential for improvement and ontology learning will likely to become an effective tool for ontology development. However, my personal opinion is that humans will always be needed in the ontology development process. This is because the main motivation for developing ontologies is to codify information that cannot be (fully) inferred by algorithms. Currently only humans are able to codify such

[1]Although as discussed in Section 5.6, a step-by-step transition path between a traditional full-text search and my system can be followed. I.e., only such investments must be made that are well-motivated by actual user needs.

[2]which does not have to be perfect

information. If an ontology could be developed fully automatically using any ontology learning tool, this would effectively mean that there is no need for ontologies any more because artificial intelligence reached the point where it can compete with human intelligence and thus does not need ontologies for its task. In such a perfect world, humans could have a decent conversation with the information system, where they could use any constructs of their natural language. In such a situation, the need for an ontology would not appear.

Another possibility for improvement in the area of ontology development is to exploit "community intelligence", i.e., to develop ontologies massively collaboratively. In current ontology development clearly the knowledge acquisition and conceptualization steps form the bottleneck because there are usually only few domain experts and ontology engineers available and they are expensive. If the expertise that is needed to develop an ontology could be lowered significantly, the costs of ontology development would drastically drop because practically anybody could participate in the process. The recent success of "tagging" approaches and the buzz about the "Web 2.0" shows how this idea could work in practice. Indeed, the first proposals have already appeared to exploit the tagging approach for ontology development [GL06, SdVM06].

Evolution of ontology and metadata: In the current system, metadata has to be completely regenerated for all documents when the ontology changes. This is not very practical because metadata generation is time consuming. There are some possibilities to optimize this process for specific types of ontology changes. First, the number of documents could be limited where a regeneration of metadata is needed. Second, it could be examined which parts of the metadata has to be regenerated and which parts can be retained.

Choosing the right strategy for query combination: As was shown in Chapter 12, combining the content and metadata queries can harm retrieval performance in areas that are well covered by the ontology. In these areas, it would be better if the metadata query was executed alone. Providing an estimation of ontology coverage for specific topics and using this estimation to decide about the right query strategy (i.e., whether to use combination or not) seems to be a very challenging topic for future research.

Exploiting ontologies on the user interface: This work concentrated on the question whether ontologies improve IR effectiveness in the classical sense, i.e., whether they improve precision and recall measures. Information retrieval is, however, a process rather than a simple query–answer interaction with the system, as was explained in Section 2.1. Ontologies can definitely be used to support user interaction with the system and thus improve the IR process as a whole. Examples for such support include ontology navigation [PKO$^+$04, DW04], semantic disambiguation [GMM03], intelligent suggestion of alternative queries [Sto05], improved visualization of semantic metadata [NDO05] etc. This work already showed some simple solutions in this direction but there is of course room for improvement. The main question according ontology-based user interface improvements is how to evaluate the positive (or negative) effects that these techniques may have. So far it still seems to be an open question which of these techniques are really useful; especially considering the fact that most users are accustomed to a simple full-text input field a la Google and are often annoyed by any other technique that is more complicated than this simple solution.

Appendix A

Relational schema for vector space model performance testing

```
CREATE TABLE DocVector(
 doc_id integer,
 term varchar(255),
 weight real);

CREATE INDEX doc_id_idx ON DocVector (doc_id);
CREATE INDEX term_idx ON DocVector (term);

CREATE VIEW dv_length AS
SELECT doc_id, SQRT(SUM(weight*weight)) AS length
FROM docvector
GROUP BY doc_id;

CREATE TABLE dv_length_mv as select * FROM dv_length;
CREATE INDEX dv_length_idx ON dv_length_mv (doc_id);
```

Appendix B

Evaluation ontology

Table B.1: Ontology instances

Instance label	Concept label
2003 invasion of Baghdad	Event
2003 invasion of Iraq	Event
9/11 hijackers	Organization
Abdulaziz al-Omari	Person
Abu Abdallah Hassan Ben Mahmoud	Person
Abu Ghraib Prison	Organization
Abu Ghraib prison	Organization
Abu Ghraib prisoner abuse scandal	Event
Abu Musab al-Zarqawi	Person
Afghanistan	Geo-Political Entity
Ahmed Abdallah al-Nami	Person
Ahmed al-Ghamdi	Person
Ahmed al-Haznawi	Person
Ahmed al-Nami	Person
Al Jazeera	Organization
Al Jazeera TV	Organization
Al Qa'im	Geo-Political Entity
Albania	Geo-Political Entity
Ali Abdul Aziz Ali	Person
al-Qaeda	Organization
American Airlines Flight 11	Object
American Airlines Flight 77	Object
Asadullah Abdul Rahman	Person
Australia	Geo-Political Entity
Baghdad	Geo-Political Entity
Basra	Geo-Political Entity
Battle of Abu Ghraib	Event
Benyam (Benjamin) Mohammed al Habashi	Person
Brigadier General Abdullah Ali Jasmin	Person
Bulgaria	Geo-Political Entity
Camp Delta	Organization
Camp Echo	Organization
Camp Iguana	Organization
Camp X-Ray	Organization

Table B.1: Ontology instances (cont.)

Instance label	Concept label
Central Intelligence Agency	Organization
Citizens for a Free Kuwait	Organization
Coalition of the Willing	Organization
Congressional Human Rights Caucus	Organization
Croatia	Geo-Political Entity
Cuba	Geo-Political Entity
Czech Republic	Geo-Political Entity
Dalibor Lazarevski	Person
Denmark	Geo-Political Entity
Department of Defense	Organization
Dick Cheney	Person
Donald Henry Rumsfeld	Person
Dragan Markovic	Person
Enzo Baldoni	Person
Estonia	Geo-Political Entity
Fallujah	Geo-Political Entity
Fayez Banihammad	Person
Fedayeen Saddam	Organization
Feredion Jahani	Person
George H. W. Bush	Geo-Political Entity
George W. Bush	Person
Georges Malbrunot	Person
Guantanamo Bay Prison	Organization
Guantanamo Bay Prison Scandal	Event
Gulf War	Event
Hamza al-Ghamdi	Person
Hani Hanjour	Person
Hill & Knowlton	Organization
Huffman Aviation	Organization
Hungary	Geo-Political Entity
Ibrahim Ahmed Mahmoud al Qosi	Person
Information Ministry of Iraq	Organization
Iran	Geo-Political Entity
Iraq	Geo-Political Entity
Iraq disarmament crisis	Event
Iraq War	Event
Iraqi invasion of Kuwait	Event
Iraq’s Special Republican Guard	Organization
Islamic Army in Iraq	Organization
Islamic Front for the Iraqi Resistance	Organization
Islamic Resistance Movement	Organization
Italy	Geo-Political Entity
Jaish Ansar al-Sunna	Organization
James (Jimmy) W. Walter	Person
Khalid al-Mihdhar	Person
Khalid Sheikh Mohammed	Person
Kirkuk	Geo-Political Entity

Table B.1: Ontology instances (cont.)

Instance label	Concept label
Kuwait	Geo-Political Entity
Kuwait City	Geo-Political Entity
Latvia	Geo-Political Entity
Lauri Fitz-Pegado	Person
Liberation Tower	Object
Lithuania	Geo-Political Entity
London	Geo-Political Entity
Lotfi Raissi	Person
Majed Mashaan Moqed	Person
Majed Moqed	Person
Major General Abul Ali Jasmin	Person
Major General Victor Renuart	Person
Manadel al-Jamadi	Person
Marine Corps	Organization
Marwan al-Shehhi	Person
Marwan Ibrahim Kassar	Person
Mohamed al-Kahtani	Person
Mohamed Atta al Sayed	Person
Mohammed Jawdat Hussein	Person
Mohammed Saeed al-Sahaf	Person
Mohand al-Shehri	Person
Muqtada al-Sadr	Person
Mushabib al-Hamlan	Person
Mustafa Ahmed al-Hawsawi	Person
National Library of Iraq	Organization
National Museum of Iraq	Organization
Nawaf al-Hazmi	Person
New York	Geo-Political Entity
Nijirah al-Sabah	Person
Oil Ministry of Iraq	Organization
Operation Abilene	Event
Operation Bayonet Lightning	Event
Operation Bulldog Mammoth	Event
Operation Chamberlain	Event
Operation Clear Area	Event
Operation Desert Farewell	Event
Operation Desert Scorpion	Event
Operation Eagle Curtain	Event
Operation Enduring Freedom	Event
Operation Industrial Sweep	Event
Operation Iron Hammer	Event
Operation Iron Promise	Event
Operation Ivy Blizzard	Event
Operation Ivy Cyclone	Event
Operation Ivy Lightning	Event
Operation Ivy Needle	Event
Operation Ivy Serpent	Event

Table B.1: Ontology instances (cont.)

Instance label	Concept label
Operation Longstreet	Event
Operation Northern Delay	Event
Operation Option North	Event
Operation Panther Squeeze	Event
Operation Peninsula Strike	Event
Operation Phantom Fury	Event
Operation Phantom Linebacker	Event
Operation Planet X	Event
Operation Plymouth Rock	Event
Operation Red Dawn	Event
Operation Resolute Sword	Event
Operation Rifles Blitz	Event
Operation Silverdao	Event
Operation Soda Mountain	Event
Operation Telic	Event
Operation Tiger Clean Sweep	Event
Operation Valiant Strike	Event
Operation Vigilant Resolve	Event
Operation Warrior Sweep	Event
Operation Yellow Ribbon	Event
Osama bin Laden	Person
Pakistan	Geo-Political Entity
Pentagon	Object
Poland	Geo-Political Entity
Portugal	Geo-Political Entity
Preparations for 2003 invasion of Iraq	Event
Raja Azad	Person
Ramzi Binalshibh	Person
Redouan Chekkouri	Person
Republic of Macedonia	Geo-Political Entity
Romania	Geo-Political Entity
Ronald Schulz	Person
Saddam Hussein	Person
Sadeq Muhammad Sa'id Ismail	Person
Saeed al-Ghamdi	Person
Sajad Naeem	Person
Salem al-Hazmi	Person
Salim Ahmed Hamdan	Person
Samarra	Geo-Political Entity
Satam al-Suqami	Person
Saud Nasir al-Sabah	Person
Saudi Arabia	Geo-Political Entity
Scorpions	Organization
September 11, 2001 Terrorist Attack	Event
Slovakia	Geo-Political Entity
Slovenia	Geo-Political Entity
Soviet Union	Geo-Political Entity

Table B.1: Ontology instances (cont.)

Instance label	Concept label
Soviet war in Afghanistan	Event
Spain	Geo-Political Entity
Staff Lieutenant General Mezahem Saab Al Hassan Al-Tikriti	Person
Syria	Geo-Political Entity
Taliban	Organization
Tawfiq bin Attash	Person
The letter of the eight	Object
The Pentagon	Organization
Thomas M. Pappas	Person
Tigris River	Geographical Feature
Tikrit	Geo-Political Entity
U.S. Department of Commerce	Organization
Umm Qasr	Geo-Political Entity
UN Security Council	Organization
United Airlines Flight 175	Object
United Airlines Flight 93	Object
United Arab Emirates	Geo-Political Entity
United Kingdom	Geo-Political Entity
United Nations	Organization
United Nations Security Council Resolution 660	Object
United States	Geo-Political Entity
United States Court of Appeals	Organization
United States Department of Defence	Organization
United States Navy	Organization
United States Supreme Court	Organization
United States war in Afghanistan	Event
US Army	Organization
Veteran Intelligence Professionals for Sanity	Organization
Vilnius letter	Object
Wail Alshehri	Person
Wail al-Shehri	Person
Waleed al-Shehri	Person
War on Terrorism	Event
Wassef Ali Hassoun	Person
World Trade Center	Object
Yarmuk Hospital	Organization
Zacarias Moussaoui	Person
Zakariyah Essabar	Person
Ziad Jarrah	Person
Zoran Naskovski	Person

Table B.2: Ontology relations

Source label	Target label	Relation label
2003 invasion of Baghdad	2003 invasion of Baghdad	isSubEventOf
2003 invasion of Iraq	2003 invasion of Iraq	isSubEventOf

Table B.2: Ontology relations (cont.)

Source label	Target label	Relation label
Abu Ghraib prisoner abuse scandal	Abu Ghraib prisoner abuse scandal	isSubEventOf
Guantanamo Bay Prison Scandal	Guantanamo Bay Prison Scandal	isSubEventOf
Iraq disarmament crisis	Iraq disarmament crisis	isSubEventOf
Iraq War	Iraq War	isSubEventOf
Iraqi invasion of Kuwait	Iraqi invasion of Kuwait	isSubEventOf
Operation Abilene	Operation Abilene	isSubEventOf
Operation Bayonet Lightning	Operation Bayonet Lightning	isSubEventOf
Operation Bulldog Mammoth	Operation Bulldog Mammoth	isSubEventOf
Operation Chamberlain	Operation Chamberlain	isSubEventOf
Operation Clear Area	Operation Clear Area	isSubEventOf
Operation Desert Farewell	Operation Desert Farewell	isSubEventOf
Operation Desert Scorpion	Operation Desert Scorpion	isSubEventOf
Operation Eagle Curtain	Operation Eagle Curtain	isSubEventOf
Operation Enduring Freedom	Operation Enduring Freedom	isSubEventOf
Operation Industrial Sweep	Operation Industrial Sweep	isSubEventOf
Operation Iron Hammer	Operation Iron Hammer	isSubEventOf
Operation Iron Promise	Operation Iron Promise	isSubEventOf
Operation Ivy Blizzard	Operation Ivy Blizzard	isSubEventOf
Operation Ivy Cyclone	Operation Ivy Cyclone	isSubEventOf
Operation Ivy Lightning	Operation Ivy Lightning	isSubEventOf
Operation Ivy Needle	Operation Ivy Needle	isSubEventOf
Operation Ivy Serpent	Operation Ivy Serpent	isSubEventOf
Operation Longstreet	Operation Longstreet	isSubEventOf
Operation Northern Delay	Operation Northern Delay	isSubEventOf
Operation Option North	Operation Option North	isSubEventOf
Operation Panther Squeeze	Operation Panther Squeeze	isSubEventOf
Operation Peninsula Strike	Operation Peninsula Strike	isSubEventOf
Operation Phantom Fury	Operation Phantom Fury	isSubEventOf
Operation Phantom Linebacker	Operation Phantom Linebacker	isSubEventOf
Operation Planet X	Operation Planet X	isSubEventOf
Operation Plymouth Rock	Operation Plymouth Rock	isSubEventOf
Operation Red Dawn	Operation Red Dawn	isSubEventOf
Operation Resolute Sword	Operation Resolute Sword	isSubEventOf
Operation Rifles Blitz	Operation Rifles Blitz	isSubEventOf
Operation Silverdao	Operation Silverdao	isSubEventOf
Operation Soda Mountain	Operation Soda Mountain	isSubEventOf
Operation Telic	Operation Telic	isSubEventOf
Operation Tiger Clean Sweep	Operation Tiger Clean Sweep	isSubEventOf
Operation Valiant Strike	Operation Valiant Strike	isSubEventOf
Operation Vigilant Resolve	Operation Vigilant Resolve	isSubEventOf
Operation Warrior Sweep	Operation Warrior Sweep	isSubEventOf
Operation Yellow Ribbon	Operation Yellow Ribbon	isSubEventOf
September 11, 2001 Terrorist Attack	September 11, 2001 Terrorist Attack	isSubEventOf
United States war in Afghanistan	United States war in Afghanistan	isSubEventOf

Table B.3: Ontology temporal relations

Source label	Target label	Relation label[1]
2003 invasion of Iraq	Iraq	happensAt
Abdulaziz al-Omari	September 11, 2001 Terrorist Attack	initiates
Abu Abdallah Hassan Ben Mahmoud	Islamic Army in Iraq	leads
Abu Ghraib prison	Abu Ghraib prisoner abuse scandal	isInvolvedIn
Afghanistan	Soviet war in Afghanistan	isInvolvedIn
Ahmed al-Ghamdi	September 11, 2001 Terrorist Attack	initiates
Ahmed al-Haznawi	September 11, 2001 Terrorist Attack	initiates
Ahmed al-Nami	September 11, 2001 Terrorist Attack	initiates
Ali Abdul Aziz Ali	al-Qaeda	isMemberOf
American Airlines Flight 11	Abdulaziz al-Omari	interactsWith
American Airlines Flight 11	Mohamed Atta al Sayed	interactsWith
American Airlines Flight 11	Satam al-Suqami	interactsWith
American Airlines Flight 11	September 11, 2001 Terrorist Attack	isInvolvedIn
American Airlines Flight 11	Wail al-Shehri	interactsWith
American Airlines Flight 11	Waleed al-Shehri	interactsWith
American Airlines Flight 11	World Trade Center	interactsWith
American Airlines Flight 77	Hani Hanjour	interactsWith
American Airlines Flight 77	Khalid al-Mihdhar	interactsWith
American Airlines Flight 77	Majed Moqed	interactsWith
American Airlines Flight 77	Nawaf al-Hazmi	interactsWith
American Airlines Flight 77	Pentagon	interactsWith
American Airlines Flight 77	Salem al-Hazmi	interactsWith
American Airlines Flight 77	September 11, 2001 Terrorist Attack	isInvolvedIn
Asadullah Abdul Rahman	Guantanamo Bay Prison	interactsWith
Asadullah Abdul Rahman	Guantanamo Bay Prison Scandal	isInvolvedIn
Baghdad	Iraq	isPartOf
Basra	Iraq	isPartOf
Benyam (Benjamin) Mohammed al Habashi	Guantanamo Bay Prison	interactsWith
Benyam (Benjamin) Mohammed al Habashi	Guantanamo Bay Prison Scandal	isInvolvedIn
Camp Delta	Guantanamo Bay Prison	isChildOrganizationOf
Camp Echo	Guantanamo Bay Prison	isChildOrganizationOf
Camp Iguana	Guantanamo Bay Prison	isChildOrganizationOf

[1] The ontology formalism allows to specify the validity time for the temporal relations. I did not use this feature, however, during the evaluation. Therefore, the temporal specifications are omitted from this table.

Table B.3: Ontology temporal relations (cont.)

Source label	Target label	Relation label
Camp X-Ray	Guantanamo Bay Prison	isChild-OrganizationOf
Congressional Human Rights Caucus	Gulf War	isInvolvedIn
Congressional Human Rights Caucus	United States	isLocatedAt
Dalibor Lazarevski	Islamic Army in Iraq	interactsWith
Department of Defense	Guantanamo Bay Prison Scandal	isInvolvedIn
Dick Cheney	Gulf War	isInvolvedIn
Donald Henry Rumsfeld	Guantanamo Bay Prison Scandal	isInvolvedIn
Dragan Markovic	Islamic Army in Iraq	interactsWith
Enzo Baldoni	Islamic Army in Iraq	interactsWith
Fallujah	Iraq	isPartOf
Fayez Banihammad	September 11, 2001 Terrorist Attack	initiates
Fedayeen Saddam	2003 invasion of Iraq	isInvolvedIn
Feredion Jahani	Islamic Army in Iraq	interactsWith
George H. W. Bush	Gulf War	isInvolvedIn
George W. Bush	2003 invasion of Iraq	isInvolvedIn
Georges Malbrunot	Islamic Army in Iraq	interactsWith
Guantanamo Bay Prison	Cuba	isLocatedAt
Guantanamo Bay Prison	Guantanamo Bay Prison Scandal	isInvolvedIn
Gulf War	Iraq	happensAt
Hamza al-Ghamdi	September 11, 2001 Terrorist Attack	initiates
Hani Hanjour	September 11, 2001 Terrorist Attack	initiates
Hill & Knowlton	Gulf War	isInvolvedIn
Hill & Knowlton	United States	isLocatedAt
Ibrahim Ahmed Mahmoud al Qosi	al-Qaeda	isMemberOf
Ibrahim Ahmed Mahmoud al Qosi	Guantanamo Bay Prison	interactsWith
Information Ministry of Iraq	Baghdad	isLocatedAt
Iran	Soviet war in Afghanistan	isInvolvedIn
Iraq	Gulf War	initiates
Iraq War	Iraq	happensAt
Iraqi invasion of Kuwait	Kuwait	happensAt
Islamic Army in Iraq	2003 invasion of Iraq	isInvolvedIn
Islamic Front for the Iraqi Resistance	2003 invasion of Iraq	isInvolvedIn
Islamic Resistance Movement	2003 invasion of Iraq	isInvolvedIn
Jaish Ansar al-Sunna	2003 invasion of Iraq	isInvolvedIn
James (Jimmy) W. Walter	September 11, 2001 Terrorist Attack	initiates
Khalid al-Mihdhar	September 11, 2001 Terrorist Attack	initiates

Table B.3: Ontology temporal relations (cont.)

Source label	Target label	Relation label
Khalid Sheikh Mohammed	al-Qaeda	isMemberOf
Kirkuk	Iraq	isPartOf
Kuwait	Gulf War	isInvolvedIn
Kuwait City	Kuwait	isPartOf
Lauri Fitz-Pegado	Gulf War	isInvolvedIn
Lauri Fitz-Pegado	Hill & Knowlton	isMemberOf
Liberation Tower	Gulf War	isInvolvedIn
Liberation Tower	Kuwait	isLocatedAt
London	United Kingdom	isPartOf
Lotfi Raissi	September 11, 2001 Terrorist Attack	initiates
Majed Mashaan Moqed	September 11, 2001 Terrorist Attack	initiates
Majed Moqed	September 11, 2001 Terrorist Attack	initiates
Major General Victor Renuart	2003 invasion of Baghdad	isInvolvedIn
Manadel al-Jamadi	Abu Ghraib prisoner abuse scandal	isInvolvedIn
Marwan al-Shehhi	Huffman Aviation	isMemberOf
Marwan al-Shehhi	September 11, 2001 Terrorist Attack	initiates
Marwan Ibrahim Kassar	Islamic Army in Iraq	interactsWith
Mohamed al-Kahtani	al-Qaeda	isMemberOf
Mohamed al-Kahtani	Guantanamo Bay Prison	interactsWith
Mohamed Atta al Sayed	Huffman Aviation	isMemberOf
Mohamed Atta al Sayed	September 11, 2001 Terrorist Attack	initiates
Mohammed Jawdat Hussein	Islamic Army in Iraq	interactsWith
Mohammed Saeed al-Sahaf	Information Ministry of Iraq	leads
Mohand al-Shehri	September 11, 2001 Terrorist Attack	initiates
Mushabib al-Hamlan	al-Qaeda	isMemberOf
Mustafa Ahmed al-Hawsawi	September 11, 2001 Terrorist Attack	initiates
National Library of Iraq	2003 invasion of Baghdad	isInvolvedIn
National Museum of Iraq	Baghdad	isLocatedAt
Nawaf al-Hazmi	September 11, 2001 Terrorist Attack	initiates
New York	United States	isPartOf
Nijirah al-Sabah	Gulf War	isInvolvedIn
Nijirah al-Sabah	Saud Nasir al-Sabah	interactsWith
Oil Ministry of Iraq	Baghdad	isLocatedAt
Osama bin Laden	al-Qaeda	leads
Osama bin Laden	Soviet war in Afghanistan	isInvolvedIn
Pakistan	Soviet war in Afghanistan	isInvolvedIn
Pentagon	September 11, 2001 Terrorist Attack	isInvolvedIn

Table B.3: Ontology temporal relations (cont.)

Source label	Target label	Relation label
Raja Azad	Islamic Army in Iraq	interactsWith
Ramzi Binalshibh	al-Qaeda	isMemberOf
Redouan Chekkouri	Camp Delta	interactsWith
Ronald Schulz	Islamic Army in Iraq	interactsWith
Saddam Hussein	2003 invasion of Iraq	isInvolvedIn
Saddam Hussein	Gulf War	isInvolvedIn
Sadeq Muhammad Sa'id Ismail	Camp Delta	interactsWith
Saeed al-Ghamdi	al-Qaeda	isMemberOf
Saeed al-Ghamdi	September 11, 2001 Terrorist Attack	initiates
Sajad Naeem	Islamic Army in Iraq	interactsWith
Salem al-Hazmi	September 11, 2001 Terrorist Attack	initiates
Salim Ahmed Hamdan	Guantanamo Bay Prison	interactsWith
Samarra	2003 invasion of Iraq	isInvolvedIn
Satam al-Suqami	September 11, 2001 Terrorist Attack	initiates
Saud Nasir al-Sabah	Gulf War	isInvolvedIn
Saudi Arabia	Gulf War	isInvolvedIn
Saudi Arabia	Soviet war in Afghanistan	isInvolvedIn
Scorpions	2003 invasion of Iraq	isInvolvedIn
September 11, 2001 Terrorist Attack	New York	happensAt
Soviet Union	Soviet war in Afghanistan	isInvolvedIn
Soviet war in Afghanistan	Afghanistan	happensAt
Staff Lieutenant General Mezahem Saab Al Hassan Al-Tikriti	Fedayeen Saddam	leads
Tawfiq bin Attash	al-Qaeda	isMemberOf
The letter of the eight	Czech Republic	interactsWith
The letter of the eight	Denmark	interactsWith
The letter of the eight	Hungary	interactsWith
The letter of the eight	Iraq disarmament crisis	isInvolvedIn
The letter of the eight	Italy	interactsWith
The letter of the eight	Poland	interactsWith
The letter of the eight	Portugal	interactsWith
The letter of the eight	Spain	interactsWith
The letter of the eight	United Kingdom	interactsWith
Thomas M. Pappas	Abu Ghraib prisoner abuse scandal	isInvolvedIn
UN Security Council	Iraq disarmament crisis	isInvolvedIn
UN Security Council	United Nations Security Council Resolution 660	interactsWith
United Airlines Flight 175	Ahmed al-Ghamdi	interactsWith
United Airlines Flight 175	Fayez Banihammad	interactsWith
United Airlines Flight 175	Hamza al-Ghamdi	interactsWith
United Airlines Flight 175	Marwan al-Shehhi	interactsWith
United Airlines Flight 175	Mohand al-Shehri	interactsWith

Table B.3: Ontology temporal relations (cont.)

Source label	Target label	Relation label
United Airlines Flight 175	September 11, 2001 Terrorist Attack	isInvolvedIn
United Airlines Flight 175	World Trade Center	interactsWith
United Airlines Flight 93	Ahmed al-Haznawi	interactsWith
United Airlines Flight 93	Ahmed al-Nami	interactsWith
United Airlines Flight 93	Saeed al-Ghamdi	interactsWith
United Airlines Flight 93	September 11, 2001 Terrorist Attack	isInvolvedIn
United Airlines Flight 93	Ziad Jarrah	interactsWith
United Nations Security Council Resolution 660	Gulf War	isInvolvedIn
United Nations Security Council Resolution 660	Iraqi invasion of Kuwait	isInvolvedIn
United States	2003 invasion of Iraq	isInvolvedIn
United States	Gulf War	initiates
United States	Soviet war in Afghanistan	isInvolvedIn
United States Court of Appeals	Guantanamo Bay Prison Scandal	isInvolvedIn
United States Court of Appeals	United States	isLocatedAt
United States Navy	US Army	isChildOrganizationOf
United States Supreme Court	Guantanamo Bay Prison Scandal	isInvolvedIn
United States Supreme Court	United States	isLocatedAt
United States war in Afghanistan	Afghanistan	happensAt
Veteran Intelligence Professionals for Sanity	2003 invasion of Iraq	isInvolvedIn
Vilnius letter	Albania	interactsWith
Vilnius letter	Bulgaria	interactsWith
Vilnius letter	Croatia	interactsWith
Vilnius letter	Estonia	interactsWith
Vilnius letter	Iraq disarmament crisis	isInvolvedIn
Vilnius letter	Latvia	interactsWith
Vilnius letter	Lithuania	interactsWith
Vilnius letter	Republic of Macedonia	interactsWith
Vilnius letter	Romania	interactsWith
Vilnius letter	Slovakia	interactsWith
Vilnius letter	Slovenia	interactsWith
Wail Alshehri	September 11, 2001 Terrorist Attack	isInvolvedIn
Wail al-Shehri	September 11, 2001 Terrorist Attack	initiates
Waleed al-Shehri	September 11, 2001 Terrorist Attack	initiates
Wassef Ali Hassoun	2003 invasion of Iraq	isInvolvedIn
World Trade Center	New York	isLocatedAt
World Trade Center	September 11, 2001 Terrorist Attack	isInvolvedIn
Yarmuk Hospital	Baghdad	isLocatedAt

Table B.3: Ontology temporal relations (cont.)

Source label	Target label	Relation label
Zacarias Moussaoui	al-Qaeda	isMemberOf
Zakariyah Essabar	al-Qaeda	isMemberOf
Ziad Jarrah	al-Qaeda	isMemberOf
Ziad Jarrah	September 11, 2001 Terrorist Attack	initiates
Zoran Naskovski	Islamic Army in Iraq	interactsWith

Table B.4: Ontology temporal attributes

Source label	Attribute label	Time specification
George H. W. Bush	isActiveDuring	1989-01-20;1993-01-20
George W. Bush	isActiveDuring	2001-01-20;
2003 invasion of Iraq	happensDuring	2002-11-09[0.3];[2] 2003-03-17[0.7]; 2003-03-20[1.0]; 2003-05-01[0.5]; 2003-12-13[0.2]
September 11, 2001 Terrorist Attack	happensDuring	2001-09-11;2001-09-12
Soviet war in Afghanistan	happensDuring	1979-12-27;1989-02-15
Gulf War	happensDuring	1990-08-02;1991-02-28
Iraq War	happensDuring	2003-03-19;
United States war in Afghanistan	happensDuring	2001-10-07;
War on Terrorism	happensDuring	2001-09-11;
Iraqi invasion of Kuwait	happensDuring	1990-08-02;1990-08-03

Table B.5: Ontology lexical layer

Labels (in bold) and their synonyms
2003 invasion of Baghdad
Invasion of Baghdad
2003 invasion of Iraq
2003 Invasion of Iraq
Abdulaziz al-Omari
Abdulaziz Alomari
Abu Musab al-Zarqawi
al-Zarqawi
Ahmed Abdallah al-Nami
Ahmed Abdallah Alnami, Ahmed Abdallah al-Nawi, al-Nami, Alnami, al-Nawi
Ahmed al-Ghamdi
Ahmed Salah al-Ghamdi, Ahmed Alghamdi
Ahmed al-Haznawi
Ahmed Ibrahim al-Haznawi, al-Haznawi

[2]This is a fuzzy temporal specification. The form of the specification is the following: `date_spec_1[subjective_value1];...;date_spec_N[subjective_valueN]`, where the subjective value denotes the subjective value *after* the specified time point (date).

Table B.5: Ontology lexical layer (cont.)

Labels (in bold) and their synonyms
Ahmed al-Nami
Ahmed Abdallah al-Nami, Ahmed al-Nami, Ahmed al-Nawi
Al Jazeera
al Jazeera, Al Jazeera
al-Qaeda
al Qaeda terrorist network, al Qaeda, al-Qaida
Benyam (Benjamin) Mohammed al Habashi
Binyam Mohammed
Central Intelligence Agency
CIA
Citizens for a Free Kuwait
CFK
Congressional Human Rights Caucus
Human Rights Caucus
Department of Defense
DoD
Dick Cheney
Cheney
Donald Henry Rumsfeld
Henry Rumsfeld, Secretary Rumsfeld, Rumsfeld
Fayez Banihammad
Fayez Rashid Ahmed Hassan al Qadi Banihammad
George H. W. Bush
President Bush, Bush, George Bush, President George H.W. Bush, George Bush Snr.
George W. Bush
President George W. Bush, Bush, George Bush, President Bush
Guantanamo Bay Prison
Guantanamo Bay
Gulf War
Persian Gulf War, Operation Desert Storm, Gulf War I, First Persian Gulf War, Operation Desert Shield, Operation Granby
Hamza al-Ghamdi
Hamza Alghamdi
Hani Hanjour
Hani Saleh Hanjour
Hill & Knowlton
H&K
Ibrahim Ahmed Mahmoud al Qosi
Ibrahim Ahmed Mahmoud
Information Ministry of Iraq
Information Ministry
Iraq War
war of Iraq, 2003 Iraq war, Operation Iraqi Freedom, 2003 Iraq War, Second Gulf War, 2003 occupation of Iraq
Iraqi invasion of Kuwait
Invasion of Kuwait by Iraq

Table B.5: Ontology lexical layer (cont.)

Labels (in bold) and their synonyms
Islamic Army in Iraq
IAI
Jaish Ansar al-Sunna
al-Sunna
James (Jimmy) W. Walter
Jimmy Walter
Khalid al-Mihdhar
Khalid Almihdhar
Liberation Tower
The Kuwait Telecommunications Tower
Lotfi Raissi
Raissi
Majed Mashaan Moqed
Moqued
Majed Moqed
Majed Mashaan Moqed, Majed Moqued
Major General Victor Renuart
Victor Renuart
Marine Corps
Marines
Marwan al-Shehhi
Marwan Yousef al-Shehhi, Marwan Alshehhi
Mohamed Atta al Sayed
Mohamed Atta al-Sayed , Mehan Atta, Mohammed Atta, Mohammad El Amir, Mohamed El Sayed, Muhammad Muhammad Al Amir Awag Al Sayyid Atta, Muhammad Muhammad Al-Amir Awad Al Sayad, Mohamed Mohamed Elamir Awad Elsayed, Mohamed Atta
Mohand al-Shehri
Mohand Alshehri
National Library of Iraq
National Library
Nawaf al-Hazmi
Nawaq Alhazmi, Rabia al Makki , Nawaf M.S. Al Hazmi
New York
New York City
Nijirah al-Sabah
Nayirah, Nurse Nayirah
Operation Enduring Freedom
OEF
Operation Phantom Fury
Operation Al-Fajr
Operation Telic
Op TELIC
Osama bin Laden
Usama bin Laden, bin Laden
Saddam Hussein
Iraqi President Saddam Hussein

Table B.5: Ontology lexical layer (cont.)

Labels (in bold) and their synonyms
Saeed al-Ghamdi
Saeed Alghamdi, al-Ghamdi
Salem al-Hazmi
Salem Alhazmi
Satam al-Suqami
Satam M. A. al-Suqami
Saud Nasir al-Sabah
Nasir al-Sabah
September 11, 2001 Terrorist Attack
9/11, September 11, 2001 attacks, 2001 terrorist attack, September 11, 2001 attack, terrorist attack on September 11, 2001, September 11, 2001 terrorist attack
Staff Lieutenant General Mezahem Saab Al Hassan Al-Tikriti
General Mezahem, Lieutenant General Mezahem
Thomas M. Pappas
Colonel Pappas
UN Security Council
United Nations Security Council
United Arab Emirates
UAE
United Kingdom
U.K.
United States
U.S., USA, United States of America
United States Supreme Court
Supreme Court
United States war in Afghanistan
2001 war in Afghanistan
Veteran Intelligence Professionals for Sanity
VIPS
Wail al-Shehri
Wail Alshehri
Waleed al-Shehri
Waleed M. al-Shehri, Waleed M. Alshehri
War on Terrorism
War on Terror, GWOT
World Trade Center
WTC, The Twin Towers
Ziad Jarrah
Ziad Samir Jarrah

Appendix C

Ontology-based heuristic rules

C.1 XML Schema for the definition of ontology-based heuristic rules

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">
  <xs:element name="heuristics">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="rule"/>
      </xs:sequence>
      <xs:attribute name="disable_all" use="required"
        type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="rule">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="condition"/>
        <xs:element minOccurs="0" ref="temporal_restriction"/>
         <xs:element ref="target_concept"/>
      </xs:sequence>
      <xs:attribute name="amplification_factor"
        use="optional" type="xs:decimal"/>
      <xs:attribute name="enable" use="required"
        type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="condition">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="cardinality"/>
        <xs:element ref="target_relation"/>
      </xs:sequence>
      <xs:attribute name="weight_threshold"
        use="optional" type="xs:decimal"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:anyURI"/>
  <xs:element name="cardinality" type="xs:integer"/>
  <xs:element name="target_relation">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:anyURI">
          <xs:attribute name="type" type="xs:NCName"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="temporal_restriction" type="xs:anyURI"/>
  <xs:element name="target_concept" type="xs:anyURI"/>
</xs:schema>
```

C.2 XML definition of the ontology-based heuristic rules used during the evaluation

```
<heuristics
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="heuristics_for_evaluation.xsd"
  disable_all="no">

 <rule enable="yes" amplification_factor="0.98"
  name="3␣Participants">
  <condition weight_threshold="0.1">
   <source_concept>
    urn:ircon:ontology#Participant
   </source_concept>
   <cardinality>3</cardinality>
   <temporal_relation>
    urn:ircon:ontology#isInvolvedIn
   </temporal_relation>
  </condition>
  <temporal_attribute>
   urn:ircon:ontology#happensDuring
  </temporal_attribute>
  <target_concept>urn:ircon:ontology#Event</target_concept>
 </rule>

 <rule enable="yes" amplification_factor="0.98"
  name="2␣Participants␣1␣Subevent">
  <condition weight_threshold="0.1">
   <source_concept>
    urn:ircon:ontology#Participant
   </source_concept>
```

```
    <cardinality>2</cardinality>
    <temporal_relation>
     urn:ircon:ontology#isInvolvedIn
    </temporal_relation>
   </condition>
   <condition weight_threshold="0.2">
    <source_concept>urn:ircon:ontology#Event</source_concept>
    <cardinality>1</cardinality>
    <relation>urn:ircon:ontology#isSubEventOf</relation>
   </condition>
   <temporal_attribute>
    urn:ircon:ontology#happensDuring
   </temporal_attribute>
   <target_concept>urn:ircon:ontology#Event</target_concept>
  </rule>

  <rule enable="yes" amplification_factor="0.98"
   name="Participants␣and␣Agent">
   <condition weight_threshold="0.1">
    <source_concept>
     urn:ircon:ontology#Participant
    </source_concept>
    <cardinality>3</cardinality>
    <temporal_relation>
     urn:ircon:ontology#isInvolvedIn
    </temporal_relation>
   </condition>
   <condition weight_threshold="0.2">
    <source_concept>urn:ircon:ontology#Agent</source_concept>
    <cardinality>1</cardinality>
    <temporal_relation>
     urn:ircon:ontology#initiates
    </temporal_relation>
   </condition>
   <temporal_attribute>
    urn:ircon:ontology#happensDuring
   </temporal_attribute>
   <target_concept>urn:ircon:ontology#Event</target_concept>
  </rule>

  <rule enable="yes" amplification_factor="0.98"
   name="2␣Agents">
   <condition weight_threshold="0.1">
    <source_concept>urn:ircon:ontology#Agent</source_concept>
    <cardinality>2</cardinality>
    <relation>urn:ircon:ontology#initiates</relation>
   </condition>
   <temporal_attribute>
    urn:ircon:ontology#happensDuring
   </temporal_attribute>
  <target_concept>urn:ircon:ontology#Event</target_concept>
 </rule>

 <rule enable="yes" amplification_factor="0.98"
  name="Participants␣and␣Location">
  <condition weight_threshold="0.1">
   <source_concept>
    urn:ircon:ontology#Participant
   </source_concept>
   <cardinality>2</cardinality>
   <temporal_relation>
    urn:ircon:ontology#isInvolvedIn
   </temporal_relation>
  </condition>
  <condition weight_threshold="0.2">
   <source_concept>
    urn:ircon:ontology#Location
   </source_concept>
   <cardinality>1</cardinality>
   <temporal_relation>
    urn:ircon:ontology#eventHappensAt
   </temporal_relation>
  </condition>
  <temporal_attribute>
   urn:ircon:ontology#happensDuring
  </temporal_attribute>
  <target_concept>urn:ircon:ontology#Event</target_concept>
 </rule>

 <rule enable="yes" amplification_factor="0.7"
  name="Leader␣and␣Member">
  <condition weight_threshold="0.2">
   <source_concept>urn:ircon:ontology#Agent</source_concept>
   <cardinality>1</cardinality>
   <temporal_relation>
    urn:ircon:ontology#leads
   </temporal_relation>
  </condition>
  <condition weight_threshold="0.1">
   <source_concept>urn:ircon:ontology#Agent</source_concept>
   <cardinality>2</cardinality>
   <temporal_relation>
    urn:ircon:ontology#isMemberOf
   </temporal_relation>
  </condition>
  <temporal_attribute>
   urn:ircon:ontology#isActiveDuring
  </temporal_attribute>
  <target_concept>
   urn:ircon:ontology#Organization
  </target_concept>
</rule>

<rule enable="yes" amplification_factor="0.7"
  name="Members">
  <condition weight_threshold="0.1">
    <source_concept>urn:ircon:ontology#Agent</source_concept>
    <cardinality>3</cardinality>
    <temporal_relation>
      urn:ircon:ontology#isMemberOf
    </temporal_relation>
  </condition>
  <temporal_attribute>
    urn:ircon:ontology#isActiveDuring
  </temporal_attribute>
  <target_concept>
    urn:ircon:ontology#Organization
  </target_concept>
</rule>

<rule enable="yes" amplification_factor="0.5"
  name="Interaction">
  <condition weight_threshold="0.1">
    <source_concept>
      urn:ircon:ontology#Participant
    </source_concept>
    <cardinality>3</cardinality>
    <temporal_relation>
      urn:ircon:ontology#interactsWith
    </temporal_relation>
  </condition>
  <temporal_attribute>
    urn:ircon:ontology#isActiveDuring
  </temporal_attribute>
  <target_concept>
    urn:ircon:ontology#Participant
  </target_concept>
</rule>

<rule enable="yes" amplification_factor="0.7"
  name="Part␣of">
  <condition weight_threshold="0.1">
    <source_concept>
      urn:ircon:ontology#Location
    </source_concept>
    <cardinality>1</cardinality>
    <temporal_relation>
      urn:ircon:ontology#isPartOf
    </temporal_relation>
  </condition>
  <temporal_attribute>
   urn:ircon:ontology#isActiveDuring
  </temporal_attribute>
  <target_concept>
   urn:ircon:ontology#Location
  </target_concept>
 </rule>

 <rule enable="yes" amplification_factor="0.7"
  name="Participants␣at␣Location">
  <condition weight_threshold="0.1">
   <source_concept>
    urn:ircon:ontology#Participant
   </source_concept>
   <cardinality>4</cardinality>
   <temporal_relation>
    urn:ircon:ontology#isLocatedAt
   </temporal_relation>
  </condition>
  <temporal_attribute>
   urn:ircon:ontology#isActiveDuring
  </temporal_attribute>
  <target_concept>
   urn:ircon:ontology#Location
  </target_concept>
 </rule>

 <rule enable="yes" amplification_factor="0.9"
  name="Participants␣and␣Events␣at␣Location">
  <condition weight_threshold="0.1">
   <source_concept>
    urn:ircon:ontology#Participant
   </source_concept>
   <cardinality>2</cardinality>
   <temporal_relation>
    urn:ircon:ontology#isLocatedAt
   </temporal_relation>
  </condition>
  <condition weight_threshold="0.2">
   <source_concept>urn:ircon:ontology#Event</source_concept>
   <cardinality>1</cardinality>
   <temporal_relation>
    urn:ircon:ontology#happensAt
   </temporal_relation>
  </condition>
  <temporal_attribute>
   urn:ircon:ontology#isActiveDuring
  </temporal_attribute>
  <target_concept>
   urn:ircon:ontology#Location
```

```
    </target_concept>
  </rule>

  <rule enable="yes" amplification_factor="0.7"
    name="1␣Subevent">
    <condition weight_threshold="0.2">
      <source_concept>urn:ircon:ontology#Event</source_concept>
      <cardinality>1</cardinality>
      <relation>urn:ircon:ontology#isSubEventOf</relation>
    </condition>
    <temporal_attribute>
      urn:ircon:ontology#happensDuring
    </temporal_attribute>
    <target_concept>urn:ircon:ontology#Event</target_concept>
  </rule>

  <rule enable="yes" amplification_factor="0.9"
    name="2␣Subevents">
    <condition weight_threshold="0.1">
      <source_concept>urn:ircon:ontology#Event</source_concept>
      <cardinality>2</cardinality>
      <relation>urn:ircon:ontology#isSubEventOf</relation>
    </condition>
    <temporal_attribute>
      urn:ircon:ontology#happensDuring
    </temporal_attribute>
    <target_concept>urn:ircon:ontology#Event</target_concept>
  </rule>

</heuristics>
```

References

[All83] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[AM01] Javed A. Aslam and Mark H. Montague. Models for Metasearch. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 276–284, 2001.

[AYCR$^+$05] Sihem Amer-Yahia, Pat Case, Thomas Rölleke, Jayavel Shanmugasundaram, and Gerhard Weikum. Report on the DB/IR Panel at SIGMOD 2005. *SIGMOD Record*, 34(4):71–74, December 2005.

[Bat89] Marcia J. Bates. The Design of Browsing and Berrypicking Techniques for the Online Search Interface. *Online Review*, 13(5):407–424, 1989.

[Bat90] Marcia J. Bates. Where should the person stop and the information search interface start? *Information Processing & Management*, 26(5):575–591, 1990.

[BCM$^+$03] Franz Baader, Diego Calvanese, Deborah L. McGuinnes, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.

[Bec00] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000.

[BG04] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Recommendation, World Wide Web Consortium, February 2004. Available from `http://www.w3.org/TR/rdf-schema/`.

[BH91] Franz Baader and Philipp Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In Ray Myopoulos and John Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 452–457, Sydney, Australia, August 1991. Morgan Kaufmann.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[BR99] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[BWK$^+$03] Sabine Bergle, René Witte, Michelle Khalife, Zhuoyan Li, and Frank Rudzicz. Using Knowledge-poor Coreference Resolution for Text Summarization. In *Workshop on Text Summarization, Document Understanding Conference (DUC)*, Edmonton, Canada, 2003. NIST.

[CGT90] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Surveys in Computer Science. Springer Verlag (Heidelberg, FRG and NewYork NY, USA), 1990.

[Cho94] Jan Chomicki. Temporal Query Languages: A Survey. In Dov M. Gabbay and Hans Jürgen Ohlbach, editors, *Proceedings of the 1st International Conference on Temporal Logic*, volume 827 of *LNAI*, pages 506–534, Berlin, July 1994. Springer.

[Cle91] Cyril W. Cleverdon. The significance of the Cranfield tests on index languages. In *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12, New York, NY, USA, 1991. ACM Press.

[CLS05] Philipp Cimiano, Günter Ladwig, and Steffen Staab. Gimme' the context: context-driven automatic semantic annotation with C-PANKOW. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005*, pages 332–341, Chiba, Japan, May 2005. ACM.

[Cro00] Bruce W. Croft. *Combining approaches to information retrieval*, chapter 1, pages 1–36. Kluwer Academic Publishers, 2000.

[CV05] Philipp Cimiano and Johanna Völker. Text2Onto – A Framework for Ontology Learning and Data-driven Change Discovery. In Andres Montoyo, Rafael Munoz, and Elisabeth Metais, editors, *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 3513 of *Lecture Notes in Computer Science*, pages 227–238, Alicante, Spain, June 2005. Springer.

[Dah96] Michael Dahr. *Deductive Databases: Theory and Applications*. International Thomson Publishing, December 1996.

[DAR02] DARPA Agent Markup Language project. *DAML-Time Homepage*, 2002. Accessible from the URL `http://www.cs.rochester.edu/~ferguson/daml/`.

[dB05] Jos de Bruijn. The Web Service Modeling Language WSML. Technical Report D16.1 version 0.2, WSMO Deliverable, March 2005. Available from `http://www.wsmo.org/2004/d16/d16.1/v0.2/`.

[DEG$^+$03] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. In *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003*, pages 178–186, Budapest, Hungary, May 2003.

[Dey01] Anind K. Dey. Understanding and Using Context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.

[DP86] Didier Dubois and Henri Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York, 1986.

[DP89] Didier Dubois and Henri Prade. Processing Fuzzy Temporal Knowledge. *IEEE Transactions of Systems, Man and Cybernetics*, 19(4):729–744, 1989.

[DP00] Didier Dubois and Henri Prade. *Fundamentals of Fuzzy Sets*, volume 7 of *The handbooks of fuzzy sets series*. Kluwer Academic Publishers, 2000.

[DRS01] Alex Dekhtyar, Robert Ross, and V. S. Subrahmanian. Probabilistic temporal databases, I: algebra. *ACM Transactions on Database Systems (TODS)*, 26(1):41–95, 2001.

[DS98] Curtis E. Dyreson and Richard T. Snodgrass. Supporting valid-time indeterminacy. *ACM Transactions on Database Systems*, 23(1):1–57, March 1998.

[DS04] Mike Dean and Guus Schreiber. OWL Web Ontology Language Reference. Recommendation, W3C, February 2004.

[Dut88] S. Dutta. An Event-based Fuzzy Temporal Logic. In *Proc. 18th IEEE Intl. Symp. on Multiple-Valued Logic*, pages 64–71, Palma de Mallorca, Spain, 1988.

[DW04] John Davies and Richard Weeks. QuizRDF: Search Technology for the Semantic Web. In *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS-37 2004)*, 2004.

[EJS98] Opher Etzion, Sushil Jajodia, and Suryanarayana Sripada, editors. *Temporal databases: research and practice*, volume 1399 of *Lecture Notes in Computer Science*, New York, NY, USA, 1998. Springer-Verlag Inc.

[Fel70] William Feller. *An Introduction to Probability Theory and Its Applications, Vol. 1 & 2*. Wiley, 1970.

[Fer03] Reginald Ferber. *Information Retrieval. Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. Dpunkt Verlag, March 2003.

[Fit96] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Graduate texts in computer science. Springer-Verlag, New York, 1996.

[FLGP02] Mariano Fernández-López and Asunción Gómez-Pérez. OntoWeb Deliverable 1.4: A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies. Technical report, EU IST Project IST-2000-29243 OntoWeb, 2002.

[FMG05] Blaz Fortuna, Dunja Mladenic, and Marko Grobelnik. Semi-Automatic Construction of Topic Ontology. In *Conference on Data Mining and Data Warehouses (SiKDD 2005)*, October 2005.

[FMJ$^+$05] Tim Finin, James Mayfield, Anupam Joshi, R. Scott Cost, and Clay Fink. Information Retrieval and the Semantic Web. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, 2005.

[Fow04] Martin Fowler. Inversion of Control Containers and the Dependency Injection pattern. Technical report, ThoughtWorks, 2004. Available from `http://www.martinfowler.com/articles/injection.html`.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, Massachusetts, 1994.

[GL06] Domenico Gendarmi and Filippo Lanubile. Community-Driven Ontology Evolution Based on Folksonomies. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4277 of *LNCS*, pages 181–188. Springer, 2006.

[GMM03] R. Guha, Rob McCool, and Eric Miller. Semantic search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 700–709, New York, NY, USA, 2003. ACM Press.

[GP97] Asunción Gómez-Pérez. *Handbook of Applied Expert Systems*, chapter Knowledge Sharing and Reuse. CRC Press, 1997.

[GP01] Asunción Gómez-Pérez. Evaluation of ontologies. *International Journal of Intelligent Systems*, 16(3):391–409, 2001.

[GPFLC04] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Advanced Information and Knowledge Processing. Springer, 1st edition, 2004.

[Gru93] Thomas Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[GV95] Lluis Godo and Lluis Vila. Possibilistic Temporal Reasoning Based on Fuzzy Temporal Constraints. In Chris Mellish, editor, *IJCAI'95: Proceedings International Joint Conference on Artificial Intelligence*, Montreal, aug 1995.

[GW02] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, February 2002.

[HH00] James Hendler and Jeff Heflin. Searching the Web with SHOE. In *Artificial Intelligence for Web Search. Papers from the AAAI Workshop.*, pages 35–40. AAAI Press, May 2000.

[HMS04] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing SHIQ-Description Logic to Disjunctive Datalog Programs. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 152–162, 2004.

[HMSS01] Andreas Hotho, Alexander Maedche, Steffen Staab, and Rudi Studer. SEAL-II - The Soft Spot between Richly Structured and Unstructured Knowledge. *The Journal of Universal Computer Science*, 7(7):566–590, 2001.

[HSC02] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-CREAM - Semi-automatic CREAtion of Metadata. In Asunción Gómez-Pérez and V. Richard Benjamins, editors, *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002*, volume 2473, pages 358–372. Springer, 2002.

[HT04] Andrew Hunt and David Thomas. *Unit-Tests mit JUnit*. Hanser, 2004.

[Hun03] John Hunt. *Guide to the Unified Process featuring UML, Java and Design Patterns*. Springer Professional Computing. Springer Verlag, September 2003.

[JBCV98] Dean Jones, Trevor Bench-Capon, and Pepijn Visser. Methodologies for Ontology Development. In *Proc. IT&KNOWS Conference, XV IFIP World Computer Congress*, Budapest, Hungary, August 1998.

[JDB$^+$98] Christian S. Jensen, Curtis E. Dyreson, Michael Böhlen, James Clifford, Ramez Elmasri, Shashi K. Gadia, Fabio Grandi, Pat Hayes, Sushil Jajodia, Wolfgang Käfer, Nick Kline, Nikos Lorentzos, Yannis Mitsopoulos, Angelo Montanari, Daniel Nonen, Elisa Peressi, Barbara Pernici, John F. Roddick, Nandlal L. Sarda, Maria Rita Scalas, Arie Segev, Richard T. Snodgrass, Mike D. Soo, Abdullah Tansel, Paolo Tiberio, and Gio Wiederhold. The Consensus Glossary of Temporal Database Concepts — February 1998 Version. *Lecture Notes in Computer Science*, 1399:367–405, 1998.

[JP01] Bernard J. Jansen and Udo W. Pooch. A review of Web searching studies and a framework for future research. *Journal of the American Society for Information Science and Technology*, 52(3):235–246, 2001.

[JS03] Bernard J. Jansen and Amanda Spink. An Analysis of Web Documents Retrieved and Viewed. In *International Conference on Internet Computing*, pages 65–69, 2003.

[JSS00] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing & Management*, 36(2):207–227, 2000.

[KC05] Pawel Jan Kalczynski and Amy Chou. Temporal Document Retrieval Model for business news archives. *Information Processing & Management*, 41(3):635–650, May 2005.

[Kim96] Ralph Kimball. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. John Wiley & Sons, 1996.

[KKPS02] José Kahan, Marja-Riitta Koivunen, Eric Prud'Hommeaux, and Ralph R. Swick. Annotea: An Open RDF Infrastructure for Shared Web Annotations. *Computer Networks*, 39(5):589–608, August 2002.

[KL05] Bo-Yeong Kang and Sang-Jo Lee. Document indexing: a concept-based approach to term weight estimation. *Information Processing & Management*, 41(5):1065–1080, September 2005.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[KPT$^+$05] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic Annotation, Indexing, and Retrieval. *Journal of Web Semantics*, 2(1):49–79, 2005.

[Kur95] Werasak Kurutach. Modelling fuzzy interval-based temporal information: a temporal database perspective. In *Proceedings of 1995 IEEE International Conference on Fuzzy Systems*, pages 741–748, Yokohama, Japan, March 1995.

[Kur04] Dominik Kuropka. *Modelle zur Repräsentation natürlichsprachlicher Dokumente. Ontologie-basiertes Information-Filtering und -Retrieval mit relationalen Datenbanken*. Advances in Information Systems and Management Science. Logos Verlag, Berlin, 2004.

[Kur05] Dominik Kuropka. Uselessness of simple co-occurrence measures for IF&IR – a linguistic point of view. In *Proceedings of the 8th International Conference on Business Information Systems*, Poznan, Poland, 2005.

[LGPSS99] M. F. Lopez, A. Gomez-Perez, J. P. Sierra, and A. P. Sierra. Building a chemical ontology using Methontology and the Ontology Design Environment. *IEEE Intelligent Systems*, 14(5):37–45, January/February 1999.

[Mae02] Alexander Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academic Publishers, 2002.

[MC04] Donald Metzler and W. Bruce Croft. Combining the language model and inference network approaches to retrieval. *Information Processing & Management*, 40(5):735–750, 2004.

[McG03] Deborah L. McGuinness. Ontologies Come of Age. In Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pages 171–194. MIT Press, 2003.

[ME99] Philippe Martin and Peter Eklund. Embedding Knowledge in Web documents. In *Proceedings of the Eighth International World Wide Web Conference*, pages 325–341, Toronto, Canada, May 1999. Elsevier.

[MM04] Frank Manola and Eric Miller. RDF Primer. Recommendation REC-rdf-syntax-19990222, World-Wide Web Consortium, February 2004. Available from `http://www.w3.org/TR/rdf-primer/`.

[MMSW93] Andreas Mayer, Bernhard Mechler, Andreas Schlindwein, and Rainer Wolke. *Fuzzy Logic: Einführung und Leitfaden zur praktischen Anwendung; Mit Fuzzy-Shell in C++*. Addison-Wesley, 1993.

[MMV02] Boris Motik, Alexander Maedche, and Raphael Volz. A Conceptual Modeling Approach for Semantics-Driven Enterprise Applications. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 1082–1099, London, UK, October 2002. Springer-Verlag.

[MS91] E. McKenzie and R. Snodgrass. An Evaluation of Relational Algebras Incorporating the Time Dimension in Databases. *ACM Computing Surveys*, 23(4):501–543, December 1991.

[MSSV02] Alexander Maedche, Steffen Staab, Rudi Studer, and Raphael Volz. SEAL - Tying Up Information Integration and Web Site Management by Ontologies. *IEEE Data Engineering Bulletin*, 25(1), March 2002.

[Mv04] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. Recommendation, W3C, February 2004. Available from `http://www.w3.org/TR/owl-features/`.

[Nag04] Gábor Nagypál. Creating an Application-Level Ontology for the Complex Domain of History: Mission Impossible? In *Proceedings of Lernen - Wissensentdeckung - Adaptivität (LWA 2004), FGWM 2004 Workshop*, pages 287–294, Berlin, Germany, October 2004.

[Nag05] Gábor Nagypál. Methodology for building SWS ontologies in DIP. Delievarable D3.11, DIP Consortium, June 2005.

[Nag07] Gábor Nagypál. *Semantic Web Services. Concepts, Technologies, and Applications*, chapter Ontology Development, pages 75–96. Springer, 2007.

[NDO05] Gábor Nagypál, Richard Deswarte, and Jan Oosthoek. Applying the Semantic Web – The VICODI Experience in Creating Visual Contextualization for History. *Literary and Linguistic Computing*, 20(3):327–349, 2005.

[NM03] Gábor Nagypál and Boris Motik. A Fuzzy Model for Representing Uncertain, Subjective, and Vague Temporal Knowledge in Ontologies. In Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, volume 2888 / 2003 of *Lecture Notes in Computer Science*, pages 906 – 923. Springer-Verlag, 2003.

[Noy04] Natasha Noy. Representing Classes As Property Values on the Semantic Web. Working draft, W3C, July 2004.

[NW97] Hung T. Nguyen and Elbert A. Walker. *A First Course in Fuzzy Logic*. CRC Press, New York, 1997.

[Ohl04] Hans Jürgen Ohlbach. Relations between fuzzy time intervals. In *Proceedings of the 11th International Symposium on Temporal Representation and Reasoning (TIME 2004)*, pages 44–51, 2004.

[Pea94] Judea Pearl. *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, San Francisco, Calif., 1994.

[PH03] J. Pan and I. Horrocks. Web ontology reasoning with datatype groups. In *In Proceedings of the 2003 International Semantic Web Conference (ISWC 2003)*, volume 2870/2003, pages 47–63. Springer-Verlag, October 2003.

[PKO$^+$03] Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, and Miroslav Goranov. Towards Semantic Web Information Extraction. In *Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC2003)*, Florida, October 2003. Springer Verlag.

[PKO$^+$04] Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, and Angel Kirilov. KIM – a semantic platform for information extraction and retrieval. *Natural Language Engineering*, 10(3-4):375–392, September 2004.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. *Web Ontology Language (OWL) Abstract Syntax and Semantics*. W3C Recommendation, February 2004. Available from `http://www.w3.org/TR/owl-semantics/`.

[Rec04] Alan Rector. Representing Specified Values in OWL: “value partitions” and “value sets”. Working draft, W3C, August 2004.

[RL04] Daniel E. Rose and Danny Levinson. Understanding user goals in web search. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 13–19, New York, NY, USA, 2004. ACM Press.

[RM96] Berthier A. N. Ribeiro and Richard Muntz. A belief network model for IR. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA, 1996. ACM Press.

[RSA04] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi Aragao. A hybrid approach for searching in the semantic web. In *Proceedings of the 13th international conference on World Wide Web (WWW '04)*, pages 374–383, New York, NY, USA, 2004. ACM Press.

[Rum01] Melissie Clemmons Rumizen. *The Complete Idiot's Guide to Knowledge Management*. Alpha, 2001.

[RWB99] S. E. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track. *NIST SPECIAL PUBLICATION SP*, 500(242):253–264, 1999. NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC 7).

[RWRR01] Alan L. Rector, Chris Wroe, Jeremy Rogers, and Angus Roberts. Untangling taxonomies and relationships: personal and practical problems in loosely coupled development of large ontologies. In *Proceedings of the international conference on Knowledge capture*, pages 139–146. ACM Press, 2001.

[SAA$^+$99] Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, Nigel Shadbolt, Walter Van de Velde, and Bob Wielinga. *Knowledge Engineering and Management – The CommonKADS Methodology*. MIT Press, December 1999.

[SAB$^+$03] Y. Sure, H. Akkermans, J. Broekstra, J. Davies, Y. Ding, A. Duke, R. Engels, D. Fensel, I. Horrocks, V. Iosif, A. Kampman, A. Kiryakov, M. Klein, Th. Lau, D. Ognyanov, U. Reimer, K. Simov, R. Studer, J. van der Meer, and F. van Harmelen. On-To-Knowledge: Semantic Web Enabled Knowledge Management. In N. Zhong, J. Liu, and Y. Yao, editors, *Web Intelligence*, pages 277–300. Springer-Verlag, 2003.

[Sal86] Gerard Salton. Another look at automatic text-retrieval systems. *Communications of the ACM*, 29(7):648–656, 1986.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[SBA$^+$02] Amit Sheth, Clemens Bertram, David Avant, Brian Hammond, Krysztof Kochut, and Yashodhan Warke. Managing Semantic Content for the Web. *IEEE Internet Computing*, 6(4):80–87, 2002.

[Sch05] Steven Schockaert. Construction of Membership Functions for Fuzzy Time Periods. In Judit Gervain, editor, *Proceedings of the Tenth ESSLLI Student Session*, pages 297–305, 2005.

[SCK] Steven Schockaert, Martine De Cock, and Etienne E. Kerre. Fuzzifying Allen's Temporal Interval Relations. *IEEE Transactions on Fuzzy Systems*. to appear.

[SCK05] Steven Schockaert, Martine De Cock, and Etienne E. Kerre. Imprecise Temporal Interval Relations. In *International Workshop on Fuzzy Logic and Applications (WILF 2005)*, Crema, Italy, September 2005.

[SdVM06] Peter Spyns, Aldo de Moor, Jan Vandenbussche, and Robert Meersman. From Folksologies to Ontologies: How the Twain Meet. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, volume 4275 of *LNCS*, pages 738–755. Springer, 2006.

[SFJ$^+$02] Urvi Shah, Tim Finin, Anupam Joshi, R. Scott Cost, and James Mayfield. Information retrieval on the Semantic Web. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 461–468, New York, NY, USA, 2002. ACM Press.

[SJ04] Amanda Spink and Bernard J. Jansen. A study of Web search trends. *Webology*, 1(2):Article 4., 2004. Available from `http://www.webology.ir/2004/v1n2/a4.html`.

[Sme96] Philippe Smets. Imperfect Information: Imprecision and Uncertainty. In *Uncertainty Management in Information Systems*, pages 225–254. Kluwer Academic Publishers, 1996.

[SMH01] Steffen Staab, Alexander Maedche, and Siegfried Handschuh. An Annotation Framework for the Semantic Web. In S. Ishizaki, editor, *Proc. of The First International Workshop on MultiMedia Annotation, Tokyo, Japan*, January 2001.

[Sow00] John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[SRN04] Maria L. Silveira and Berthier Ribeiro-Neto. Concept-based ranking: a case study in the juridical domain. *Information Processing & Management*, 40(5):791–805, September 2004.

[SS02] York Sure and Rudi Studer. On-To-Knowledge Methodology. On-To-Knowledge Deliverable 18, AIFB, University of Karlsruhe, 2002. Available from `http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/OTK-D18_v1-0.pdf`.

[SSS03] Nenad Stojanovic, Rudi Studer, and Ljiljana Stojanovic. An Approach for the Ranking of Query Results in the Semantic Web. In *ISWC 2003*, pages 500–516, October 2003.

[Sto05] Nenad Stojanovic. *Ontology-based Information Retrieval: Methods and Tools for Cooperative Query Answering*. PhD thesis, University of Karlsruhe, Universität Karlsruhe (TH), Institut AIFB, D-76128 Karlsruhe, 2005.

[Sur03] York Sure. *Methodology, tools & case studies for ontology based knowledge management*. PhD thesis, University of Karlsruhe, May 2003.

[SW01] Barry Smith and Christopher Welty. FOIS introduction: Ontology — towards a new synthesis. In *Proceedings of the international conference on Formal Ontology in Information Systems*, pages 3–9. ACM Press, 2001.

[Ull88] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, Rockville, MD, 1988.

[vB98] Bas van Bakel. Modern Classical Document Indexing: A Linguistic Contribution to Knowledge-Based IR. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pages 333–334. ACM, 1998.

[VFC05] David Vallet, Miriam Fernández, and Pablo Castells. An Ontology-Based Information Retrieval Model. In *The Semantic Web: Research and Applications: Second European Semantic Web Conference, ESWC 2005*, volume 3532 of *Lecture Notes in Computer Science*, pages 455–470, Heraklion, Crete, Greece, 2005. Springer.

[Vil94] Luis Vila. A Survey on Temporal Reasoning in Artificial Intelligence. *AICOM (Artificial Intelligence Communications)*, 7(1):4–28, 1994.

[vPH01] Frank van Harmelen, Peter F. Patel-Schneider, and Ian Horrocks. DAML+OIL (March 2001) Reference Description. Technical report, DARPA, March 2001. Available from `http://www.daml.org/2001/03/reference`.

[vSW97] Gertjan van Heijst, A. Th. Schreiber, and Bob J. Wielinga. Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*, 46(2-3):183–292, 1997.

[WW97] Misha Wolf and Charles Wicksteed. Date and Time Formats. W3C Note, September 1997.

[YN06] Ümit Yoldas and Gábor Nagypál. Ontology Supported Automatic Generation of High-Quality Semantic Metadata. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, volume 4275 of *LNCS*, pages 791–806. Springer, 2006.

[Yol06] Ümit Yoldas. Konzeption und Implementierung eines ontologiebasierten Informationsextraktionssystems. Master's thesis, University of Karlsruhe, 2006. tutored by: Gábor Nagypál and Prof. Dr. P. C. Lockemann.

[Zad65] Lotfi A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[Zad78] Lotfi Zadeh. Fuzzy Sets as a Basis for Possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[Zhe04] Wang Zheng. Effiziente Verarbeitung und Benutzerfreundliche Darstellung ungenauer Zeitintervalle. Master's thesis, University of Karlsruhe, 2004. tutored by Gábor Nagypál and Prof. Dr. P. C. Lockemann.

[ZYZ$^+$05] Lei Zhang, Yong Yu, Jian Zhou, ChenXi Lin, and Yin Yang. An enhanced model for searching in semantic portals. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 453–462, New York, NY, USA, 2005. ACM Press.

Ontologies and semantic metadata can theoretically solve all problems of traditional full-text search engines. In practice, however, ontologies and semantic metadata are always imperfect. They may miss facts, contain erroneous information, and sometimes even our knowledge of the world that should be represented in the ontology is imperfect. Moreover, the high complexity of ontology reasoning makes this technology hard to use in large-scale information retrieval (IR) systems, where performance is of paramount importance.

This work had pursued two goals. First, it provided techniques to decrease the negative effects of imperfection, and to make ontology reasoning applicable for large-scale IR. The provided solutions include among others a novel fuzzy temporal model, an IR system that combines full-text search with semantic search, and methods for completely automatic semantic metadata extraction from unstructured textual documents.

Second, this work analyzed whether the negative effect of the inherent ontology imperfection has a higher impact than the positive effect of exploiting the ontology features for IR. To answer this question, a complete ontology-based information retrieval system based on the previously devised solutions was implemented and thoroughly evaluated. During this evaluation, the retrieval performance of the new system was compared with a baseline full-text search engine.

The evaluation results show that even imperfect ontologies can dramatically increase the quality of results, if all ontology features are exploited, including ad-hoc, non-taxonomical relations, and temporal information.

ISBN 978-3-86644-190-3

www.uvka.de