Search Relevance based on the Semantic Web

Veli Bicer

Karlsruhe Institute of Technology

Search Relevance based on the Semantic Web

Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften

(Dr. rer. pol.)

von der Fakultät für Wirtschaftswissenschaften
des Karlsruher Institut für Technologie (KIT)

genehmigte

DISSERTATION

von

M.Sc. Veli Bicer

Tag der mündlichen Prüfung:	23.02.2012
Referent:	Prof. Dr. Rudi Studer
Koreferent:	Prof. Dr. Wolfgang Nejdl
Prüfer:	Prof. Dr. Detlef Seese
Vorsitzender:	Prof. Dr. Wolf Fichtner

2012 Karlsruhe

To Ada

Abstract

The amount of data stored and shared as well as the active number of users is continuously increasing on the Web. When that large number of people meet with that large amount of information on the Web, finding the right information in such a quite large scale is definitely a difficult task. Remarkable progress has been made over the last decade with the help of search technologies applied on the Web that are able to collect, maintain and filter the information according to users' needs. Search relevance is a core topic in Web search and directly affects the search performance. Several proposals are made in this regard employing different techniques to represent, model and measure relevance of a particular search result to the users.

In this thesis, we explore the challenge of search relevance in the context of semantic search. We firstly introduce a general framework for relevance as it is understood in the context of Web search which allows us to compare different notions of relevance that exist in literature. Specifically, the notion of semantic relevance can be distinguished from the other types of relevance in Information Retrieval (IR) in terms of employing an underlying semantic model. We propose the emerging Semantic Web data on the Web which is represented in RDF graph structures as an important candidate to become such a semantic model in a search process. However, the semantic gap between the structured representation of this type of data and the textual content is a major challenge for its applicability to Web search. For this purpose, we propose a probabilistic generative model to be trained from existing datasets of the Semantic Web data in order to bridge this semantic gap.

Based on this model, we introduce novel ranking models by extending long-studied IR-based models that have been widely applied to the Web search. Specifically we present a semantic relevance model that can model and determine relevance based on the semantic data by exploiting existing datasets to improve retrieval performance. In fact using Semantic Web data as a relevance source is orthogonal to existing search paradigms and can be used in a variety of contexts. We also show how semantic relevance can be utilized to extend different types of IR-based models (i.e. generative or discriminative) to search different types of data (i.e. unstructured and structured).

Acknowledgements

I am grateful to many people who supported me during this long process of obtaining a Ph.D degree. First and foremost, I am deeply indebted to my Ph.D. supervisor, Prof. Dr. Rudi Studer, who gave me the opportunity to do this research and also the freedom and trust which I definitely needed to complete it successfully and with a joy. In addition, I would like to thank my Ph.D. committee, Prof. Dr. Wolfgang Nejdl, and Prof. Dr. Detlef Seese, for their helpful comments and suggestions.

Furthermore, Thanh Tran made this work possible through his invaluable guidance, insight and encouragement. I owe him a lot, especially for his commitment and patience during our collaborations and also pushing the limits to meet many unlikely deadlines.

Special thanks also go to my colleagues at FZI and AIFB with whom I worked for more than four years. I would like to thank Andreas Abecker, Darko Anicic, Saartje Brockmans, Catherina Burghart, Tobias Conte, Heike Döhmer, Eugenie Giesbrecht, Heiko Haller, Joachim Kleb, Yongtao Ma, Sinan Sen, Tuvshintur Tserendorj, Raphael Volz, Max Völkel, Andreas Wagner, and Jens Wissmann.

I am also grateful to Anna Gossen, Radoslav Nedkov, Ying Xu, Ze Li, and Adrian Philipp who helped me a lot with their ideas and development work as thesis students or assistants at FZI.

As always, I am thankful to my everlasting mentor and teacher, Ali Dogru.

I'm deeply touched by the continuous support of my family, without whom I could not walk that long path of life. I would especially like to thank Kader for her constant love, support, understanding and care for our child. Finally, I'd like to thank my daughter, my source of life, Ada to whom I dedicate this work.

Contents

List of Figures

LIST OF FIGURES

List of Tables

List of Abbreviations

DR	Data Retrieval
i.i.d.	Independent and identically distributed
IR	Information Retrieval
JRT	Joined Resource Tree
LDA	Latent Dirichlet Allocation
LOD	Linked Open Data
LTR	Learning-to-Rank
OWL	Web Ontology Language
RDF(S)	Resource Description Framework (Schema)
SRM	Semantic Relevance Model
SVM	Support Vector Machines
TRM	Topical Relational Model
URI	Uniform Resouce Identifier

1

Introduction

This chapter introduces a general overview of the thesis, focusing on the problem definition that motivate the work, an outline of the proposed solution to address the problem, and the contributions of this research work. Section 1.1 presents a brief introduction to the challenges of search, sketching the limitations of the current state-of-the-art in terms of addressing the relevance problem. Section 1.2 defines the scope of the thesis including main research questions. Section 1.3 briefly mentions the specific contributions of this research. Finally Section 1.4 provides the overall structure of this thesis.

1.1 Motivation

Searching the Web is a very common activity for a large number of people on the planet similar to their other everyday activities. According to Internet World Stats[1], there are currently about 2 billion active Web users, which was a number of only 360 million by Dec. 2000. In parallel to the number of users, the amount of information maintained and shared on the Web is also continuously increasing. When that large number of people meet with that large amount of information on the Web, finding the right information in such a quite large scale is definitely a difficult task and that's why *search engines* become a part of our everyday life. In fact a search engine is an application of Information Retrieval (IR) techniques to large data collections and the Web is the major application domain and also success story for its practical use. By collecting, storing and pre-processing the data, it is able to return information in response to users' specific information needs.

Due to the reliance on the IR, the majority of the search techniques employs a *keyword-based retrieval* paradigm. Despite its robustness and well-founded probabilistic principles, keyword-based search provides limited capabilities to grasp and exploit the actual conceptual information involved in the users' information needs. Besides, the exponential growth of the Web has also introduced a sort of diversity both in terms of the data to be processed and the specific information needs formulated by the users. Today search is not only conducted purely on a keyword basis. A modern search engine utilizes different sorts of data such as user gen-

[1] http://www.internetworldstats.com/

erated content (e.g. blogs, personal profiles), link structure (e.g. Web hyperlink graph, social networks), and other document features (e.g. image, title, tags, cluster). Different techniques are developed incorporating these sources to improve the search effectiveness by delivering more relevant results to the users.

The concept of *relevance* plays a crucial role in this regard, and it is the main task of a search engine to formalize, quantify and measure relevance in order to achieve its purposes. While it seems to be intuitive at the first sight, relevance is nevertheless hard to define and even harder to represent and measure in a well-defined way. Different assumptions are made in this direction. In its simplest case, relevance can be considered between a document and query if the document frequently includes the query terms in its text (i.e. textual relevance) [131, 196], or a document is popular according to the hyperlinks it receives from other documents (i.e. hyperlink relevance) [121, 156, 175]. Further assumptions can also be made to consider, for instance, the relevance according to users' profiles (i.e. personalized relevance) [45, 220], or according to connections or annotations in a social network [22, 66, 86, 173, 239, 258]. No matter how the relevance is modeled, every approach in the search literature presents its unique view about the so-called *relevance source* and acts accordingly to improve the search accuracy.

Semantic search provides another perspective to relevance with the assumption that relevance can be determined by a given semantic model instead of the abovementioned relevance sources. Actually the idea of using underlying semantics as searching by meanings rather than keywords is not relatively new in the IR field. Early attempts in this direction has been made to utilize a thesaurus of concepts as semantic model and aim to achieve a concept-based retrieval instead of using terms [54, 84, 232]. They have been quite successful in terms of disambiguation purposes but limited with respect to understanding actual content of a document since the thesaurus information is restricted to say only about synonymy, hypernyms, hyponymy, or meronymy. Later, the *Semantic Web* community has proposed to use expressive ontologies as semantic model to shift the Web search beyond keyword-based capabilities [115, 149, 154, 216, 224]. Under this perspective, the idea was to exploit large ontology-based knowledge bases (KBs) to transform existing Web data into a more expressive, high-level representation and to exploit this KBs by querying with (semi-)structured queries such as SPARQL. Using this expressiveness, it is a way to obtain more precise results without a need for a probabilistic approximation. However, from a search perspective, this idea is closer to data retrieval instead of IR models because it requires the data in a non-ambiguous, non-redundant, formal representation. In addition, the applicability of converting large volume of Web data, especially text documents, into formal ontological knowledge at an affordable cost is currently not practical which is also known as *knowledge acquisition bottleneck* [191]. Furthermore, such a conversion is definitely bring a big degree of *information loss* since documents hold a value of their own, and are not equivalent to their representation in a KB. In this respect, *semantic annotations* can become an alternative to use semantic model for search as they consider keeping the textual information (i.e. documents) and the KB separated [40, 162, 228], but still having similar bottleneck of the abovementioned SW approaches in terms of preprocessing all documents to be annotated with semantic entities.

Meanwhile, very recent Semantic Web initiative have given rise to publicly sharing "raw

data" on the Web, also known as Linked Open Data (LOD)[1] [24]. Based on the core stack of technologies such as Uniform Resource Identifiers (URI), Resource Description Framework (RDF), RDF Schema (RDFS) and Web Ontology Language (OWL), the trend of publishing and linking Semantic Web data coming from different sources has resulted in large amounts of data to be available on the Web. Billions of RDF triples are now publicly available on the Semantic Web. For example, currently LOD provides a large amount of datasets containing knowledge from different domains including healthcare and life sciences[2], environment[3], music[4], government[5] as well as generic datasets such as DBPedia[6] and Freebase[7]. Collectively, the amount of data published in the scope of the LOD project comprises about 300 datasets containing over 31 billion RDF triples, which are connected by around 500 millions of links by September 2011. These figures was around one-third of the current size in November 2009. In addition to its growing size, another important issue is the nature of the data to be healthy, condensed wealth of information. This is mainly due to several aspects: First, the providers of the data includes important sources ranging from Wikipedia to several government and public agencies all over the World. Secondly, the data is freely accessible and verifiable allowing knowledge to be built into a global database so it can be enriched via crowdsourcing [64]. In the context of Web search, exploiting such Web of data offer a number of opportunities, but also presents several challenges:

1. **Semantic Web data is an important relevance source to improve search performance.** The core of Semantic Web data is RDF which represents the data in a typed graph structure in which entities and relationships are explicitly defined. According to a recent study of the *Yahoo!* Web query log it has been shown that over 70% of all queries contain a semantic resource (entity, type, relation, or attribute) [183, 237]. As the amount of data is quite large in the sense of capturing the World of knowledge, it is an important aspect related to search for understanding what users are looking for. In addition, users' information needs can be diverse. Thus the diversity of the Web data is also an important asset to meet different information needs concerning different domains. On the current Web, this issue is mostly managed by the so-called vertical search engines which offers specialized search capabilities – e.g. prominent examples include scientific literature search [8], medical search [9], patent retrieval [10], environment [11], and book search [12]. In this regard, there is a gap between the expected results of a vertical search engine and generic search engine, because the former utilizes a very tailored techniques according to the particular characteristics of the domain. Web data is an important input to the

[1] http://linkeddata.org/
[2] http://www.linkedct.org/
[3] http://www.geonames.org/
[4] http://musicbrainz.org/
[5] http://www.data.gov/semantic/index
[6] http://dbpedia.org/
[7] http://www.freebase.com/
[8] http://www.scirus.com/
[9] http://www.ncbi.nlm.nih.gov/pubmed/
[10] http://www.google.com/patents
[11] http://www.portalu.de/
[12] http://www.freebooksearch.net/

search process in order to better understand a particular search request and to filter the search results accordingly using the semantic model.

2. **More robust and IR-based techniques are required to incorporate semantic Web data into the search process.** The major techniques search Web data is mainly about structured query processing that is not directly applied to the IR-based search models that we frequently observe in Web search. In practice, IR-based ranking models are successfully applicable by search engines and further improvements are made to improve search relevance. These models are utilizing well-founded probabilistic approaches that aim to obtain high precision within top search results instead of dominant structured query processing paradigm highly common to retrieve more deterministic results from the semantic Web data with the cost of complex queries and data representation requirements. Considering this trade-off, it is why a successful application of the Web data to improve search relevance requires to develop techniques that follow the existing line of work in the IR field in terms of exploiting data as relevance source in the underlying probabilistic models that have been successful for ranking search results so far. However, one of the challenges in this regard is the so-called *semantic gap* between semantic information captured in the Web data and the actual documents on the Web mostly represented as textual data. This is in fact a classic Natural Language Processing (NLP) problem in disguise where the goal is to obtain a more precise understanding of text and to represent it in a way that can be processed by machines (e.g. as structured data) [155]. However, such an approach is not feasible in the context of Web search due to its lack of robustness of the NLP techniques. Instead, our aim is to represent and model the relevance by utilizing the Web data and to construct a retrieval model that ranks the text-based documents w.r.t. this type of relevance. Actually, the Web data is also useful in this regard because it comes with a large amount of textual data appearing as attributes in an RDF graph and can be used to *train a machine learning model* to fill the semantic gap between the structured and unstructured data.

3. **A semantic search approach based on the semantic Web data needs to be comparable with existing IR-based methods and provide similar probabilistic foundation to measure relevance.** There are several ranking models proposed in the IR field to provide an estimation for relevance. Probabilistic ranking models, such as BM25 [197] or language models [180], view relevance as an explicit random variable in a probabilistic model and provide a measurement based on the probability of a document to generate a query. Such a probabilistic approach is crucial in the context of Web search since there are a lot of search results to be retrieved and not every one of those is equivalently relevant to the query. Therefore a semantic search approach needs to be compliant and based on the same underlying probabilistic foundation of the IR-based models. In addition, IR models have traditionally been compared against each other using standard sets of queries and corpora. Due to characteristic differences, most of semantic search methods could not be compared with the IR-based methods saying so little about their effectiveness on improving search relevance.

4. **Semantic Web data is also a source of information to answer queries for users'**

information needs. Since the Web data is growing fast, it provides an alternative to the documents on the Web as a target of search. In fact, the techniques that employ keyword search capabilities on structured data have become an area of interest very recently and offer an intuitive way for ordinary Web users [251]. In addition, similar techniques have already been applied by the major Web sites to query their databases and widely used by the user on a daily basis. However, relevance of the structured search results as response to keyword queries is still in question because searching the data with this type of queries is relatively new field and only a few approaches exists in this direction. In this thesis, we propose that similar principles of relevance applied for document retrieval can also be applied to improve the relevance while searching semantic Web data.

1.2 Research Questions and Aim

The research problem addressed in this thesis can be briefly stated as follows:

Search relevance is the core challenge in the context of Web search and over the years several improvements have been made. Mainstream search models are based on plain keywords and do not consider semantic data as a relevance source. In addition, existing semantic search models do not offer practical solutions to utilize emerging semantic Web data in the widely known search settings due to their lack of robustness and knowledge acquisition bottleneck.

Therefore, this thesis further expands the research problem above into the following specific research questions:

- *Q1: Can emerging semantic Web data in RDF be utilized as a source of information to improve search relevance?*
  Different approaches applied in the context of Web search have been reviewed as "current approaches for relevance" and a generic sketch of relevance as understood in the IR field is drawn. An important research question of this thesis is to determine if semantic Web data is also useful for representing relevance and to further derive ranking models based on such a representation.

- *Q2: How to close the "semantic gap" between structured representation of the semantic Web data and widely used textual representation of Web documents?*
  With the aim of measuring relevance of a textual document, a probabilistic interpretation of the available semantic Web data is needed. In fact, existing datasets can be directly used to train such a probabilistic interpretation using machine learning techniques. This thesis researches a probabilistic model to be utilized to bridge the "semantic gap" to calculate the degree of relevance of a document w.r.t. selected relevance representations on the semantic Web data.

- *Q3: Can the search relevance based on the semantic Web data be incorporated into IR-based ranking models?*
  IR-based ranking models (e.g. probabilistic ranking models) are widely used in the IR

field to measure textual relevance and have a strong probabilistic foundation. We consider that semantic relevance is orthogonal to the classical retrieval problem and different ranking models can be extended to utilize semantic relevance using the Web data. Instead of proposing a new ranking model or to adopt query processing in some of the existing semantic search mechanisms, in this thesis we aim to research how existing well-founded models of the IR can be extended to use the semantic Web data.

- *Q4: Can the semantic Web data be also utilized to answer user queries as a response to their information need?*
  The Semantic Web data definitely contains useful information for the users and search techniques to access this information in an intuitive way to the user is already in place. However, determining the relevance of search results in this type of data is currently sub-optimal. We hypothesize that similar principles of seach relevance applied for document retrieval in the IR-field can also be used to extend the search functionality towards more precise and relevant results to users' actual information need.

Starting from the abovementioned problem statement and research questions, the focus of this thesis lies in providing a principled perspective and underlying techniques to utilize emerging Semantic Web data to improve search relevance. To this end, we present our relevance-based semantic search architecture and show on the basis of concrete approaches and search applications how semantic data can be exploited successfully throughout the search process. We show how it can be used orthogonally to extend some cutting-edge IR-based and also data retrieval models.

1.3 Contribution

Our contribution falls into three major categories:

- **Study and comparison of different views and approximations of search relevance in the field of IR and identify the notion of relevance from a semantic search point of view.** Despite the large number of approaches, search relevance has been a topic of interest due to the increasing data on the Web and fundamental limitations in the state of the art. Existing semantic search approaches either employ techniques to improve particular IR tasks (e.g. query expansion) or consider radically new paradigms apart from existing relevance models in the IR. In this work, we study the strengths and weaknesses of different proposals towards search relevance from both the IR and the semantic search fields and propose a conceptual framework on how to understand the relevance from a semantic search point of view.

- **Bridging the semantic gap in the context of Web search.** Existing semantic search approaches have made certain assumptions about the semantic gap between the textual representation and semantic model either considering that the keywords directly match the elements of the semantic model (i.e. thesaurus), data is transformed into and represented in a high-level ontology representation, or documents are tagged with semantic

annotations. We propose that another effective way of closing the semantic gap is possible by using already existing Semantic Web data for training a probabilistic model that can be further used to measure the degree of semantic relevance.

- **Definition and realization of novel semantic relevance models.** As introduced before, relevance can be determined based on the semantic data by exploitation of emerging datasets to improve the retrieval performance. Such a relevance source is orthogonal to existing search paradigms and can be used in a variety of contexts. We show how semantic relevance can be utilized to extend different types of IR-based models (i.e. generative or discriminative) to search different types of data (i.e. unstructured and structured).

- **Experimental evaluation and application to real-world scenarios.** For the proposed approaches presented in this thesis, we provide detailed experiment results in comparison to the existing IR models and approaches. In addition, system implementations of the research work are successfully integrated into some commercial applications. In particular, in collaboration with colleagues from disy Informationsysteme GmbH, the approach presented in Chapter 6 is successfully integrated with the Cadenza environmental information system to enable intuitive access to the environmental data. In addition, the relational learning component presented in Chapter 7 is utilized within the Information Workbench application from the FluidOps corporation in order to enable learning from graph structured RDF data.

1.4 Overview of the Thesis

This thesis has been divided into seven chapters. In Chapter 2, we present a brief overview of the IR field, particularly in the context of Web search. The chapter also describes the main IR-based models for both document and data retrieval. We also present an overview of some important machine learning approaches as another field that is crucial in the context of Web search and also for the work presented in the upcoming chapters.

In Chapter 3 we explore the concept of relevance, both in general as studied in IR, and in the scope of semantic search where relevance is determined according to the semantic content. We begin with discussing a unified view of relevance that has been developed in the IR field over the years and provides a conceptual explanation to the notion of relevance. Then, we explore the relevance in terms of its interpretations and implementations in practice. We categorize the relevance into five major types depending on the criteria of the source of information to represent and measure relevance. In addition, we present a framework of relevance as it is understood in the unified view and also applied in practice. We then present a semantic search view of relevance and review major approaches in semantic search in this direction. Last but not least, we present two hypothesizes for semantic relevance based on the Web data that is further explored while developing the novel ranking methods of this thesis.

In Chapter 4 we introduce our probabilistic model, called Topical Relational Model (TRM), that is trained by using an existing Semantic Web dataset in RDF and aims to bridge the semantic gap between the structured and textual representation. TRM particularly addresses the problem in a probabilistic semantic interpretation of structured data and relies on machine

learning techniques to construct such a model. By using the RDF structure and available textual data in a dataset, it trains a topic model that is able to assign probabilities to individual words from a trained probability distribution. In addition, TRM also detects the topical correlations between the structured representation and the learned topics and weights them according to the topical distance between these two separate worlds. In brief, it is a principled way to mine already existing data and provide a probabilistic model that can further be used in a number of tasks we develop in the further chapters.

In Chapter 5, we present our first ranking model for document retrieval that utilizes semantic data for relevance. In particular, we study how these data can be used to infer the information need and how the IR-based relevance model approach can be extended in a way to employ semantic data to construct a query model. To this end, we propose novel methods to sample from semantic data to obtain a more fine-grained relevance model based on semantic entities. Instead of using the entities directly, we employ TRM derived from the semantic data, and represent the query as a mixture of entity-related topics. Since this new model is based on an IR-based paradigm of relevance model [133], we obtain a model comparable with other IR-based models that can be evaluated in similar settings. That's why we conduct the experiments using TREC collections and show that our approach can improve the existing IR-based baselines. The main contribution of this work is a robust and effective way to exploit semantic data for classical Web document retrieval.

In Chapter 6 we turn our focus into another type of search where the search results are not documents anymore but structured results from an underlying database. We extend existing keyword query processing over structured data approaches that have gained a lot of interest as keywords have proven to be an intuitive mean for accessing information and the amount of available structured data increases rapidly. In this regard, we propose a novel model that utilizes the semantic structure of underlying data to create the so-called *edge-specific relevance model* that is able to rank the structured search results with a greater relevance accuracy to the user queries.

Chapter 7 focuses on the ranking problem from another angle with a discriminative way of ranking by using the semantic information available in RDF data to decide whether a document is relevant to a query or not. In particular, Learning-to-rank (LTR) approaches recently offer a novel way of ranking in the IR that learns a ranking function from a given training data. However, the effectiveness of any LTR approach highly depends on the features extracted from the training data, and usually constructed based on conventional features. Thus, in this chapter we show how to utilize semantic data to better describe the documents instead of conventional features and train a ranking function that can measure the relevance. However one challenge here is the learning from the semantic data which is denormalized graph structure and existing learning techniques are difficult (if not impossible) to be applied in this setting. That's why a novel relational learning technique is developed and presented in this chapter which applies multiple kernel learning on the RDF-based structured data. Finally in Chapter 8 we make a summary of our contributions and final remarks.

In every chapter we try to make the content as self-contained as possible. Thus, every chapter can be considered as an independent piece that can be read in isolation. In addition, certain details about the background of the work is introduced in Chapter 2 and 3 and it is suggested

that the readers to follow these two chapters before reading the following text. Explicit references are also made between the chapters when we reuse some parts of the work introduced in other chapters.

2

Background

This chapter of the thesis presents an overview of the major concepts and practices in the course of search systems. In particular it begins with a general definition of Information Retrieval and its major distinctions from data retrieval. Further, we review the major lines of approaches particularly in two parts: The first part reviews the approaches of Web search from a document retrieval point of view, while in the second part the major line of approaches are presented from a data retrieval point of view. We also present some important Machine Learning approaches that are crucial in the context of Web search and also for the work presented in the upcoming chapters.

2.1 Information Retrieval vs. Data Retrieval

The work presented in this thesis is a part of the broad field of *Information Retrieval* (IR). Although Web search, a common application of IR, is highly known and associated with IR, the actual term is very broad and takes place in different forms (e.g. enterprise search, desktop search etc.). Despite its long history and advance developments, the general definition of Gerard Salton in his classic 1968 textbook [200] is still up-to-date and valid:

"Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information"

Here, the term information is very general, and IR includes approaches on a wide range of types of information. At its infancy of the field, the word information primarily meant *text*, specifically the kinds of text one might find in a library. This is, in fact, one of the reasons why some IR pioneers made a sharp distinction between IR and *Data Retrieval* (DR) in the early works [9, 229]. However, today information exists in many more forms, often quite different from library documents or Web pages, which were the solely focus of early retrieval systems. Despite the fact that text information is perhaps the most famed application of IR still today, recently we also find different sorts of information emerging on the Web. For example, the emerging Web databases, multimedia content (e.g. images, videos, maps), or even text documents combined with metadata blurs such a distinction between IR and DR. In section

2.2.3, we will elaborate that accessing those different types of information (i.e. structured data) is an extension to IR, but before that it is useful to recall the difference between IR and DR [9, 229]:

- In DR we normally search for an exact match of an information item whether it is present in the database or not. In IR, instead of such a retrieval, more generally we want to find those items which partially match the information need and then mostly select from only a few (top-k) of the best ones.

- DR actually depends on a more deductive type of inference to retrieve the information whereas in IR it is far more common to use inductive inference in the sense that it is specified with a degree of uncertainty and hence our confidence in the inference is variable. Such a distinction makes DR a more deterministic approach, while IR is considered as a more probabilistic approach.

- One important distinction is also the classification of the information because DR approaches mostly tend to classify the information in terms of the classes (e.g. distinct clusters) each of which possesses particular attributes and relationships that are applicable to every instance of that class. In IR such a classification either does not exist or is vaguely defined since the information mostly possesses only a proportion of all the attributes possessed by all the members of that class, and hence no attribute is necessary nor sufficient for membership to a class. Therefore, ontological descriptions are mostly common in DR.

- The query language for DR will generally be of the artificial kind, one with restricted grammar and vocabulary, while in IR we prefer to use natural language although there are some notable exceptions. In DR the query is generally an explicit specification of information need, while in IR it is invariably incomplete and ambiguous. That's why some sorts of querying is relatively difficult in IR and the extent of the match in IR is assumed to indicate the likelihood of the relevance of that item. For example, it is easy to "find the accounts with number 1234000" while it is difficult to reach deterministic results such as finding the accounts with balance greater than 1234000.

Such differences have resulted in a bias in both fields of research in a way that IR has mostly focused on more lightweight but scalable approaches, while DR is usually considered to be about more deterministic way of retrieving the data. However, in the last ten years, such a distinction has started to be blurred mainly due to recent developments emerging from both directions: First, IR-style search became a crucial requirement of data retrieval resulting in techniques of *keyword-search on structured data* as more and more textual data becomes part of databases [251]. The structured data becomes more available on the Web (which is the main playground of IR) in the form of Web databases, linked open data etc. making the retrieval on this data a major requirement [24]. Moreover, such Web of data is not only important as a main source of information, but also as a global semantic model to improve the capabilities of the traditional search [72, 228].

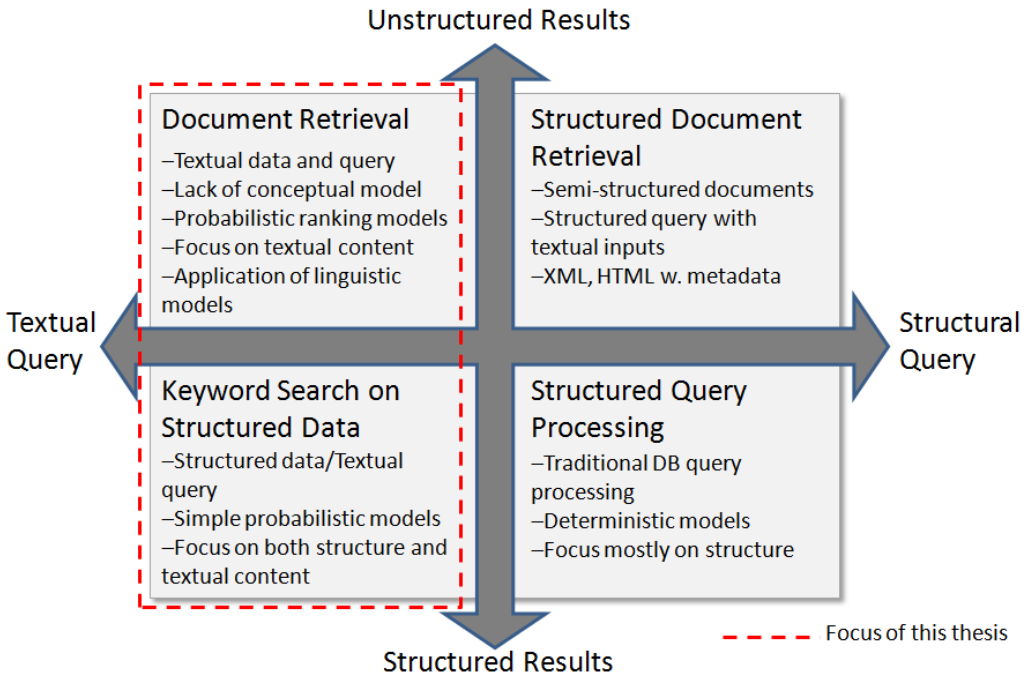


Figure 2.1: Comparison of retrieval approaches based on query-result dimensions

Fig. 2.1 illustrates such a categorization of the approaches by considering the two dimensions of query and result types. The upper-left part concerns the IR-based approaches where the users' information need is formulated as textual queries and the results are mostly unstructured documents. Over the years, several probabilistic models and improvements are proposed for search and ranking which we briefly summarize in Sec. 2.2.2. However, the majority of such query processing techniques are based on keywords or the use of natural language and therefore provide limited capabilities to grasp and exploit the conceptualizations involved in both users' information needs (i.e. queries) and the actual information (i.e. documents). In fact, in order to determine the level of relevance between textual queries and documents, improvements can be achieved by using the available structured data as a semantic model that can enhance our understanding of a particular search request. However, existing approaches in this direction such as [72, 228] do not provide a well-defined probabilistic model which limits their applicability and robustness to extend the already existing IR models. This will be our core focus in Ch. 5.

On the other hand, the use of textual queries is also intuitive to retrieve structured results that leads to the approaches falling in the category of *keyword-search on structured data* in the lower-left part of Fig. 2.1. Generally speaking those approaches aim to obtain structural interpretations of users' queries and return structured information as results from, e.g., a database. Although much attention has been focused on finding efficient techniques to process keyword queries and to retrieve structured data in these settings, only a small number of dedicated work can be found on the ranking of results and understanding their relevance to the information

need [114, 143, 150]. We review these approaches in Sec. 2.2.3. In addition, those approaches employ top-k query processing techniques to focus only on the best results in order to terminate as early as possible since in the Web setting users are mostly interested in top results instead of retrieving all the results like in traditional database query processing [101]. However, the traditional top-k query processing in relational databases requires the user to specify a ranking function in terms of the relational schema which is difficult to formulate by ordinary Web users for their keyword queries. Instead, IR-based ranking functions are replaced as a ranking function of top-k query processing. That's why ranking becomes crucial element of this type of search and existing approaches offer poor results to fill this gap because they consider only term-based weighting and distance metrics which are not a good candidate to capture the relevance between the query and structured data. In fact, more sophisticated probabilistic models of relevance can be adapted for this purpose and it will will be our core focus in Ch. 6.

The upper-right part of Fig. 2.1 concerns the structured document retrieval which considers the document components of varying granularity (e.g. XML, HTML) and allows users to retrieve document components that are more focused on their information needs. Major examples include book search, XML search, or other multimedia documents [6, 76, 147, 242]. Finally, the lower-right part concerns the traditional database query processing where structural queries are used to retrieve structured results. Although the first two categories of Fig. 2.1 can be related to these last two categories in terms of the types of the data (e.g. structured, unstructured) or the applied techniques (e.g. top-k query processing), we make such a distinction between them in order to better position the discussion in the following for the field of IR. Therefore, hereafter we refer to the upper-left category of approaches as *document retrieval* and the lower-left category of approaches as *data retrieval* unless it is explicitly stated otherwise.

2.2 Searching the Web

Web searching is perhaps the most famed application of IR today, and on the Web we may find relevant information in the form of text, but also as structured data emerging as Web databases. A modern search system must have the capability to find, organize and present to the user all of these very different manifestations of information, because all of them may have some relevance to the user's information need. In the previous section, we presented a categorization of approaches that constitute the core part of a modern search system. In the following, we detail our discussion of major approaches: First, we briefly present the basic elements of a search engine. Then we present the latest approaches for searching the Web in terms of document retrieval and data retrieval perspectives.

2.2.1 Basic Elements of a Search Engine

Basically a search engine can be considered to architecturally have four main components (see Fig. 2.2): Crawler, indexing, ranking, and user interaction components. The crawler collects different sorts of data from the Web according to some prioritization strategies. Different collections may exist out of the crawling process each of which is provided as an input to indexing. The indexing component takes these collections and creates indexes or data structures that en-

able fast search of the documents. Mainly it is the task of the indexing component to transform the data into an internal structure by means of some well known techniques. For example, parsing, stemming, and stop word removal are highly common practices within indexing for textual data.

The user interaction component provides the interface between users and search engine. The input queries are processed (e.g., removing stop words, stemming, etc.) and transformed to index terms that are understandable by the search engine. In addition, query expansion is also widely used technique at this step. Typically, a user submits a query made up of a few words and phrases. Query expansion techniques applied to the original query allow more specific representations of the information need. Another task of the user interaction component is logging which is done by most of the major search engines. They keep the logs of user's interaction (query logs, click information, etc.) which provides valuable information to a number of tasks, especially for learning-to-rank that we discuss later.

The ranking component (or ranker) is a central component and responsible for matching between processed queries and indexed documents. The ranker can directly take the queries and documents as inputs and compute a matching score using some heuristic formulas, and can also extract some features for each query-document pair and combine these features to produce the matching score.

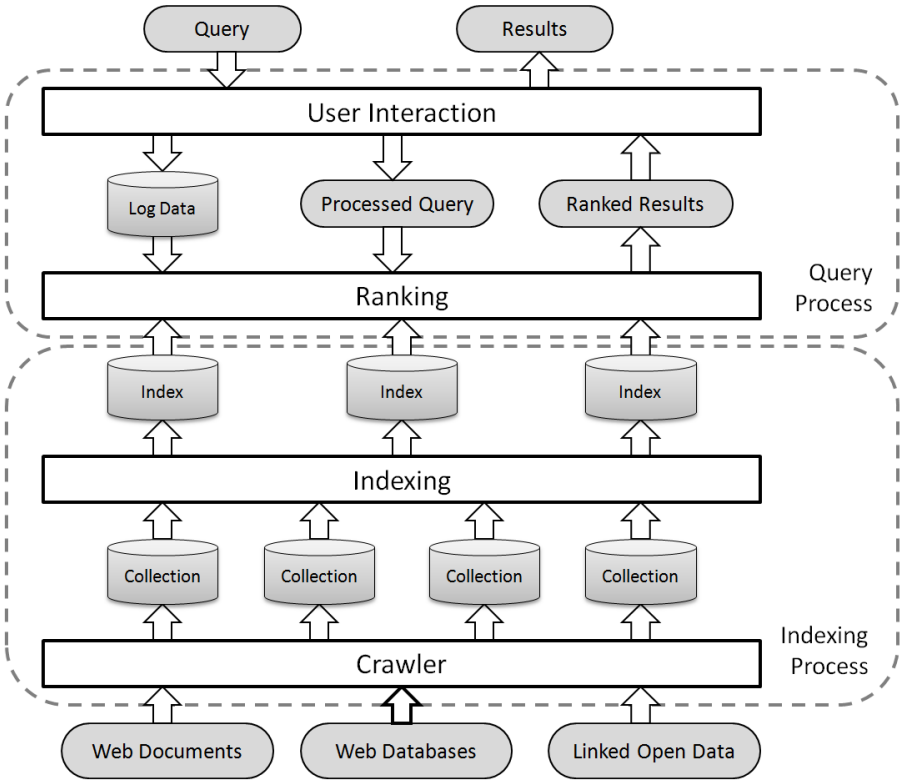


Figure 2.2: A conceptual architecture for a search engine system

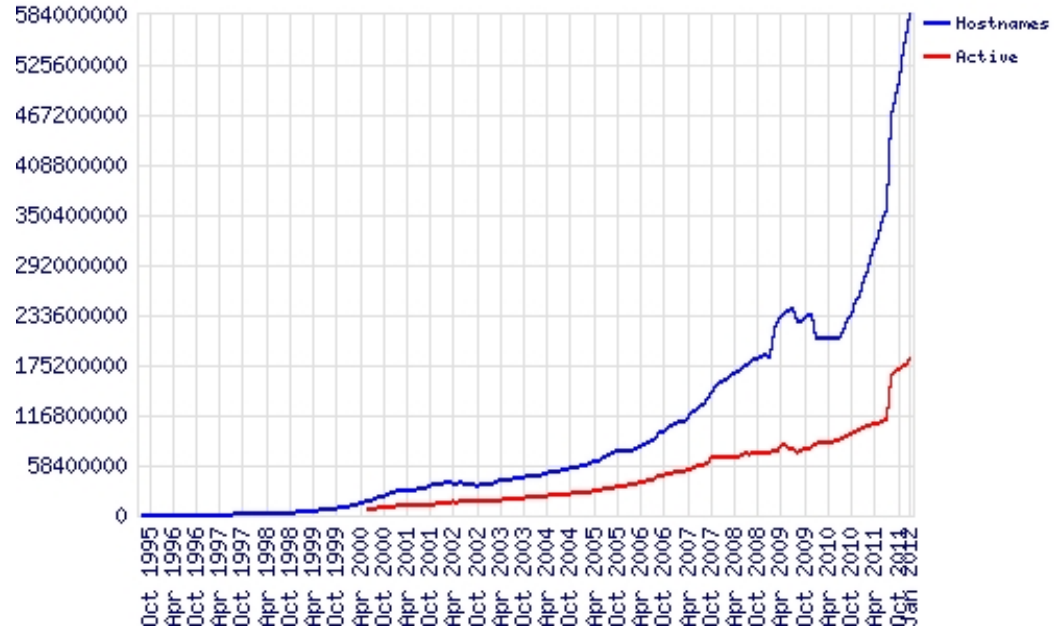


Figure 2.3: Number of Web sites (Source: Netcraft Web server survey (Jan. 2012)).

2.2.2 Web Search: A Document Retrieval Perspective

With the fast development of the Web, users start to experience a flood of information. A study[1] conducted in 2005 estimated the World Wide Web to contain 11.5 billion pages by January 2005. In the same year, Yahoo![2] announced that its search engine index contained more than 19.2 billion documents. It was estimated by http://www.worldwidewebsize.com/ that there were about 25 billion pages indexed by major search engines as of October 2008. Recently, the Google blog[3] reported that about one trillion web pages have been seen during their crawling and indexing. According to the above information, we can see that the number of webpages is growing very fast. Actually, the same story also happens to the number of websites. According to a report[4] from Netcraft, the evolution of websites from 1995 to 2012 is shown in Fig. 2.3. Additionally, this amount of information is not only restricted to Web pages. According to the Internet World Stats[5], the number of Internet users are above 2 billion by March 2011. As most of those users are involved in various activities on the Web such as writing blogs, engaging in social networks, or uploading videos and images, the associated information of these users are also an important type of information and contributes to the size of the Web.

The extremely large size of the Web makes it generally impossible for common users to locate their desired information by browsing the Web. This is widely known as the *information*

[1] http://www.cs.uiowa.edu/ asignori/web-size/
[2] http://www.iht.com/articles/2005/08/15/business/web.php
[3] http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html
[4] http://news.netcraft.com/archives/category/web-server-survey/
[5] http://internetworldstats.com/stats.htm

overload problem on the Web. It is a highly important problem as 80% of active Web users use Google search engine every day. According to the IDC Institute[1], people spend on average 32.7 hours per week online and spend approximately 7 hours weekly just for searching the Web, without opening any websites. In such an abundant amount of information, any tool that we interact with is acting as an *information filtering system* that filters out non-relevant information while displaying the most relevant ones [87]. Take major search engines (e.g. Google, Bing) as an example. The algorithmic filtering is applied in these search engines, where a number of algorithms filter billions of Web pages to find the most relevant results in various stages of the architecture in Fig. 2.2 (e.g. crawling, ranking, and user interaction). As a consequence, the effectiveness of information filtering highly depends on the decision of "what is relevant and what is not" and different approaches have different definitions of relevance shaping highly their success [148].

Although Web search engines use hundreds of factors to determine the relevance of a document, one of the main factors for relevance is considered to be *popularity*. In the early works such as Google's PageRank approach [33, 175], HITS [121] or SALSA [169], popularity is solely governed by *link analysis* (e.g. number of links that point to a Web page) and the authority value is calculated independent from the query and content. Clearly, these approaches indicate that document quality and authority depend on social and collaborative aspects (e.g. links) of the community. Nearly all major search engines now combine such link analysis scores, similar to those used by Google, with more traditional IR scores aside. However, one problem of these algorithms is the fact that they do not explicitly take into account the actual content of the document, including its metadata or semantic interpretations [5]. A similar analogy can be drawn between popularity-based ranking approaches and the *collaborative filtering*-based recommendation systems. As widely known, these systems also do not take the actual content into account, and user recommendations are produced based on neighborhoods in a user-item graph. Despite their success, recently it is shown that hybrid systems that combine collaborative-based and content-based features can significantly improve recommendations for textual items [144]. Analogously this leads to the hypothesis that "being popular does not always mean to being relevant" to particular information needs of the users [5, 194].

To this end, several improvements are proposed for ranking in the IR literature to decide "what is relevant and what is not". Personalized Web search is one of those improvements to ranking that can provide different search results for different users based upon individual interests, preferences, and needs [68, 141, 184]. In this regard, it differs from generic Web search that returns identical search results to all users for identical queries. Since the main goal of personalization is to gain the capability to change the search functionality to the particularities of the users, capturing and representing users' (short-term) context and (long-term) profile becomes an important step for personalization. This information can be specified by the users (explicit collection) [45, 184, 210] or automatically learned from users' activities (implicit collection) [109, 142, 209, 217, 220]. Collected information is then processed and organized as a user profile in a certain structure, depending on the need of the personalization algorithm. Despite the recent increasing interest in personalization approaches, the effectiveness of personalization is still in question, especially to infer users' information needs that are

[1] http://www.idc.com/getdoc.jsp?containerId=224072

not static and randomly changing. Recent research work has shown that user search histories and profiles inevitably contain noise that is irrelevant or even harmful to actual search request [68, 221]. This makes personalization strategies unstable to *decide on relevance* and not a proper candidate to overcome the information overload problem on the Web.

Another emerging paradigm is *social search* that deals with the search within (or using) a social environment. In this paradigm, the first assumption is the existence of an environment where a community of users actively participate to improve the search process. With the emergence of social media sites such as Digg (websites), Twitter (status messages), Cite-U-Like (research papers) or social networking sites such as MySpace and Facebook, a vast amount of user-generated data becomes available to optimize the search. *User tags* are one sort of this data that describes Web resources and enables some kind of manual indexing where the content of a resource is represented by manually assigned terms [22, 173, 239]. In [93], the potential of tags for enhancing search is investigated and it is found that there exists a reasonably high overlap between search query terms and tags. The users queries are mapped to the tags in [243] which trains a probabilistic latent topic model for tags and returns tags that fall into the same latent topic(s) for a given user query. In addition, tags are also used for personalized search in which the rank of a Web document is decided by topical matching between the user and document tags [173, 245]. Despite some improvements, the dependence on tags for Web search introduces a number of challenges. First, tags are inherently noisy. Similar to anything that users create, the tags can also be off topic, inappropriate, misspelled or spam. Therefore, obtaining high-quality tags is a difficult task. In addition, tags are mostly syntactic features, and missing conceptual interpretations and structural relationships which result in uncontrolled vocabulary for better indexing and ranking. Finally, manual tagging demands a high user effort that is only practical in a limited scope (e.g within a Web site) and difficult to apply to and control in large scale Web settings.

Another type of social search is to exploit the concept of *online community* that represents a group of users sharing common goals, traits, or interests. Here the main assumption is that the social network of a search user provides richer and reliable clues about the purposes and interests of his/her information needs. So the results of a search request are biased based on such information as which network the user belongs to, who the user's peers are and what their interests are. Theoretically such an approach is not something new, but an integration of previous *link analysis* and *personalized search* approaches operating on the data coming from the social network. In [86, 166], the search history of the user and his/her peers in the network are analyzed for ranking and the documents appearing in the other users' search history are given more ranking weights. In [258], the query and document language models are expanded based on this information. [59] proposes the notion of peer-sensitive object-rank, where peers receive resources from their friends and rank them using different trust values for each peer, i.e., assigning higher score values to resources from trusted friends. In addition, [12] provides a framework to cast the different users of such networks into a unified graph model representing their mutual content and tags, and derives scoring functions out of each entity of this graph. Despite the useful information emerging from the social network, this type of social search is highly dependent on the community mostly in a query-independent fashion. That's why determining the degree of relevance based on this information can highly be misleading since

the notion of relevance is highly subjective and dependent on the query and the user's temporary context. In addition, most of the approaches are still depending on the users' tag to determine their interests (as well as search history) which inherently brings up the tag-related challenges discussed above.

2.2.2.1 Ranking Models for Document Retrieval

Because of the central role of ranking in search engines as shown in Fig. 2.2, great attention has been paid to the ranking models in IR and several ranking models are defined. Intuitively, a ranking model defines a matching score between the query and the document and can be classified into four types, *probabilistic models*, *algebraic and logical models*, *information theoretic models*, and *Bayesian models*. The early ranking models retrieve documents based on the occurrences of the query terms in the documents. Examples include the Boolean model [130] which basically can predict whether a document is relevant to the query or not, but cannot predict the degree of relevance.

In the Vector Space model (VSM) the terms and documents are considered as vectors in a Euclidean space, in which the inner product of two vectors can be used to measure their similarities [201]. For an effective vector representation, TF-IDF weighting is adopted where the TF of a term t in a vector is defined as the normalized number of its occurrences in the document, and the IDF of it is defined as $IDF(t) = log\frac{N}{n(t)}$ where N is the total number of documents in the corpus, and $n(t)$ is the number of documents containing term t. The TF-IDF model of VSM is an algebraic and logical model but in nature it is also related to probabilistic models and the term independence is an implicit assumption of the model.

Both Boolean model and VSM make implicit assumptions about the relevance which is not always desired when defining a ranking model. Therefore, probabilistic models are proposed based on the early theoretical work of *Probability Ranking Principle* of Robertson [195]. It proposes the binary independence retrieval (BIR) model that assumes that relevance is an event of the probabilistic space and can be learned directly from sample data. The success of BIR leads to some famous ranking models such as the BM25 model [197], which can be categorized as probabilistic ranking model by extending BIR to include document and query term weights. The basic idea of BM25 is to rank documents by the likeliness of their relevance, but it is not a single model, instead defines a whole family of ranking models, with slightly different components and parameters. One of the popular instantiations of the model is as follows:

$$BM25(d,q) = \sum_{i=1}^{n} IDF(q_i)\frac{tf_{q_i}(k_1+1)}{tf_{q_i} + k_1(1-b+b\frac{|d|}{avgdl})}$$

where k_1 and b are free parameters and $avgdl$ is the average document length.

The success of probabilistic models leads to the use of *statistical language models* for IR which has been widely used in other fields such as speech recognition, machine translation, and handwriting recognition. A statistical language model assigns a probability to a sequence of terms, possibly all of those in the vocabulary. In its simplest form, a language model is directly associated with a document and, given query q as input, documents are ranked based on the query likelihood, or the probability that the document's language model θ_d will generate the

terms in the query (i.e., $P(q \mid d)$) [127, 179]. By further assuming the independence between terms, one has $P(q \mid \theta_d) = \prod_{i=1}^{n} p(q_i \mid \theta_d$, if query q contains terms $q_1, ..., q_n$. For example, a document language model θ_d might assign a probability of 0.1 to the word "Audrey" and 0.05 to the word "Hepburn" (i.e., $p(Audrey|\theta_d) = 0.1$, $p(Hepburn|\theta_d) = 0.05$) and if our query q is "Audrey Hepburn", we would have $p(q|\theta_d) = 0.1 * 0.05 = 0.005$. Thus intuitively, the more frequently a query word occurs in document d, the higher the query likelihood would be for d, capturing the basic TF retrieval heuristics. Thus the ranking model is now reduced to estimating the language model θ_d in which a maximum likelihood (ML) estimation is used in a simple case. A smoothed model is also obtained by interpolating the ML estimate with a background language model obtained from the entire corpus and θ_d and is defined as follows (C represents the collection model) [255]:

$$p(t \mid \theta_d) = \lambda \frac{c(t,d)}{|d|} + (1-\lambda) p(t \mid C)$$

where $p(t \mid C)$ is the background language model for term t, $c(t,d)$ is the count of the term t in document d, and λ is a smoothing factor.

There are many variants of language models, some of them even go beyond the query likelihood retrieval model (e.g., the models based on KL divergence [127] or model-based feedback [254]). When considered as an individual ranking model, language models are comparable to other ranking models such as BM25 or VSM. However, the uniqueness of language models emerges from their underlying probabilistic interpretation which allows to establish more general formal frameworks to unify different language models and facilitate systematic extensions of the language model to introduce new ranking models. In this regard, two general formal frameworks are proposed: The first one is the *risk minimization* framework whose basic idea is to formalize the retrieval problem generally as a decision problem with Bayesian decision theory, and provide a solid theoretical foundation for thinking about problems of action and inference under uncertainty [256]. Language models are introduced into the framework as models for the observed data, particularly the documents and queries. The second formal framework is the *generative theory of relevance* framework developed by Victor Lavrenko [131, 133] which also constitutes the basis of our work in this thesis. It is mainly based on the generative relevance hypothesis that can be stated as follows:

Generative Relevance Hypothesis. For a given information need, queries expressing that need and documents relevant to that need can be viewed as independent random samples from the same underlying generative model.

In other words, to define the relevance in a generative way, Lavrenko takes a somewhat radical step and assume that queries and documents originate from a *latent representation space* $\mathbb{S}$. One can think of $\mathbb{S}$ as information or knowledge space which is rich enough to encode all desired attributes of documents and queries (including nonexistent attributes). Then a query is the result of applying a transform function $Q : \mathbb{S} \rightarrow \mathbb{Q}$ to some point x in the representation space where $Q(x)$ filters and shuffles components of x to make it look like a query. To get a document, similarly one can apply a different transform function $D : \mathbb{S} \rightarrow \mathbb{D}$. Both D and Q are deterministic and able to generate documents and queries from a representation space. That's why both of these functions are called *generators* or *generative density allocation*

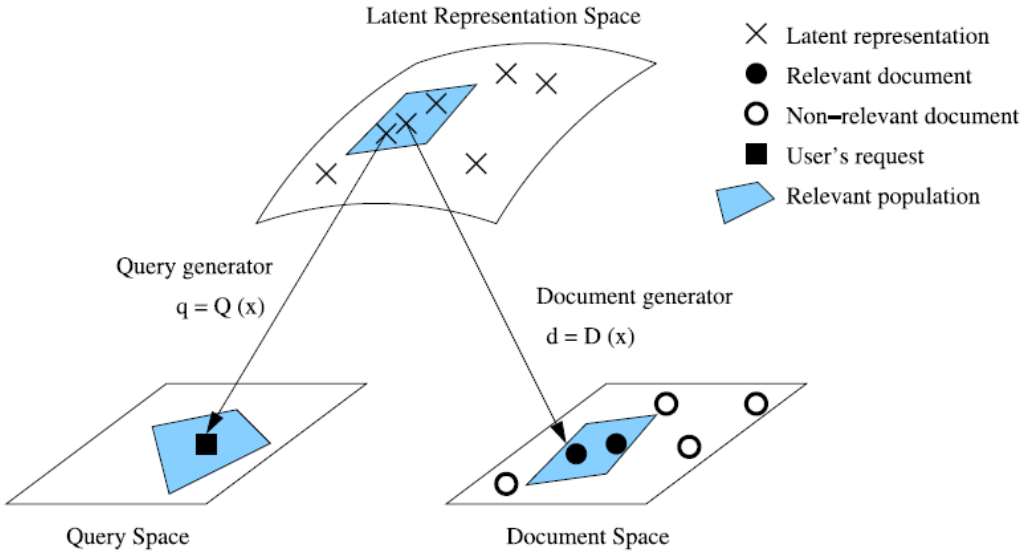


Figure 2.4: Representation of documents, queries and relevance in Lavrenko's model (Source [131]).

functions.

Depending on the allocated representation space and the generators, the framework can be instantiated for different ranking models, called *relevance models*. In basic form, a set of pseudo-feedback documents D_F are placed as representation space (i.e. $\mathcal{S} = D^{-1}(D_F)$) and unigram model is used as a generator function [133]. Over the years, the relevance models for cross-lingual retrieval [132], image retrieval [106], handwriting retrieval [135], or video retrieval [134] are proposed. In this work, we also provide two extensions of the framework: A relevance model for document retrieval by employing the structured RDF data as representation space is proposed in Ch. 5. In Ch. 6, another extension of the framework is also presented that employ pseudo-feedback entities as representation space for keyword search on structured data.

2.2.3 Web Search: A Data Retrieval Perspective

In the recent years, we observe increasing amount of structured data on the Web which emerges under different names and representations. This situation results in the Web to be rapidly deepened by the massive network of the data while the surface Web still has linked billions of static HTML pages. In fact, it is highly believed that a far more significant amount of information is available on this deep Web.

One prominent example of the Web data has emerged under name of *Linked Open Data* (LOD) [24]. In fact it can be regarded as basically semantic data published in RDF on the Web. The concept of linked data is about publishing and establishing links between data from different sources. The community specifically focused on this topic has worked out best practices

for exposing, sharing, and connecting data on the Semantic Web. The four principles of linked data published by Tim Berners-Lee are [24]:

- Use Uniform Resource Identifiers (URI) to identify things.
- Use HTTP URIs so that these things can be referred to and looked up ("dereference") by people and user agents.
- Provide useful information (i.e., a structured description) about the thing when its URI is dereferenced.
- Include links to other, related URIs in the exposed data to improve discovery of other related information on the Web.

Based on these simple principles, a large amount of legacy data has been transformed and exposed as linked data. This is supported by the W3C Linking Open Data community project. Its main goal is to augment the Web with open and interlinked RDF datasets. In October 2007, datasets consisted of over two billion RDF triples, which were interlinked by over two million RDF links. By May 2009 this had grown to 4.2 billion RDF triples, interlinked by around 142 million RDF links. Some datasets that can be found on the linked data are:

- DBpedia represents the structured data counterpart of Wikipedia. It contains data extracted from Wikipedia texts. There are about 2.18 million resources described by 218 million triples, including abstracts in 11 different languages.
- DBLP Bibliography provides bibliographic information about publications; it contains about 800,000 articles, 400,000 authors, and approx. 15 million triples.
- GeoNames provides RDF descriptions of more than 6,500,000 locations worldwide.
- UMBEL is a lightweight ontology of 20,000 classes and their relationships derived from OpenCyc. It has links to 1.5 million named entities from DBpedia and YAGO, another dataset that has been built to capture the information in Wikipedia.

There are also domain-independent LOD sources such as OpenCyc, DBpedia and Freebase, and sources capturing knowledge of some particular domains such as music, health care and life science.

Linked data has attracted much interests from researchers of the Semantic Web community. Increasingly, this large amount of data has become a topic also for database researchers. Representing one growing Web of data, it opens new practical problems that the long-studied theories of database research can applied in the field of Web databases. It is not only a testbed for existing database concepts and techniques but also introduces new challenges that are relevant for database research.

Also, linked data has attracted interests from industry. Large companies including Google, Yahoo, Microsoft have embraced this new development and are now studying various ways to commercially exploit it. Besides for Web search and other domains of commercial applications,

linked data has become a subject for politics. The governments of the US[1] and Great Britain[2] are now releasing a large amount of linked data and elaborating on applications to expose them to citizens.

Actually the Web data is not only limited to LOD. The Web databases that are hidden and cannot be accessed directly through static URL links are also an important part of the Web [43]. Mostly, they can be accessed via specialized query interfaces which direct the query to the database and generate a list of dynamic results. Because current crawlers cannot effectively access those databases, such data become invisible to traditional search engines, and thus remains largely hidden from users. It is predicted that there are about 450,000 Web databases among which 348,000 were structured by 2004 [43].

There are two direct implications of the Web databases related to our work. Some of this data has already been made public either on the LOD (e.g. DBLP, IMDB etc.) or via virtual integration techniques that based on constructing mediator systems, potentially one for each domain (e.g. used cars, real-estate, or books). Therefore it can be regarded as a domain-specific source of data that will further increase the currently existing amount of open data on the Web. In addition, searching these databases is another challenge for Web search. Mostly, access to the data is wrapped with domain-specific forms that allow the users to pose their queries via the interfaces highly tailored to the underlying data model [153]. The search relevance is suboptimal in this case which is highly specialized with some heuristics and a generic method to search over these databases is still a challenge.

2.2.3.1 Ranking Models for Data Retrieval

As the data on the Web increases exponentially, ranking models to retrieve and list the information from this type of data have also emerged. In particular, we can categorize most of those approaches that employ an IR-like search on the structured data as *keyword-search on the structured data* [251]. As the nature of structured data highly differs from the unstructured one, the traditional methods (i.e. TF-IDF, PageRank, LM etc.) of IR cannot be directly applied to this well-known problem. In particular, the traditional IR methods consider that a document as major information unit is usually unstructured or slightly structured so that query processing on documents mostly concerns a ranking scheme of how the words of such a document are generated from an underlying probabilistic model. However, such a generation is more complex for structured data. The information unit which is retrieved is mostly a join of tuples that is generated traditionally via a structured query processing such as SQL, XQuery or XPath, and the ranking scheme also needs to take this issue into account. In this sense, ranking models for keyword search on structured data are more complex and computationally expensive.

Formally defined, a user query Q is considered as a set of keywords $(q_1, ..., q_m)$ as in the case of document retrieval. However, here any keyword matches a resource (or tuple) r if it matches any of its attributes $\mathcal{A}(r)$, which includes most of the textual content. An answer to the user query Q is a minimal rooted directed tree that can be found in the data, which contains at least one matching resource for every keyword in Q. This is commonly referred to as a Steiner

[1] http://www.data.gov/semantic/index
[2] http://data.gov.uk/

tree [2, 114] and data is mostly represented (or transformed) as a graph. In this work, we use the term *Joined Resource Tree* (JRT) to make clear that an answer is a joined set of resources represented as $JRT = \{r_1, \ldots, r_n\}$. In recent work, subgraphs have also been considered as keyword answers [225], instead of trees. While this difference in semantics requires more advanced mechanisms for result computation, we consider that this aspect is orthogonal to the problem of *keyword search result ranking* studied in Chapter 6: given a user query, the goal here is to rank Steiner trees (or graphs) according to the user's perceived degrees of *relevance*. In other words, we assume to produce a ranking of keyword search results that corresponds to the degree to which they *match the user information needs*.

DBXplorer [4] and DISCOVER [96] are one of the early systems where keyword search on structured data is applied on a database setting. These approaches are mainly based on the assumption that the columns in a database contain the query keywords and can easily be retrieved from a proprietary index structure to indicate possible starting points for search algorithm. DBExplorer and DISCOVER utilize such an index to retrieve the database tables (i.e. keyword relations) to find the tables where the tuples of these columns reside in. Then, given a schema graph of the underlying database, an intermediate representation, called *candidate network*, is created which is actually a connected tree of keyword relations where for every two adjacent keyword relations, there exists an edge in the schema graph of the database. In fact, a candidate network is a template that can produce a set of (possibly empty) JRTs, and it corresponds to a relational algebra that joins a sequence of relations to obtain JRTs over the relations involved. However, the result of this process can generate a number of candidate networks (CNs) because the initial columns containing query keywords vary. The generation of such candidate networks is the main focus of these approaches while ranking is considered only to sort the final JRTs in the result list.

The problem of keyword search on structured data can also be viewed as a more general problem of searching data graphs which became available after the aforementioned approaches of searching relational databases. In this regard the keyword search on data graphs can be used both in the context of relational database (as database can trivially be converted to a data graph as we present in Ch. 6) as well as in the context of more semi-structured data such as the data in XML or RDF. Then, instead of creating candidate networks that generate query templates, graph based approaches directly search the data graph to retrieve the answers to the keyword query. In particular, the problem is to find a set of subgraphs of a data graph each of which is *connected* and contains at least one node for each keyword of the query. In other words, these subgraphs correspond to JRTs in our formalism of the problem and each entity node of this graph corresponds to the resources in those JRTs. Different requirements for the property of subgraphs that should be returned have also been proposed. There are mainly two different structural requirements: (1) a reduced tree that contains all the keywords that is referred as tree-based semantics; (2) a subgraph, such as r-radius steiner graph [138], or multi-center induced graphs [185, 225] that is called subgraph-based semantics. The aim of the search algorithm in this case is to find all the subgraphs via graph exploration and mostly implemented via the forward, backward or bidirectional search algorithms [13, 90, 114]. Ranking is also a major issue in this sort of approaches as there are many subgraphs that can fulfill a request and a suitable strategy for ranking is needed that finds the best answers and orders them accordingly.

Ranking issue in this regard is discussed in many other work including [96, 143, 211]. In fact, ranking can be considered orthogonal to the both aforementioned types of retrieval techniques because those approaches to ranking aim at designing ranking functions on JRTs that capture both the textual information (e.g., IR-Styled ranking) and structural information (e.g., the size of the JRTs). Generally speaking, there are two categories of ranking functions, namely, the attribute level ranking function and the tree level ranking function. The main idea behind ranking is to (1) assign each resource r in the JRT a score $Score(r)$, and then to combine the individual scores using a monotonic aggregation function $agg(\cdot)$ to obtain the final score $Score(JRT)$. Most commonly used is the IR-style ranking function adopted from TF-IDF based ranking. For instance, Discover [143] uses the sum of individual TF-IDF scores obtained via pivoted normalization weighting [211]:

$$\begin{aligned} Score(JRT) &= \sum_{r \in JRT} Score(r), \\ Score(r) &= \sum_{v \in r,Q} weight(v,r) * weight(v,Q), \\ weight(v,r) &= \frac{ntf}{ndl} * idf, \\ ntf &= 1 + ln(1 + ln(tf)), \\ ndl &= (1-s) + s * \frac{dl}{avgdl}, \\ idf &= ln\frac{N}{df+1}, \end{aligned} \tag{2.1}$$

where ntf is the normalized term frequency (the normalized number of occurrences of v in r), df is the document frequency (the number of r containing the term v), N is the total number of resources in the collection, idf is the inverse document frequency, dl is the length measured as the number of terms contained in r, $avgdl$ is the average length in the collection, s is a constant usually set to 0.2, and ndl is the normalized document length.

In fact, Liu et al. [143] adopts this weighting of DISCOVER and extends it via four different normalization methods. The goal is to obtain customized (1) idf and (2) ndl values, and to introduce (3) inter-document weight normalization as well as (4) Steiner tree size normalization. They are motivated by the facts that each column text has different vocabularies and thus shall be treated as a single collection, whereas a Steiner tree is actually an "aggregated resource" and thus requires additional normalization beyond the resource level. Similar to document length, the size of the tree shall have an effect on relevance.

The last factor is very specific to this keyword search setting. Closely related to this Steiner tree size metric is the length of "root to matching nodes" paths [90], or "leaf to center nodes" paths [225]. All these metrics aim to capture the goal of what is known as proximity search, i.e., to minimize the distance between search terms (matching resource nodes) in the data graph. Applying this heuristic yields higher scores to those Steiner trees that are more compact. Intuitively speaking, the assumption here is that trees with more closely related matching nodes more likely capture the intended information need.

Besides IR-style scores defined for matching nodes (IR), and scores representing the distances between nodes in the result (proximity), the third category of scores commonly used is node prestige derived via PageRank [114]. Further, while most scoring functions are designed to be monotonic, examples can be constructed where the resulting ranking contradicts human perception [150]. This gives rise to non-monotonic aggregation functions [150] that however, preclude the large body of existing query processing algorithms.

2.3 Machine Learning for Information and Data Retrieval

In recent years, we have witnessed successful application of machine learning techniques to a wide range of information retrieval problems, including Web search engines, recommendation systems, online advertising, etc. It also crucial for our work presented in the following chapters to understand some of the core machine learning techniques.

2.3.1 Supervised Learning

Supervised learning refers to any machine learning process that learns a function from an input type to an output type using data comprising examples that have both input and output values. Two typical examples of supervised learning are *(1) classification* and *(2) regression*. In these cases, the output types are respectively categorical (the classes) and numeric. Supervised learning stands in contrast to unsupervised learning, which seeks to learn structure in data, and to reinforcement learning in which sequential decision making policies are learned from reward with no examples of "correct" behavior.

In regression, we are typically interested in inferring a real-valued function (called a regression function), whose values correspond to the mean of an output variable conditioned on one or more independent variables. This is represented as functional dependency $y_i = f(x_i) + \epsilon_i$ from N observed variables $\{x_i, y_i\}_{i=1}^{N}$. Many different techniques for estimating this regression function have been developed including parametric, semi-parametric, and nonparametric methods. Linear regression, for example, assumes the regression function as $f(x_i) =< \phi(xi), w >$ by a parameter w, whereas Gaussian Processes for regression try to derive the functional relationship directly from the data, that is, they do not parameterize the regression function.

Regarding the classification, there are different sorts of techniques that are applied including instance-based learning, rule-based learning, decision trees, logistic regression or support-vector machines (SVM). No matter which classification algorithm is used, the end result is the division of the input space into regions belonging to a single class. Mostly, the input space is denoted by the Cartesian product of the attributes, all possible combinations of possible values and a linear discriminant function divide the space into half-spaces each with examples falling under one typical class.

2.3.2 Topic Models

Topic models emerge as a set of tools to discover the hidden thematic semantics of word correlations in an underlying text corpus [26, 28, 215]. In fact, it is based on the assumption that

any basic unit (e.g. document) of text corpus is a mixture of topics, where a topic is formally defined as a probability distribution over a fixed vocabulary of words. For example, Fig. 2.5 illustrates such a generation of a document from a number of topics. Each document in the corpus is assumed to be generated as follows: One chooses a distribution over topics, called topic proportions of the document. Then, for each word in that document, one chooses a topic at random according to this distribution, and draws a word from that topic. In this setting, a topic model is a generative model which specifies a simple probabilistic procedure by which text data can be generated. In fact documents themselves are observed data in the topic models. The topic structure including the topics, per-document topic proportions, and per-document per-word topic assignments are hidden representations that need to be infered by "reversing" the generative process. In order to achieve this, a Bayesian model is proposed in *Latent Dirichlet Allocation* (LDA) approach [28]. Mainly, variational methods [28] or Markov Chain Monte Carlo (MCMC) techniques are used for learning topic models.

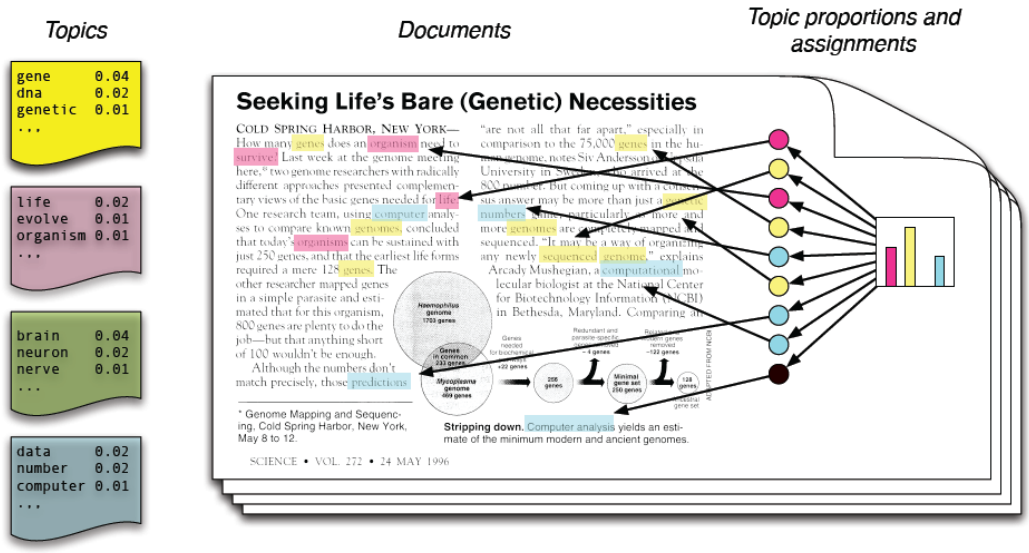
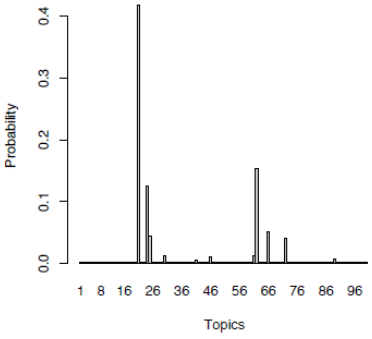


Figure 2.5: Illustration of the generative process of topic models: The document has a distribution over a number of topics (on the left) and each word is drawn from a selected topic (source:[25])

Furthermore, Fig. 2.6 depicts a result of an example inference on the document presented above. Among the 100 topics, there are a number of topics which are given a high probability for that document. For this document, the topics such as "genetics", "evolution", "disease", and "computers" are given a high probability. Fig. 2.6 also shows high probability words for those topics that are learned out of an inference process. Note the semantic closeness of the most probable terms in each topic that stems from the property that the inferred hidden structure of topics resembles a thematic structure of the corpus. Such a topic structure performs the annotation of each document in the corpus and the topics are useful semantic information to be further exploited in a number of tasks such as IR, classification, and document browsing.

In the scope of IR, topic models are useful to address the *vocabulary mismatch problem* that



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Figure 2.6: An illustration of topic proportions and related topics for the example document from 100-topic LDA model (source [25]).

results from the discrepancy between the documents and query keywords. A topic in this sense is a group of different words that occur in a similar context. Thus, a query and a document represented in such a low-dimensional space can still have high similarity even if they do not share a word. In [238], Wei and Croft successfully showed that using LDA [28] to model a document as a mixture of topics performs better than the one topic (cluster) assumption. Basically, in this retrieval model, a query $\mathbf{q} = q_1, ..q_{|q|}$, is generated from a document $\mathbf{d}$ using the following process: A multinomial distribution $\boldsymbol{\theta}^d$ over topics is selected for $\boldsymbol{d}$. Then, a latent topic z is selected with probability $P(z \mid \boldsymbol{\theta}^d) = \theta^d_z$. Finally, each $q \in \mathbf{q}$ is generated with probability $P(q \mid \boldsymbol{\beta}, z)$, i.e. the probability of word q being observed through repeated sampling from words in topic z. Here every $\beta_z \in \boldsymbol{\beta}$ refers to a multinomial distribution of the topic z over the words and thus assigns probabilities to every word in the vocabulary. Using this generative model, the probability of generating the query $\mathbf{q}$ from the document $\mathbf{d}$ is defined as:

$$P(\mathbf{q} \mid \mathbf{d}) = \prod_{q \in \mathbf{q}} \sum_{z} P(q \mid \boldsymbol{\beta}, z) P(z \mid \boldsymbol{\theta}^d)$$

LDA-based topic model actually differs from its predecessor model Probabilistic Latent Semantic Analysis (PLSA) in that it places Dirichlet priors over the parameters of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. Experiments showed that this helps to overcome the problem of overfitting.

Instead of learning topics and model parameters in a completely unsupervised manner, recent work uses predefined topics. Ramage et al. assume a one-to-one correspondence between LDA topics and user tags and focus on learning word-tag distributions [190]. Also ontology concepts have been used as topics [44]. To establish the connection between these predefined topics and the collection, documents were assumed to be annotated with a set of topics (tags) [190], or topics (concepts) to be associated with a set of words [44].

2.3.3 Relational Learning

Relational learning considers a learning context in which there exist relationships between training examples, or the examples have a complex internal structure – i.e. consist of multiple elements and relationships between those elements. In other words, the "relational" may refer to both an internal (e.g. Web documents linking each other) or external relational structure (e.g. multi-relational database) describing the examples. Learning in such a setting is relatively hard task when compared with the classical ML algorithms. Traditionally, most ML algorithms assume a attribute-value learning, where one learns from a set of examples each represented as a single vector of features. In this setting, each example is assumed to be independent and identically distributed (i.i.d). However, when relationships are present among the examples, this assumption is violated and relationships among examples need to be exploited by the learning algorithm.

Many different kinds of learning tasks have been defined in relational learning, and an even larger number of approaches have been proposed for tackling these tasks. Fig. 2.7 depicts a categorization of the main approaches. In the following, we give an overview of two prominent lines of work in relational learning, namely *Inductive Logic Programming* and *Statistical Relational Learning*.

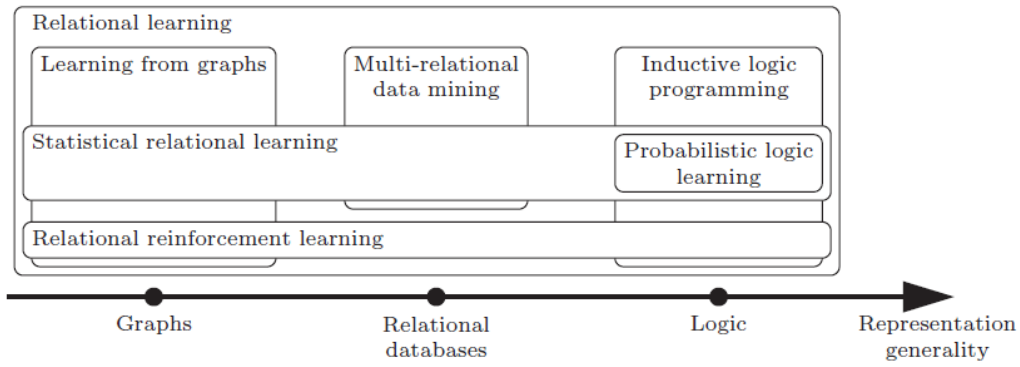


Figure 2.7: An overview of relational learning tasks and data representations.

2.3.3.1 Inductive Logic Programming

Inductive Logic Programming (ILP) is an important method used in ML and knowledge discovery fields where the main focus is on the development of techniques for relational data mining. It is a central component of the framework presented in this work. Originally ILP systems worked with sets of examples and background knowledge. Later they became also applicable for learning from large databases.

The major difference of ILP from the typical data mining methods is that other than learning from a single table, the ILP systems can work with a database containing multiple related tables. The other distinguishing feature of ILP systems is the ability to take into account background knowledge as a logic program. The contribution of working with background knowl-

edge is the retrieval of additional information and respectively more flexibility and accuracy of learning results.

The traditional approaches that are working with data presented as a single table are called *propositional or attribute-value approaches*. The patterns they can learn can be expressed in propositional logic. There are a number of problems resulting from the assumption of the data being represented in a single table. The most obvious is that the possible loss of information which is not contained in the processed table can lead to inaccurate and insufficient results. The other problems concern the overhead resulting from the integrating relations through natural joins when applying propositional representation to multiple relations. The patterns or rules that the ILP are able to learn are formulated in first-order logic which can formally express complex relations in data. Therefore ILP is referred to as relational learning or first-order learning method.

Logic programming lies at the basis of the ILP. Logic clauses are the basic elements that comprise logic programs. A conjunction of clauses $c_1, \ldots, c_n$ which are true build a database or a clausal theory used in logic programs. It is also referred to as a knowledge base. Clauses consist of two main parts: the head and the body, whereas the body of a clause is the conditional part and the head corresponds the conclusion if the clause is seen as a first-order rule. The head and the body of a clause contain atoms which are some predicates applied on variables or constants. A clause can be expressed in the following form:

$$h_1 \vee \ldots \vee h_n \leftarrow b_1 \wedge \ldots \wedge b_m$$

where the h_i are the head and b_j are the body atoms. The symbol ' $\leftarrow$ ' stands for implication and all variables contained in a clause are implicitly universally quantified. The notation corresponds to the disjunction of literals (atoms or their negation):

$$h_1 \vee \ldots \vee h_n \vee \neg\, b_1 \vee \ldots \vee \neg\, b_m$$

or a set of literals: $\{\, h_1 , \ldots , h_n , \neg\, b_1 , \ldots , \neg\, b_m \,\}$

For example:

$$\textit{grandmother(X, Z)} \leftarrow \textit{mother(X, Y), son(Z, Y)}$$

The clause says that if X is a mother of Y and Z is a son of Y, then X is a grandmother of Z. *grandmother(X, Z)* is the head and *mother(X, Y)*, *son(Z, Y)* comprise the body of the clause. *grandmother(X, Z), mother(X, Y), son(Z, Y)* are atoms, *X, Y, Z* are variables and *grandmother, mother, son* are predicates.

There are different types of clauses. The clauses that are used in ILP most often are the definite clauses and facts. Definite clauses and facts contain exactly one atom in the head. The body of facts is empty, which implies that the clause is always true.

In addition, a substitution $\theta = V_1/t_1, \ldots, V_n/t_n$ is an assignment of terms $t_1, \ldots, t_n$ (t_i being a constant, variable or a function) to variables $V_1, \ldots, V_n$. The instantiated formula $F\theta$, where F is a term, atom, or clause and $\theta = V_1/t_1, \ldots, V_n/t_n$ a substitution, is the formula obtained by

replacing all variables V_1, ..., V_n in F by the terms t_1, ..., t_n. The formula is ground if there is no variable occurring in it.

Whereas the major focus of logic programming is deduction, ILP makes use of inductive inference. The principle of induction lies in the generalizing from observations in the presence of background knowledge. The logical settings in ILP depend on the concrete learning task. The two most well-known learning problems are concept and predicate learning.The goal of the concept learning is to find a classifier (hypothesis) that would distinguish between the instances of the target concept and the instances that don't belong to it. Formally the settings can be defined in the following way:

Given:

- The language of examples $\mathcal{L}_E$
- The language of Hypotheses $\mathcal{L}_H$
- A coverage relation *covers(H,e)*
- A set of examples E described with $\mathcal{L}_E$

To find:

- A hypothesis (or a set of Hypotheses) H described with $\mathcal{L}_H$ that covers positive examples and doesn't cover any negative examples

The coverage relation *covers(H,e)* tells if the example e is covered by the considered Hypothesis H. A set of examples includes positive and negative examples. The positive ones are those that unlike the negatives belong to the target concept. The hypothesis that covers all positive examples is called complete. Consistent hypothesis covers no negative examples. There are variations of the concept learning settings concerning the coverage relation depending on the type of learning:

- *Learning from entailment*
  This type of learning algorithm proposed by Muggleton [168] considers H as a theory and e as an example. The coverage relation *covers(H,e)* returns true iff H $\models$ e. The hypothesis logically entails the example.
- *Learning from interpretations*
  In this method which was introduced by De Raedt and Džerovski [60] e is a Herbrand interpretation and *covers(H, e)* is true iff e is model of H.
- *Learning from satisfiability*
  In this setting proposed by Wrobel and Džerovski [241] both H and e are theories and *covers(H, e)* is true iff $\{H \wedge e \not\models \bot\}$. This implies that the hypothesis together with the example should be satisfiable.

Among these three settings learning from entailment is the most widely used one and is considered to be the easiest learning method. In the presence of background knowledge B the algorithm is to be extended by considering $H \wedge B$ instead of only H in all settings of the coverage relation.

2.3.3.2 Statistical Relational Learning

The early research on relational learning has largely focused on handling relational structures of the data while ignoring the uncertainty and probabilistic inference. In this regard, Statistical Relational Learning(SRL) is an emerging branch in relational learning that deals with machine learning and data mining in relational domains where observations may be limited, partially available, or contain noise [80]. By successfully combining the statistical learning which addresses uncertainty and relational structure, a SRL model for a given relational data shows the correlations between attributes of entities and relationships among entities. SRL is usually represented with a graphical model and differentiates in terms of representation, learning method, and probabilistic inference.

Traditionally relational representations, and probabilistic/statistical reasoning have been studied independently of one another, while SRL investigates them jointly. This is required by the explosive growth in the amount of heterogeneous data that is being collected so recently, especially on the Web in the forms of Web databases, social networks, citation repositories and so on. Main characteristic of all these domains is the existence of uncertain information about varying numbers of entities and relationships among the entities. This is one of the main drawback of the traditional machine learning approaches which are able to cope either with uncertainty or with relational representations but typically not with both.

Many formalisms and representations have been developed in SRL. Among the best known approaches, the learning of probabilistic relational models (PRMs) [75, 79] are the most prominent examples. PRMs extend Bayesian networks to relational representation used in databases by utilizing relational structures as a template and modeling a joint probability distribution over the attributes in a database. Similar to Bayesian networks, PRMs are also graphical models in which each attribute corresponds to a random variable represented as a node and direct dependencies between the attributes are modeled by directed edges. The structure of PRMs is left highly flexible in the sense that edges can connect any attributes from a particular class or different classes even if they are not directly connected in the relational schema. Thus, those indirectly related attributes constitute a slot chain. Such a template is then used to construct a grounded Bayesian network by instantiating the PRM with particular data in a database and inference is performed on this grounded network. To handle one-to-many relationships, aggregations can be used.

However, one critical requirement of Bayesian networks is acyclicity that assumes no cycle in the constructed grounded network for inference. On the other hand Markov logic networks (MLNs) [193] upgrade Markov networks to first order logic (FOL) and allow networks with cycles. In MLNs FOL formulas are associated with probabilities that can be viewed as “soft” constraints on logical representations. Similar to PRMs, FOL formulas are used as a template to construct a grounded Markov network for particular data. Based on this Markov network, inference is applied to obtain a probabilistic model. E.g. the Alchemy system implements

structure and parameter learning for MLNs [181].

2.3.4 Learning-to-Rank

Learning-to-rank (LTR) – as the name implies– is a common name given to the approaches in which ML is applied to ranking problem and a ranking function is trained from a given set of training data. In the conventional ranking problem in IR, ranking is perceived as a generative process where a ranking function assigns a ranking value to a document directly based on the assumptions of how that document is generated. In this regard, LTR offers a supervised, discriminative approach in which the ranking function is automatically trained from available training data which can be obtained from a number of sources such as click-through data from the query logs of a search engine. Training data is normally a list of items in partial order, which can indicate the relationship between the existing documents and queries. A number of supervised learning techniques of classification and regression has been adopted as a learning algorithm and thus the structure of the ranking function highly depends on the underlying algorithm.

Formally, we can define the LTR as finding a ranking function *h*: $\mathcal{Q} \times \mathcal{D} \rightarrow \mathcal{R}$, which suggests the relation $r \in \mathcal{R}$ between a query in $q \in \mathcal{Q}$ and one of the related documents in $\mathcal{D}$ [253]. The relation $\mathcal{R}$ can be boolean {not relevant, relevant} or a set of ordinal values {extremely not relevant, not relevant, neutral, relevant, extremely relevant}. Here *h*(q, d_i) denotes the relational scoring of the document d_i in the set $d_1,\ldots, d_{n_q}$, when the query q has been given. However, documents and queries can not be directly used to create ranking results, since they are all assumed to be in a simple textual form. Therefore, any query-document pair is first transformed to a set of features that is represented as a feature vector <$f_1,\ldots, f_m$> where every feature f_i can be the score of a corresponding feature of the document and query under a predefined criterion. That's why every pair of $\langle q, d_i \rangle$ is regarded as a point in a m-dimensional feature space and considered as a input space $\mathcal{X} = \{x \mid x = \phi(q, d) \subseteq \Re^m, \forall < q, d >\}$ where ϕ is any transformation function from query-document pairs to feature vectors. In this way, the ranking function can be represented as *h*: $\mathcal{X} \rightarrow \mathcal{Y}$ between an input space $\mathcal{X}$ and the output space $\mathcal{Y}$, where $\mathcal{Y}$ denotes the space of ranking results. Given a training data of $\{\mathcal{X}, \mathcal{Y}\}$, our aim is to minimize the empirical loss calculated as $\sum_{x \in \mathcal{X}} \Delta(h(x), y)$ by optimizing h where $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re$ represents the loss function.

Actually, this problem can be understood as standard regression, classification or the ordinal regression problem. For a regression problem, one aims to learn models from training data, whose output scores can match given relevance labels of the inputs. Take the linear regression problem as an example, whose target function can be expressed as $\omega^T x + b$. The learning purpose is to find the corresponding vector ω and the bias b, which can make the hypothetical output closest to the real values. The parameter can be trained through minimizing the loss function $\sum (y_i - (\omega^T x_i + b))^2$ for example, which is differentiable and can use the gradient descent techniques to get a local optimum. Besides, as a classification problem, the ranking function can be considered as the categorization of the inputs into two classes (i.e. relevant or not relevant). Those approaches learn a threshold b and a vector ω and decide the classification via $sign(\omega^T x + b)$ in its simplest form. SVM is one of early algorithms that is utilized for this purpose that can separate the classes with a soft margin [107]. As an ordinal regression

problem, the output of the ranking function is considered as an ordinal set of reference labels, which can be expressed as{0, 1, ..., T}. The goal of ordinal regression is to learn a model, which can assign ordinal labels to the documents corresponding to a new query, which can reflect their relevance degrees. As such that the documents labeled 1 should be ranked higher than the ones labeled 0 and those labeled T should be ranked higher than those labeled T-1.

There exists a variety of LTR approaches in parallel to the learning algorithms available in ML. Generally speaking, those approaches can be categorized into three groups, namely pointwise, pairwise and listwise, depending on their input and output representation and the associated loss function. Here we briefly summarize these three categories:

Pointwise Approaches. Pointwise approaches to LTR consider the input training data as query-document pairs and their corresponding relevance score as an independent training instance. In this case the LTR problem can directly be translated into a regression or classification problem and builds a ranking model as a linear regression or classification function. Widely known approaches in this category include *Subset Ranking using Regression* [51], *McRank* [140] or *Pranking* [52]. Although the pointwise approaches are simple to formulate and can easily be handled with low computational costs, one problem of these approaches is that the training instances are always regarded as query-independent, and preferences among the documents are not considered in ranking. In addition, the position of the documents in the ranked list is not considered in the loss function, so the highly-ranked documents have more influence on the results. Thus without considering a document's position it can result in unconsciously emphasizing the influence of some unimportant documents. Finally, the number of associated document pairs can differ largely from query to query, which will result in training a model biased toward queries with more document pairs.

Pairwise approaches. These approaches can solve some of the problems of the pointwise approaches, since they take the preference between each pair of documents in the list (i.e. whether a document is prefered over the other) into account. Among those, pairwise approaches such as *Ranking SVM* [107], *RankBoost* [74], *RankNet* [35] are very popular and obtain effective results. The input space of a pairwise approach is represented as $< q, d_i, d_j >$, and hypothesis function is now interpreted as *h*: $\mathcal{Q} \times \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{R}$. Here, each relation $r \in 0, 1$ is equal to one when the document d_i is more preferred than d_j for query q and zero otherwise. Such a formulation of the LTR problem has certain advantages over the pointwise approaches: First, the prediction over the relative order of document pairs is closer to the nature of ranking and the methodologies on classification can be directly applied [39]. Second, the training instances can be easily obtained from the user's clickthrough data. However, one problem of the pairwise approaches is about the loss function (e.g. Ranking SVM) that in nature minimizes the misclassification of the documents pairs, rather than to minimize the errors in the final ranked list of documents. The loss function penalties the incorrect ranking of the top-ranked document pairs and the incorrect ranking of the low-ranked document pairs in the same degree. These problems are relieved by some adaptive methods, but mostly addressed by the listwise approach that can partially solve them.

Listwise approaches. In this category, the input is represented as a list of documents $d^{(i)} = (d_1^{(i)}, ..., d_{n^i}^{(i)})$, where $d_j^{(i)}$ denotes the j-th document of the query $q^{(i)}$. Each list of documents $d^{(i)}$ associates to a list of scores $y^{(i)} = (y_1^{(i)}, y_2^{(i)}, ..., y_{n^i}^{(i)})$, where $y_j^{(i)}$ represents the

relevance degree of document $d_j^{(i)}$ to the query $q^{(i)}$. These judgment scores can be obtained from human experts or user sessions. For example the scores can be the clicksthrough numbers of $d_j^{(i)}$, which can be retrieved by a search engine[107]. The output of listwise approaches is usually a permutation of the correct order of the documents to a query. The differences of listwise approach from the other two types are mainly the representation and optimization of the loss function, especially the direct optimization of the evaluation metrics such as MAP and NDCG. The loss function of listwise approach is defined on each query, which can naturally solve the biased problem caused by the different number of generated documents to different queries. It is natural to think of the idea for optimizing a training model by means of direct optimizing the measurement, which is used to evaluate the ranking results. But it is not trivial to settle this problem, because the measurements are not smooth, since they depend on the ranking positions. As most of the optimization techniques were developed to handle continuous and differential problems, some adaptive approximations should be applied. For example, the *SoftRank* approach [219] has approximated the non-smooth evaluation metrics to make its approximation smooth and differential, so that the optimization technique can be used. *SVM-MAP* has optimized a smooth and differentiable upper bound of the MAP metric [252]. *AdaRank* [244] and *RankGP* [250] have optimized the non-smooth objective measure directly, using boosting and genetic programming respectively. The use of direct evaluation metrics as loss function enables the listwise approaches to perform better over the pointwise and pairwise approaches. However, the problems mainly exist due to their higher complexity and computational costs.

2.4 Conclusion

In this chapter we presented an overview of previous techniques in IR and ML with a particular focus on the related work for the following chapters. We recall the distinction between the IR and DR (as in the context of databases) and also discuss how it gets blurred in the recent years with the approaches from both communities. Then we review main lines of work on how the search is conducted and how important to retrieve both types of data, specifically in the Web context. We also provide an overview of ML tools that are used in the IR context constituting a basis for the upcoming work presented in the following chpaters.

3

Search Relevance

The purpose of this chapter is to explore the concept of relevance, both in general as studied in IR, and in the scope of semantic search where relevance is determined according to the semantic content. Common to both of them are that they are both highly correlated with the more theoretical and unified view of relevance as defined in the early research of IR. Understanding "relevance" is a complex problem, and every approach in the literature presents its own view about it. After all, the purpose of a search engine is to retrieve relevant items in response to a user query. It is natural that most users have an almost good idea of what relevance is – it is simply a representation of their information need. However, in practice we need to transform such notion of relevance into a formal representation and build models to capture the best outcome matching the actual relevance degree. This turns out to be the main challenge. Over the years a list of theoretical works has been conducted in order to define what the relevance is and how we can interpret its properties. Based on the theoretical work and definitions, some major types of relevance have emerged and retrieval and ranking models have been developed to meet this need. Recently the studies about relevance in IR become to tend to be more practical, where an algorithmic relevance score is assigned to a search result representing an estimated likelihood of relevance of the search result to a topic of request. Thus the order in which the search results are presented to the user is determined based on this relevance score.

In the following we approach to the notion of relevance in three different parts: First, we review the general theoretical work that attempts to define a unified view of relevance. There are a number of different thoughts about the definition of relevance although a consensus exists for some major criteria or properties. Secondly we see a reflection of these theoretical ideas in the field of IR emerging as various types of relevance which has been introduced in the last two decades. That's why in the final part we discuss how the relevance should be perceived within the scope of semantic search and introduce so-called *semantic relevance* as the grounding work of this thesis.

3.1 A Unified View of Relevance

As the importance of relevance in IR and more generally in information sciences become apparent, several approaches have emerged to provide a theoretical and sound definition of relevance

[50, 164, 165, 202, 203]. One of the early attemps was done by Cooper [48] that perceives the relevance as a binary relation between a query and a document. Cooper defines such relevance as a logical proposition entailment with the inherent difficulty of transforming natural language to logical axioms. In another work [49], he also underlined several additional properties of documents such as novelty, informativeness, credibility, clarity, that could be considered to assess their relevance to user's needs. Later a seminal work defining the concept of relevance has been proposed by Saracevic in [202]. According to that, it is important to consider relevance from a user's point of view, thus underlining the fact that relevance has to be considered as a dynamic and multidimensional concept. This is also supported by Wilson [240] who introduced the concept of *situational relevance* emphasizing that the situation in which the search is performed, user's goal and background knowledge is crucial to identify the relevance.

During the 1990s, the relevance discussion has been intensified again, especially with the move from classical IR into Web search. After a critical review of previous approaches in [69], Eisenberg and Schamber underlined the various views of relevance stating that, a) relevance is a multidimensional concept based on user judgment and perception, b) it is a dependent on both internal (cognitive) and external (situational) factors, and c) it is a complex but systematic and measurable concept if approached conceptually and operationally from a user's perspective. Several studies have been conducted for subjective evaluation of these aspects of relevance. Barry [10] conducted a user study and identified 23 categories of relevance with numerous criteria concerning many subjective aspects of the documents. She underlined the need of considering other factors beyond topical relevance emphasizing its multidimensional nature. In a more recent study, Xu and Chen [247] conducted a relevance criteria study using a subset of criteria identified by Barry [10] in attempt to find more focused and core criteria used by users in making relevance judgments. They identified five core criteria, namely topicality, novelty, understandability, reliability, and scope.

Despite different views and perceptions of relevance, the mostly regarded work in the field of IR has been proposed by Mizzaro [165]. He followed the multidimensional line of Saracevic [202], and Eisenberg and Schamber [69] and proposed a relevance model in which almost any reasonable definition of relevance can be represented as a vector consisting of four variables: Information, Request, Time, Components. His perception of relevance shed light to many concrete implementation of relevance in the last decade and has also been extended by other approaches including [31, 50, 94]. Basically, Mizzaro explains these four core dimensions as follows as also summarized in [131]:

- *Information.* The first dimension of relevance is the kind of information resource for which we are defining relevance. In Mizarro's definition, this dimension can take one of three values: document, surrogate and information. Here document refers to the physical item a user will receive as the result of searching the full text of a document, or, in the case of multi-media retrieval, a complete image, a full audio or video file. Surrogate refers to a condensed representation of an information item, such as a list of keywords, an abstract, a title, or a caption. Information refers to changes in the user's state of knowledge as a result of reading or otherwise consuming the contents of a document. Note that information is a rather abstract concept, it depends on user's state of knowledge, his attentiveness, his capacity to comprehend the contents of the document and an array of

other factors.

- *Request.* The second dimension of relevance specifies a level at which we are dealing with the user's problem. Mizarro defines four possible levels: *RIN*, *PIN*, *request* and *query*. The first (*RIN*) stands for Real Information Need and defines the information that will truly help the user solve the problem that prompted him to carry out a search in the first place. Needless to say, the user may not even be fully aware of what constitutes his real information need, instead he perceives it, and forms a mental image of it. That image is called *PIN*, or Perceived Information Need. Once the user knows (or rather thinks that he knows) what he is searching for, he formulates a request. A *request* is a natural language specification of what the user wants to find, something that might be given to a knowledgeable librarian or an expert in the field. A *request* is a way of communicating the *PIN* to another human being. Finally, this request has to be turned into a *query*, which is something that can be recognized by a search engine, perhaps a list of key words, a boolean expression or an SQL query. A query is a way of communicating the request to a machine.

- *Time.* The third dimension of relevance reflects the fact that searching is not a one shot process. The user may not see any relevant items in the initial retrieved set, and this may prompt him to re-formulate the query, perhaps changing the boolean structure, or adding some additional keywords. If the user does find relevant items, information in these items may prompt him to formulate different requests, or perhaps even force him to re-think what it is he wants to find, thus changing the perception of his information need (*PIN*). The real information need (*RIN*) stays constant and unchanging, since it is refers to what the user will ultimately be satisfied with. Mizarro endows the third dimension of relevance with a discrete progression of time points: $\{i_0, p_0, r_0, q_0, q_1, ..., r_1, q_{k+1}, ..., p_1, q_{m+1}, q_{n+1}, ...\}$. Here i_0 refers to the time *RIN* came to existence, p_0 is the time when the user perceived it, and decided what he wants to search for, r_0 is the time when he formulated a natural language request, and q_0 is the time when that request turned into a query for a search engine. Proceeding further, q_1 is the time of the first re-formulation of the request into a different query, r_1 is the first re-formulation of the natural-language request, and p_1 is the first time when the user changed the perception of what he wants to find. A request change ri is always followed by one or more attempts to re-formulate the query q_j, and for every change in user's *PIN* there is at least one attempt to formulate a new request.

- *Components.* The final dimension of relevance in Mizarro's framework specifies the nature of relationship between the first and the second dimension. It can be *topical relevance*, *task relevance*, *context relevance*, or any combination of these three. Topical relevance is concerned with semantic similarity in the content of the two items. Task relevance specifies that the item or information contained in it is useful for the task the user is performing. Context includes anything that is not covered by the topic and the task. Mizarro's context is a kind of "miscellaneous" category that subsumes the notions of novelty, comprehensibility, search cost, and everything else that does not seem to fit elsewhere in his formulation.

This conceptualization of Mizzaro is in fact a framework that helps us to classify and compare different definitions and implementations of relevance. For example, in Chapter 6 when we work on the keyword search on structured data we can easily adopt his framework to structured retrieval. In this case, the document in the dimension of information now corresponds to *Joint Tuple Tree* (JTT) and each surrogate of the JTT actually corresponds to the individual tuples to be joint together in the search process. Similarly we match the query to a conjunctive query while keeping the request as a keyword query to be stated by the user. Therefore, the framework can easily be adopted to other retrieval scenarios although in the conceptualization it is stated to be only document retrieval.

In practice, Mizzaro's four dimensional model is adopted partially in many retrieval models in IR. Lavrenko's work on relevance models [131, 133] is based on the idea of modeling topical relevance (component) by replacing the RIN (request) with a hidden representation space that is able to generatively create both document and queries. Current works on personalization and discriminative models of relevance (i.e. learning-to-rank) is actually based on modeling the context relevance (component) w.r.t. the query (request) [107, 178, 213, 220]. In particular recent approaches, which mine search engine logs to capture the user's real intent using the query chains, is an actual implementation of the time dimension of relevance where the gap between the queries and the RIN is taken into account [178, 187].

3.2 Current Approaches for Relevance

3.2.1 Textual Relevance

The best known and widely applied type of relevance is *textual relevance* which considers the actual content of the document and applies a number of ranking models as we review at Sec. 2.2.2.1. Modern search engines include tens or hundreds of ranking functions that can measure such a relevance. Some of these functions depend only on the frequency of occurrence of query terms; while others apply more sophisticated means based on the page structure, term positions, graphical layout, etc. As we discuss above, one of the earliest textual relevance indicators is the vector space model (VSM) mainly scoring the document based on its term frequency.

One drawback of the VSM is that the relevance is not explicitly modeled but implicitly implied. That prevents the VSM to be extended to reflect different dimensions of relevance. This issue lead to probabilistic models of relevance which have been pioneered by *the Probability Ranking Principle of Robertson* [195]. It is at the foundation of almost every probabilistic model in IR and intuitively defines a principle that a search system should present the documents in order of their probability of being relevant to the user's request. In [195] it is defined as follows:

> The Probability Ranking Principle (PRP): If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the

> system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.

In other words, it defines the probability of relevance of a document denoted as $P(R = 1|D)$ where D is used to denote every observable property of a given document, and R is a binary variable indicating whether or not that document is relevant to a request. Then, optimal performance of a search engine is assumed to be achieved by ranking documents in decreasing order according to this probability. This intuition has become influential in the emerging ranking models of IR such as probabilistic models or language models. Based on PRP, Robertson and Jones [196] introduce *the binary independence model* (BIM) that has been extended by van Rijsbergen [229]. Here *binary* represents the boolean representation of documents and queries as term vectors. *Indepencence* means that terms are modeled as occurring in documents independently. However, the model has been re-formulated a number of times since it is first proposed and these both assumptions are not valid anymore. The BIM model was originally designed for early library collections of documents and abstracts and it is practically effective in this respect. However, for Web search engine the model is not a good fit since it ignores term frequencies and document length. Thus, BM25, also called Okapi weighting, is proposed as a way of building a probabilistic weighting to these quantities [111].

A more mature model has been adopted from statistical models of natural language in which language modeling is a mature field with a wide range of successful applications, such as discourse generation, automatic speech recognition and statistical machine translation. The use of language modeling for IR was first proposed by Ponte and Croft [179] in 1998. They took quite a radical step by removing the explicit relevance variable R of early probabilistic models. Instead, they construct a probabilistic model around the document and the user's query by hypothesizing that for every document D, there exists an underlying language model M_D. Then the document D is considered to be relevant to a query Q if the query Q seems to be a random sample from M_D. In other words, a document model M_D represents the information content that the creator of the document has formulated and if it is likely to produce the query, then it is considered to be relevant.

The interplay among the query, document and relevance variable has later been revised by Lavrenko [133] in a generative model of relevance. In his work, he extended the language modeling and has introduced the explicit relevance variable back into play. However, instead of modeling relevance as a binary variable, it is now considered as a *hidden representation space* as we discussed before in Sec. 2.2.2.1. This assumption is formulated in a form of hypothesis, called *Generative Relevance Hypothesis*, and stated as follows in [131]:

> The Generative Relevance Hypothesis (GRH): For a given information need, queries expressing that need and documents relevant to that need can be viewed as random samples from the same underlying generative model.

This intuition provides an innovative way to perceive the relevance and also a framework that can easily be extended. In particular, there are two main components in order to construct a ranking model, called *relevance model*, based on GRH. First we need to specify a form of representation for this hidden representation space. In [131], this space is also represented by a

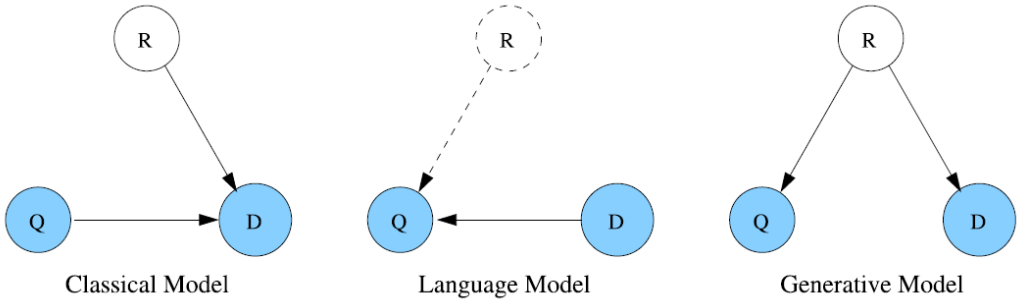


Figure 3.1: Graphical diagrams showing dependencies between the query Q, the document D and relevance R variables in different probabilistic models of IR. Left: classical probabilistic model [196]. Middle: language modeling framework [179]. Right: the generative model [131] (Shaded circles represent observable variables)

set of relevant documents which is obtained either from user via relevance feedback or from a search engine via pseudo-feedback. In fact different forms of representations can be employed in the hidden representation space as we discuss in this work. The second component for this model is a generative model which responsible for generatively producing both documents and queries according to the representation of hidden representation space. Lavrenko showed that different models can be employed as a generative model such as unigram model, mixture model, Dirichlet model or topic models (e.g. LDA). Fig. 3.1 illustrates the intuitive differences among those different probabilistic models.

3.2.2 Hyperlink Relevance

The Web graph as a linked collection of documents is an important source of information to specify a relevance score, called hyperlink relevance. The intuitive idea is that a hyperlink linking an anchor text in the source page to a target page is considered as a reference, or a vote by the source page on the target. In addition, the anchor text is considered as a description or an explanation of such relevance. By leveraging the Web graph, the search engine can determine the importance of a page independently on the textual content of the pages. This idea was originally proposed by [156] and further exploited by HITS [121] and Pagerank algorithms [175]. Later Pagerank is extended to consider the topical information in order to bias hyperlink relevance towards the topic of query [89].

Hyperlink relevance is one of the early indicators of the popularity score on the Web since those pages that have more links gets more score and accordingly assumed to be more relevant. However, this type of relevance has important drawbacks in terms of matching users' information needs due to the content not being directly considered. For example, if a user wants to get the contact details of "Rudi Studer", the hyperlink relevance of the Wikipedia page for "Rudi Studer" will be higher than his homepage at AIFB Institute because Wikipedia as a domain gets more links than the AIFB Website. However, the latter will be more relevant to the user as it contains more up-to-date information including the contact details. In fact, recent studies

show that the hyperlink relevance may not be so appropriate on certain tasks. In [194], it is shown that hyperlink relevance can be outperformed by using a learning-to-rank algorithm on the simple document features. Amento et al. [5] found similar results that simple features, such as the number of pages on a site, can be as determinant as the hyperlinks. Furthermore, in [227] it is shown that for the task of finding home pages, the type of URL were as, or more, effective than hyperlink relevance.

3.2.3 The Discriminative Models of Relevance

Mining user search activity has a huge potential to estimate the relevance and predict user satisfaction. There are many variables that can be observed in a search process such as the time user spends on the page and whether user has reformulated his or her query, but the most important observation is clickthrough data which can be considered as a vote on relevance of documents [110]. This information, also called as *implicit feedback*, is maintained by the search engine in *clickthrough logs*, which basically record a temporary user id, the query issued by the user, the results returned by the search engine and resulting user clicks.

Actually, the information in clickthrough logs has a broad range of applications including Web result pre-fetching, click spam detection, or prediction of user satisfaction, but the most important application is the estimation relevance because it helps to construct a training dataset on which we can further train [117, 187]. For example, when the user issues a query "Audrey Hepburn" and clicks on the third link from the received list of search results, this is an indication that the third document is more relevant than the first two according to user's decision process. This pairwise preferences become training data for the learning-to-rank (LTR) approaches that we introduced in Sec. 2.3.4. Then LTR trains a *discriminative model* (i.e. ranking function) based on this data that can predict the degree of relevance for any given query and document. Intuitively speaking, this discriminative model of relevance is dependent on two sources of information: First, it is naturally based on clickthrough logs as just discussed. The other information is the features associated with training data for learning algorithm. According to [110], the top features employed for learning includes TFIDF scores, rank of document in Google search engine, domain or origin of the URL, or Pagerank score which in nature can be considered as the mixture of other types of relevance. Although the discriminative models of relevance is an effective approach to utilize user-based relevance, in practice it simply acts a combination of other relevance types and thus is limited to their optimal mixture.

3.2.4 Personalized Relevance

Considering the user interest, background and activities are the main motivation for *personalized relevance*. In fact, it is a direct implementation of user-oriented definition of relevance. Although similar techniques of learning and information sources are also used in this type of relevance like discriminative models of relevance, the unique thing for personalized relevance is that it maintains a user profile for each individual. User profile can be learned from a number of resources such as 1) demographic and geographical information of the user, 2) search history, including previous queries and clickthrough information, and 3) other user documents, such as bookmarks, favorite web sites, visited pages, and emails. Since it is difficult for the

user to specify this information, the way of automatically learning user profiles from a user Web search activities is main interest in many works. Claypool et al. [46] examined the connection between explicit judgments of quality and implicit indicators of interest such as time and scrolling for general Web browsing activities. Joachims et al. [108] found the relationship between clicks and explicit judgments for a search task. In some other work [3, 73] more complex learned models to predict relevance judgments or preferences using a variety of implicit measures including clicks, dwell time and query reformulations are considered.

Based on learned user profiles, personalized relevance is measured calculating a re-ranked search results. In most of the work in this direction [45, 184, 209, 217], personalization is achieved by content analysis in which the user's topical interests are matched with topics of the document and results are filtered and re-ranked accordingly. Teevan et al. [220] and Chirita et al. [45] also exploit rich models of user interests, including documents and emails from local sources, organized in hierarchical keyword categories. Another type of personalization is also achieved by extending the abovementioned hyperlink relevance. For example, the Pagerank algorithm is adopted by setting a variant preference to Web pages to create personalized Pagerank vectors and to calculate a personalized score for each user based on his or her preferences [89, 104]. Since Pagerank is computed offline it is impossible to apply this to Web setting as a response to user query. That's why in [104] the concept of considering user's preferences as a subset of hub pages are presented. Then a set of hub vectors are computed offline and linearly combined to calculate the personalized score.

Despite its conceptual support from the user-oriented view of relevance and recent increasing interest in personalization approaches, the effectiveness of the personalization is an open debate as we also discussed in Sec. 2.2.2. It is mainly the issue that users will have different interests at different time frames and it is hard to predict and rely on the topical relevance in a particular search session [221]. That's why identifying the individual queries whether they can be personalized or not is an open research challenge so far.

3.2.5 Cluster Relevance

The idea of document clustering to improve search performance dates back to the cluster hypothesis which simply states that "closely related documents tend to be relevant to the same request" [102]. Based on this hypothesis, many works has been proposed to exploit *cluster relevance* among the search results, notably [62, 125, 146, 231]. Clusters represent structure within a document set, but do not directly induce an obvious single criterion or principle by which to rank documents; for instance, they have been used to improve rankings indirectly by serving as smoothing mechanisms in language models [146]. In addition, structural re-ranking of the search results is done based on clustering where it computes and exploits a clustering of the initial run of retrieval results [62, 125].

3.3 Search Relevance based on the Semantic Web

Summarizing the definition of relevance and current approaches to implement it, we can plot Fig. 3.2 to reflect main intuition of relevance. On the left side of the figure, we see the user

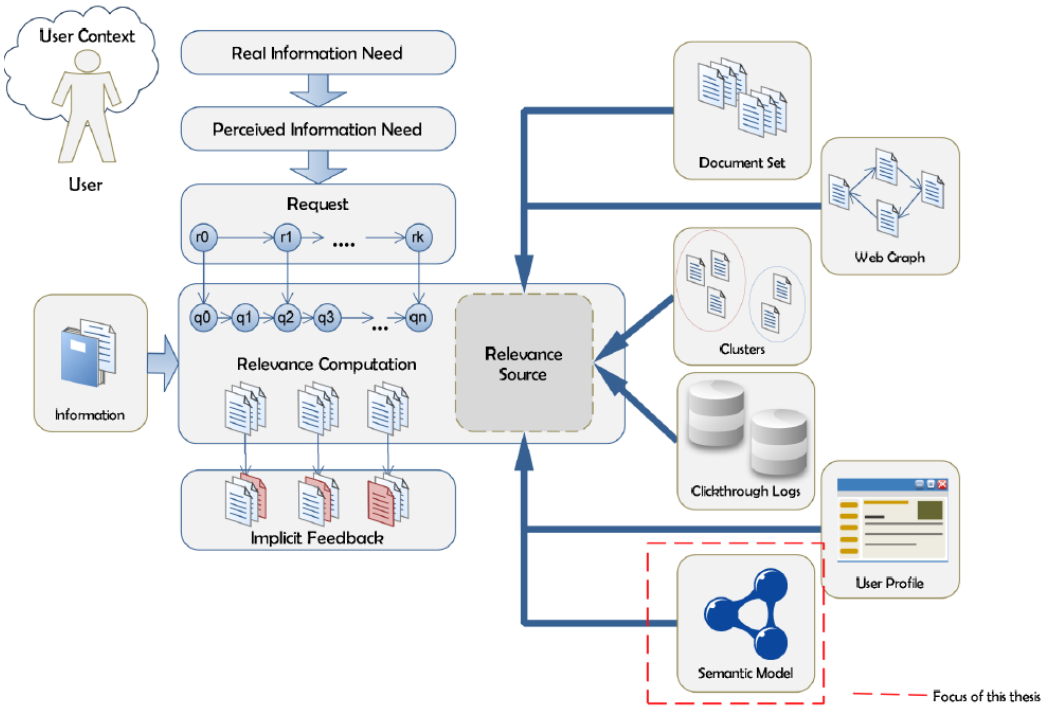


Figure 3.2: An illustration of Mizzaro's four dimentional model of relevance together with current lines of approaches to predict relevance. Using semantic models as relevance source is the main focus of our work.

and user context which indicates the initial setting of a search process. According to Mizzaro [165], we know that user's information need can be perceived in four different types: RIN is constant and PIN is derived from RIN. There are a number of requests as indicated by the time dimension and every request results in a number of queries processed by relevance technique. Different techniques exist to compute a relevance score according to the query and every one of them depends on a particular relevance source. In the previous section, we review five different relevance types and their relevance sources are conceptually shown in the figure. In practice, more than one relevance source is used as a result of the multidimensional nature of relevance and relevance assessment is done by aggregating those multiple criteria [57].

An underlying goal to any search engine is that the observations performed in the model of relevance should correlate as much as possible with equivalent observations by actual users. In this regard, it is natural to consider the idea of reducing the distance between the relevance representation in the system and the RIN in the user's mind w.r.t. the formulation of queries and the understanding of documents. This problem is a complex problem, not just by lack of ideal representation of the information and the need of automation to process that information, but also due to the involvement of diverse, difficult to capture, aspects related to human cognition, and even the definition of reality, truth, and meaning. At this point, semantic search aims to replace the underlying relevance source with a *semantic model* that is supposed to be more

related to human cognition. This is in contrast to widely adopted bag-of-words representation in the IR field by which the relevance between information need and information is determined based on literal matching of the words.

3.3.1 Approaches using Semantic Models as Relevance Source

One of the early attempts to use a semantic model is made by Croft [54] in which the domain knowledge is modeled as a thesaurus of concepts, each one of which has a name, relationships to other concepts, and a list of more or less ad-hoc rules to recognize the concept in a document. The main types of relationships only include synonymy, hypernyms, hyponymy, meronymy and similarity. By using these relationships, word occurences are replaced by word senses (i.e. concepts) and used to expand both queries and documents during indexing and retrieval. Later, Voorhees [232] carried out experiments to exploit the semantics contained within WordNet for query expansion in order to improve retrieval effectiveness by indexing with word senses. Experiments are based on TREC collections, in which all terms in the query are expanded by a combination of synonyms, hypernyms and hyponyms. Although the performance on short queries is improved, no significant improvement is achieved for long queries indicating non-robustness of the approach. Gonzalo et al. [84] used Wordnet senses to index the documents and apply retrieval based on these senses. However, in this work, they rely on manually disambiguated queries and documents that limits the applicability of the approach. Despite this, the retrieval performance has improved by 30% which shows the effectiveness of a semantic model. Further improvements are also obtained in the follow-up works such as [120, 145, 207] recently. Although theseaurus-based approaches are shown to be effective to solve the ambiguity problem in both queries and documents, their scope is only limited to that purpose due to the nature of conceptual information contained in those models – i.e. words and word senses. In other words, for any unambigous query and document, their prediction power of relevance is similar to what a keyword-based technique can achive, because matching word senses is nothing but matching words in the case of unambiguity [14].

The use of expressive ontologies as semantic model to shift the Web search beyond keyword-based capabilities has also been often considered scenario in the area of Semantic Web [149, 224]. Although the idea looks appealing in the first sight due to the power of ontologies describing domain-specific knowledge, in practice many challenges exists. In particular, a common way of *ontology-based search* is to assume that all the information is stored in a knowledge-base (KB) in a specific format (e.g. RDF) and conforming to an expressive ontology (e.g. OWL) and users specify (semi-)structured queries (e.g. SPARQL) for their information needs. The semantic relevance is mainly obtained by the underlying query engine which receives the query and executes it on a KB. In this respect, this line of approaches can be regarded in the spectrum of data retrieval (as in the sense of relational databases) rather than IR (Sec. 2.1). Under this perspective, some prominent approaches include *semantic portals* [41, 154, 216] or *YAGO-NAGA knowledge discovery* [115, 116]. However, these approaches utilize the actual semantic information (i.e. ontological axioms) to retrieve the structured query processing and ranking according to relevance is mostly regarded as a low degree of relevance. In fact, [216] employs a ranking scheme for ontology triples based on term frequency of an entity label in a relation type. Similarly, in [116] a language model based ranking is applied based on the prob-

abilities of ontology triples occurring on the witness pages (e.g. Wikipedia pages) processed by the YAGO extraction algorithm. There exists two main drawbacks of these approaches in their applicability on Web search: First, they assume a deterministic retrieval (e.g. structured query processing) which is decoupled from the actual ranking scheme that probabilistically determines the relevance to user's information need. Also in those ranking schemes, the relevance of textual content to the information need is mostly ignored by only focusing on triple-based probabilities – i.e. similarity measure between user query and ontology results are not clearly justified in comparision to standard IR-based relevance techniques. More importantly, the implicit assumption made by these approaches is the fact that the whole information is to be fully represented as a formal knowledge base which is not affordable with the current volume of unstructured information worldwide – e.g. in [191] it is argued based on experiments that by using the state-of-the-art methods such a conversion of the size of one Terabyte unstructured data would take 388 years to complete. In addition, unstructured data includes more information than its structured representation and an inevitable loss of information occurs when they are replaced with a relatively small number of axioms. Furthermore, more and more unstructured data becomes part of the Web databases where both structured and unstructured forms of information becomes substantial leading to a more hybrid data. Therefore, the dominance of one or the other form of information is a highly impractical assumption.

As another line of work, there exist ontology-based approaches which consider keeping the textual information (i.e. documents) and the KB seperated with the help of *semantic annotations* relating the text to a semantic model. KIM semantic annotation platform, for instance, focusses on the automatic population of documents on large scale and a ranking model is also utilized on top of a Lucene-based[1] IR system to index and retrieve based on annotations. Another complementary work to KIM is proposed in [40, 228] proposing a ranking model by adopting VSM that utilizes semantic annotation frequencies as weights instead of terms frequencies. Recently Meij et al. [162] presented an approach adopting Lavrenko's generative model of relevance to construct *conceptual language models* (CLM) by employing the document-level annotations as concepts in the representation space (i.e. relevance). It is unique in the sense that semantic relevance is modeled exactly as concepts annotating documents and both documents and queries are considered as random samples generated from the concepts. However, one problem of this model actually occurs in the estimation of the so-called *generative concept model* which is trained as a unigram model derived from all relevant documents annotated with that specific concept. Since these documents may contain many terms not necessarily related to the concept, an expectation-maximization (EM) based training is also used to find exact probabilities similar to the model-based feedback method presented in [254]. Thus it requires all relevant documents of the concepts to be pre-given which is only possible with a limited scope of document corpus such as those in medical domain (e.g. TREC Genomics [92] or CLEF domain-specific track [177]). This prevents its applicability to more general scenarios such as ad-hoc document retrieval. Broadly speaking, these semantic annotation based approaches also have a similar bottleneck as the abovementioned ontology-based approaches in terms of preprocessing all documents to be annotated since they are used as training data to estimate the degree of semantic relevance. In addition, the experiments in [162] reveal that

[1] http://lucene.apache.org

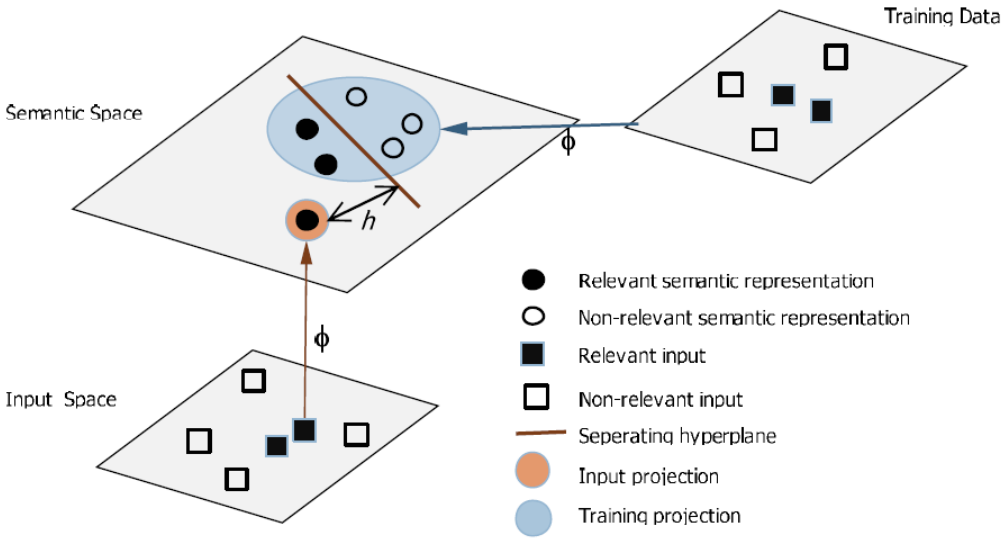


Figure 3.3: Discriminative semantic relevance

CLM shows very similar improvements as the standard relevance models and even performs poorer in some of the datasets.

3.3.2 Semantic Relevance

Although the intuitive idea of using thesearus, ontologies, or semantic annotations is useful to specify semantic relevance, in practice it is a big challenge to apply current techniques to important IR scenarios, specifically ad-hoc retrieval. Especially when we consider the gap between the available data on the Web and the requirements of current approaches, a pragmatic view would be to develop a more robust and well-founded technique instead of expecting the informaiton to be in a specific form (e.g. ontology). In fact, existing Web of data available as relational databases, RDF based linked open data or XML resources can all be used as a condensed semantic model. However, the actual issue lies on developing an IR technique that is scalable and also comparable to the well-known models of relevance currently in-use in Web search.

In particular, semantic relevance can be considered as an extension of textual relevance described above since both focus on the content of the information while the former relies on a more comprehensive relevance source instead of using bag-of-words. In this regard, we base our model on top of the existing models of probabilistic relevance that provides a well-founded probabilistic theory underlying these models. Intuitively speaking, we extend Lavrenko's generative relevance model which represents relevance as a hidden representation space (Fig. 2.4) that can be replaced by a semantic model to capture relevance as discussed in Sec. 3.2.1. Therefore, we choose to formulate this assumption in the form of a hypothesis by extending the Generative Relevance Hypothesis with an explicit semantic model. Furthermore, we do not

restrict the hypothesis to a specific information type (i.e. documents) but generalize it to any information type since we also propose a model to search structured data in Chapter 6 based on the following hypothesis:

> The Generative Semantic Relevance Hypothesis (GSRH): Given a semantic model and a projection of an information need on that semantic model, both queries and information items relevant to that need can be considered as random samples from the same underlying generative model derived from the projection.

In other words we state that the relevance can be determined using a semantic model which has the ability to generate both query and information items. In order to realize such a model, we need these components to be specified. First, we need a formal representation of the semantic model. For this, we employ an RDF-based graph structure as it is a denormalized representation of relational data where the semantics of entities (i.e. class and relationships) are explicitly encoded. Second, we need to have a generative model that enables the generation of text from a selected representation subspace – i.e. the projection of information needs on RDF graphs. The success of the proposed relevance model is highly dependent on this generative model as it assigns the actual density allocated to each word. In fact, we can learn such a generative model directly from the RDF graph as it includes both the structure and the textual information associated with that structure. In Chapter 4 we present such a model that uses an RDF data graph for training such a generative model. We show that GSRH holds as we apply it on unstructured and structured data retrieval and obtain significant improvements over the existing models of relevance in Chapter 5 and Chapter 6, respectively.

Besides, our view of semantic model can also be utilized for a discriminative model of relevance to determine the relevance based on the semantic content of training data obtained from, e.g., clickthrough logs. While a generative model of relevance makes an explicit assumption of how the query and information items are generated and constructs a relevance model based on that assumption, discriminative models typically make very few assumptions and in a sense, configure a model using the data itself [170]. A discriminative semantic relevance can be realized by mapping both the training data and the documents to be ranked (i.e. input data) to the semantic space as shown in Fig. 3.3. Based on this mapping, a discriminative model is to be learned by using the semantic information (i.e. features) that is associated with the training data. For this a learning algoritm is needed that can learn by using the information of the semantic model. Then the relevance of the any input (i.e. query and document) is to be determined by the discriminative model using their mapping to the semantic space (i.e. semantic features). We formulate this as a hypothesis as follows:

> The Disciriminative Semantic Relevance Hypothesis (DSRH): Given training data, the relevance of a query-document input can be discriminatively determined based on the mapping of that input on a semantic model and the mapping of the training data on the same semantic model.

In Chapter 7 we propose a learning algorithm, called *Semantic Kernel Machines*, that trains a discriminative model using the RDF data graph as semantic features of the training data. In

addition, we show how such a model can be used in a new learning-to-rank approach, called *Learning-To-Rank with Semantic Kernels*, based on DSRH.

3.4 Conclusion

In this chapter, we provided a comprehensive introduction to the relevance problem in the context of IR and Web search and discuss different sorts of approaches that aim to offer a solution. In particular, we identify five different types of relevance each of which provides its unique assumptions to the notion of relevance. We also discuss how these approaches relate to the unified view of relevance. Semantic relevance is offering a new outlook to the problem with a potential of being more intuitive way to represent and quantify relevance with the help of semantic models. Although the initial ideas in this direction are proposed long time ago, they are mostly neglected by the traditional IR community. With the advent of the Semantic Web, a new sort of relevance source is underway emerging as Web of data which provide a condensed representation to the World of knowledge. This intuition is a starting point to the work presented in the following chapters where a number of techniques are proposed to utilize Semantic Web for search relevance.

4

Topical Relational Models

Crossing the structure chasm and managing structured and textual data in an integrated way is a major theme in current research that recently has fueled a large amount of work, especially in terms of database and Information Retrieval (IR) integration. One major challenge to this end is bridging the "semantic gap" between text and structured data (or simply data, for brevity). This is in fact the classic Natural Language Processing (NLP) problem in disguise where the goal is to obtain a more precise understanding of text and to represent it in a way that can be processed by machines (e.g. as structured data) [155]. This “structured interpretation” of texts then can be utilized for different NLP tasks such as document classification, machine translation, retrieval, or question answering. For dealing with this problem, different directions for data extraction have been explored. High quality results can be obtained by using sophisticated machine learning techniques for inducing extraction patterns [56, 208, 212], and by using probabilistic graphical models for sequence tagging [126, 158, 204]. Yet, scaling these solutions to the large-scale setting where the amounts of text, data and data extraction patterns to be managed is large and heterogeneous, and where training data is difficult to obtain, has proven to be a hard problem.

Meanwhile, the amount of data available in databases and warehouses is continuously increasing. Recently, trends in publishing and making these data publicly available on Web, e.g. Linked Data [24], which includes large amounts of structured data published by the US and UK governments and encyclopedic sources such as DBPedia [7] and Freebase [30], suggest that much of the structured interpretation is already available and can directly be reused. Instead of extracting new data from text, bridging the gap can be partially reduced to the more tractable problem of linking text to existing data (or vice versa). However, only little work can be found in this direction that can be classified as *entity linking*. In particular, it is defined as matching a textual entity mention, possibly identified by a named entity recognizer, to a knowledge base entry that is a canonical entry for that entity. A simple solution to this is to exploit Wikipedia knowledge for linking entities to the text by using some features from the associated text to the entities and to train a classifier to make predictions [34, 55, 163]. However, the current line of work on entity linking is highly tailored to the underlying knowledge base in terms of the features the algorithms depend on. This limits their applicability to other knowledge bases and does not provide a generic solution to the problem. In addition, the entity linking

only concerns to associate the text to an entity and says so little about other semantic information available in a knowledge base – i.e. the correlations between the text and structured data such as class membership, or relationships among the entities.

To this end, understanding the structured data and extracting interesting patterns out of it have been an interest of *Statistical Relational Learning* (SRL) [80] that utilizes the structure (i.e. schema of the domain) of the relational data as a template to model a joint probability distribution over that data. With the ability of handling both uncertainty and structure, SRL approaches support to answer queries using probabilistic inference and have been successfully used for a number of tasks such as collective intelligence, link prediction, entity resolution and so on. However, the major shortcoming of the SRL approaches is their inability to handle the textual data emerging as attributes of the relations in database. In fact, all of these approaches have an assumption that the descriptive attributes of the entities have values from a certain domain of categories (e.g. year – 70s,80s,90s; location – USA, Germany, Japan; movies – action, drama, comedy) and probability distributions are specified over all these possible categories which does not hold for long textual attributes coming as bag-of-words. Actually, it is highly apparent that the textual data associated with the entities and the local structure is highly correlated. As more and more textual data is present in the databases, it becomes difficult for previous SRL approaches to properly define a joint probabilistic model as they mostly neglect textual attributes.

In this chapter, we approach the problem of bridging the gap between the textual and structured data by considering any RDF-based knowledge base[1] as training data and learn a generative probabilistic graphical model to capture the correlations between these two worlds of data. For example, consider Fig. 4.1 which shows a snapshot of actual entry for "Audrey Hepburn" entity from freebase.com. The shown entry includes several attributes of the entity such as date of birth, or height as well as the relationships to other entities such as place of birth, or religion. Although such structural information is common to any database, there are also unstructured information associated as textual data like the description for "Audrey Hepburn" from Wikipedia. This information in fact can all be represented as RDF representation as shown in Fig. 4.2.

In this work we consider that the existing text describing an entity and its structural information are not totally independent but correlated. For example, the words in the text of "Audrey Hepburn" such as "Ixelles" or "Belgium" are related to her place of birth indicated as a relationship to the entity "Belgium" in the structured data. However, due to the nature of text in an unstructured form, it is a challenge to encode such correlations in a probabilistic model because there are many words to be associated to every structural element of an entity. In order to achieve this we utilize topic models that present a low-rank representation of the text and draw topical correlations from text to structured data in order to capture underlying semantics of the RDF data. In particular, we propose a new SRL model called *Topical Relational Model* (TRM) that considers relational learning and topic modeling within the same generative probabilistic model in order to extract those correlations via a Bayesian-based learning. This way, TRM is able to utilize any RDF dataset to train a full probabilistic model and reveals important patterns of the text and structure altogether.

[1] Actually in Chapter 6 we show that any relational database can be transformed into a RDF representation.

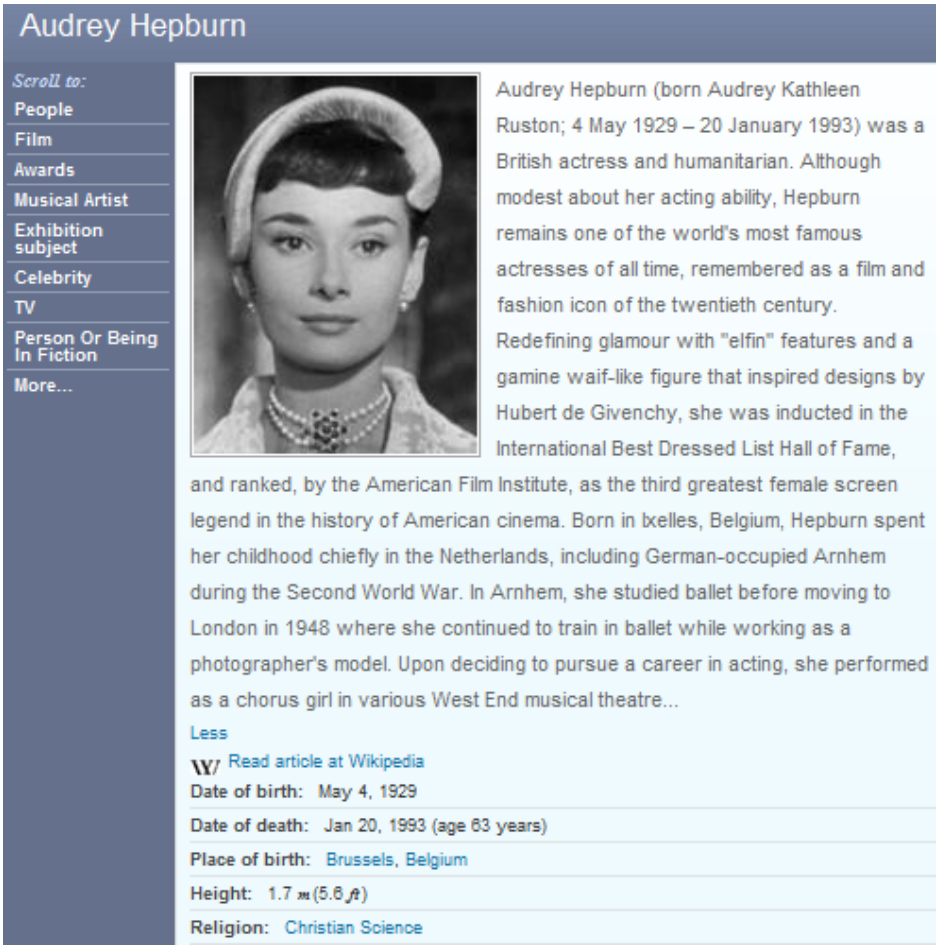


Figure 4.1: Entry for Audrey Hepburn at Freebase (Source: www.freebase.com)

The rest of this chapter is organized as follows: In Sec. 4.1 we introduce our data model and further preliminaries related to this work. Then, in Sec. 4.2 we present TRM as a template-based construction of probabilistic graphical model and show how the topical correlations can be introduced in the model. In addition, we discuss the learning of TRM using a Bayesian inferencing technique. Sec. 4.3 presents the experiment results of the trained model and finally, we discuss how TRM differs from existing approaches in Sec. 4.4 and conclude in Sec. 4.5.

4.1 Data Model and Problem Definition

4.1.1 RDF Data

Web data emerging in RDF format has unique characteristics which is the main starting point in this chapter. The structure of RDF can be regarded as a data graph in which the entities (i.e.

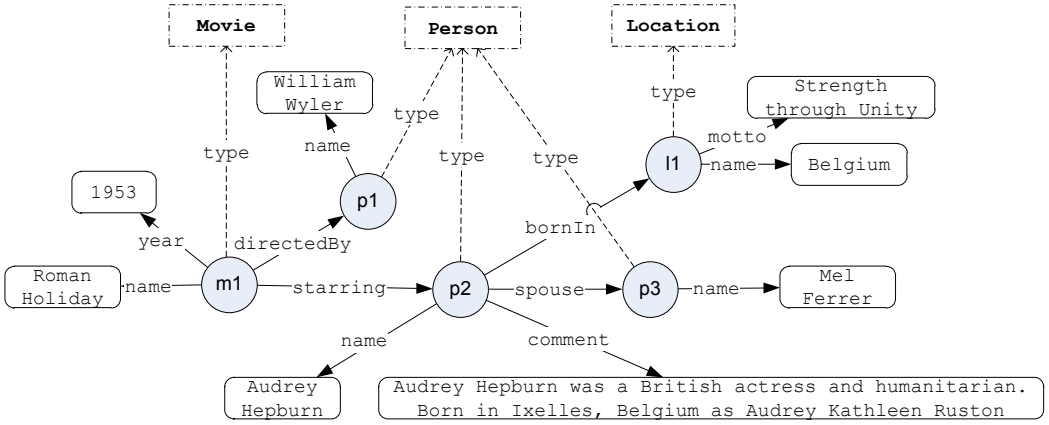


Figure 4.2: An example RDF-based data graph

actual data) constitutes the basic unit of information and represented as nodes. Each entity is labeled with an unique id and can be connected to the other entities via the relationships. Besides, every entity can have attributes, which are textual elements in the RDF graph to describe the entity further with unstructured data. Both attributes and relationships are denoted by edge labels in an RDF graph and edges are directed from a source to a target of an edge. A special relationship, *type*, also exists to indicate the class (e.g. *Person*, *Location* etc.) that an entity belongs to. Note that an entity might have more than one *type* relationship associating multiple classes to that entity. Figure 4.2 illustrates an RDF data graph with five entities and their associated relationships, attributes and classes. The edges in the RDF graph correspond to RDF triples, such as a triple <*Roman Holiday, starring, Audrey Hepburn*> denotes a relationship between the entities $AudreyHepburn$ ($p2$) and $RomanHoliday$ ($m1$). There can be more than one relationship between the entities and they can belong to more than one classes, – e.g. $Belgium$ is both a $Location$ and a $Country$. We formally define an RDF data graph as:

Definition 1 *(Data Graph) Let $\mathcal{L}_V$ and $\mathcal{L}_E$ be finite sets of vertex and edge labels respectively. A data graph can be represented as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, l_V)$ where $\mathcal{V}$ is a finite set of vertices, $l_V : \mathcal{V} \rightarrow \mathcal{L}_V$ is a vertex labeling function, and $\mathcal{E} \subseteq l_V(\mathcal{V}) \times \mathcal{L}_E \times l_V(\mathcal{V})$ is a set of labeled edges. Furthermore, the set $\mathcal{V}$ of vertices is defined as disjoint union $\mathcal{V} = \mathcal{V}_C \uplus \mathcal{V}_E \uplus \mathcal{V}_D$ of classes, entities, and data values, respectively.*

We also denote the set of classes of an entity $C(e)$, where $e \in \mathcal{V}_E$ and $C(e) = \{c \mid (e, type, c) \in \mathcal{E}\}$. Similarly, the set of relationships in which an entity is involved is denoted by $R(e) = \{r \mid (e, r, e'), (e', r, e) \in \mathcal{E}\}$ for $e', e \in \mathcal{V}_E$. In addition we denote the set of relationships in which the entity e is the source as $R^-(e)$, and the target as $R^+(e)$.

4.1.2 Topic Models

For every entity, we consider that it has text (e.g. attributes, context, related documents etc.) associated defining the entity w.r.t. the bag of words as unstructured data. Like in the case of

topic modeling, the assumption here is that this associated text is generated from a number of hidden topics that represent the semantic themes of the entity.

Definition 2 *(Topic) A semantically coherent topic in a text collection C is represented by a topic parameter β which is a probabilistic distribution of words $\{p(w \mid \beta)\}_{w \in V}$ and $\sum_{w \in V} p(w \mid \beta) = 1$ where V is the vocabulary of collection C.*

Our aim is to train a hybrid probabilistic relational model that captures both topics and relational structures in a probabilistic model by analyzing the corpus and exploiting the semantic structure available in a RDF graph. In fact, a topic model is a probabilistic model that uncovers the underlying semantic structure of a text collection based on Bayesian analysis in order to learn probability distributions over the words. Early versions of the topic models such as LDA [28] utilizes an unsupervised clustering of the similar words into the topics. In addition, the supervised topic models such as supervised LDA [27], or Labeled-LDA [190] also exist. An illustration of the topic models is previously introduced in Sec. 2.3.2.

Although it is useful to discover co-occurrence patterns of word use and to connect documents that exhibit similar patterns, the plain and coarse granular structure of topics provides a limited support in the development of further applications in IR [238]. On the other hand, an RDF data graph provides an important source of information that represents the text along with a structured form. The problem addressed here is to learn from both text and structure in a way to enable more biased topic models towards the structural information and also capture the associations between those topics and structural elements.

4.1.3 Probabilistic Graphical Models

Our work builds upon the research on probabilistic graphical models that provides a flexible and yet compact representation for complex probability distributions. We begin by introducing the classes of *random variables* and *conditional probability distributions* as the main building blocks. Let $\mathfrak{X}$ denote some set of random variables, $\mathfrak{X} = \{X_1, ..., X_n\}$, that any $X_i \in \mathfrak{X}$ can be assigned a value from a predefined domain of assignments denoted by $Val(X_i)$. For the set $\mathcal{X}$ of random variables, we denote $Val(\mathfrak{X}) = Val(X_1) \times ... \times Val(X_n)$.

Most SRL approaches are in fact specializations of a *template-based probabilistic graphical model* (TPGM) that indicates that the resulting graphical model conforms to a template derived from the relational structure [122, Ch. 6]. In this sense TPGM provides a more general framework and foundations to specify further richer models. As we also follow a similar approach in this work, here we define the fundamental building blocks of TPGM. The key concept in TPGM is a *template attribute* that defines a group of random variables that share similar signature and semantics.

Definition 3 *A* template attribute *(or attribute hereafter) A is a function $A(\alpha_1, ..., \alpha_k)$, whose range is some set $Val(A)$ and where each argument α_i is a variable associated with a particular set of values from $dom(\alpha_i)$. The tuple $\alpha_1, ..., \alpha_k$ is called the argument signature of the attribute A, and denoted by $sig(A)$. We denote the domain of attribute A by $dom(A) = dom(\alpha_1) \times ... \times dom(\alpha_k)$.*

Here we consider without loss of generality that the number of arguments in the signature of an attribute uniquely identifies the nature of the attribute and each argument is associated to a particular domain of values $dom(\alpha_i)$. For example, in our RDF graph an attribute like $Person(\alpha_1)$ is an attribute of a single argument and α_1 takes values from the set of all entities (i.e. $dom(\alpha_i)$). In fact, we normally do not consider all the possible values in a domain, but restrict it to a subset of values available in the data graph, called *object skeleton* of the attribute. Then based an attribute and its object skeleton, a set of random variables are defined in the ground graphical model:

Definition 4 *Given an attribute A, an object skeleton $\mathcal{O}(A)$ specifies all possible assignments to the arguments in the signature $sig(A)$ such that $\mathcal{O}(A) \subseteq dom(A)$. We define a set of random variables, called template variables, for $\mathcal{O}(A)$ as:*

$$\mathbf{\mathcal{X}}_{\mathcal{O}(A)} = \{A(o) \mid o \in \mathcal{O}(A)\}$$

and $Val(X_i) = Val(A)$ for each $X_i \in \mathbf{\mathcal{X}}_{\mathcal{O}(A)}$.

For example, according to our RDF graph in Fig. 4.2, the object skeleton of the $Person(\alpha_1)$ can be defined as $\mathcal{O}(Person) = \{p_1, p_2, p_3\}$ and $\mathbf{\mathcal{X}}_{\mathcal{O}(Person)}$ is given as:

$$\mathbf{\mathcal{X}}_{\mathcal{O}(Person)} = \{Person(p_1), Person(p_2), Person(p_3)\}$$

Finally, we define *template factors* that define the probability distributions over a set of ground random variables generated from a set of attributes:

Definition 5 *Given a tuple of attributes $A_1, ..., A_l$ and object skeletons $\mathcal{O}(A_1), ..., \mathcal{O}(A_l)$, a* template factor *$f$ is a function from $Val(A_1) \times ... \times Val(A_l)$ to $\mathbb{R}$. Given a tuple of random variables $X_1, ..., X_l$ such that each $X_i \in \mathbf{\mathcal{X}}_{\mathcal{O}(A_i)}$ (i.e. $Val(X_i) = Val(A_i)$), $f(\mathbf{x})$ is called an instantiated factor and $f(\mathbf{x}) \geq 0$, $\forall \mathbf{x} \in Val(X_1) \times ... \times Val(X_l)$.*

A special type of template factor is *conditional probability distribution* (CPD) for Bayesian networks. CPD groups the tuple of attributes into two sets $\mathbf{A}_c, \mathbf{A}_{Pa} \subseteq \{A_1, ..., A_l\}$ as child and parent attributes, respectively. An important constraint for CPD is that their probability distribution is sum up to unity indicated as:

$$\sum_{\mathbf{x_c} \in Val(\mathbf{A}_c)} p(\mathbf{x}_c \mid \mathbf{x}_{Pa}) = 1, \ \forall \mathbf{x}_{Pa} \ \in \ Val(\mathbf{A}_{Pa})$$

for all possible assignments to the random variables $X_i \in \mathbf{\mathcal{X}}_{\mathcal{O}(A_i)}$.

4.2 Topical Relational Models

We employ a template-based construction for TRM. For this, we use the RDF structure as a template and first consider two types of observed data from an RDF graph, namely *class* and *relationship*. In this regard, every class represents a group of entities belonging to that class and is a good candidate for a template attribute since it can be instantiated to random (template)

variables for every entity of that class and its range is set to be binary-valued indicating whether that entity belongs to that class or not. In addition, for every relationship type, a dedicated template attribute is created to be instantiated for those entities having such relationship in-between in an RDF graph. This leads to the following definition:

Definition 6 *(Relationship and Class Attributes) Given a data graph $\mathcal{G}$, for every relationship $r \in R(\mathcal{G})$, a* relationship attribute *r is defined as a binary-valued function $r : V_E \times V_E \rightarrow \{0, 1\}$ where for each $e, e' \in V_E$, $r(e, e') = 1$ indicates that a relationship r exists between the entities e, e' and zero otherwise. Similarly, for every class $c \in C(\mathcal{G})$, a* class attribute *c is defined as a binary-valued function $c : V_E \rightarrow \{0, 1\}$ where for each $e \in V_E$, $c(e) = 1$ indicates that the entity e is an instance of class C and zero otherwise.*

This definition does not restrict the domain of both attributes which is initially set to all entities V_E available in the graph $\mathcal{G}$. Since the template variables of these attributes are observed, they should be restricted to a subset of entities (or entity pairs) for which the corresponding class (or relationship) is observed. Thus, we define an object skeleton for relationship attributes as:

$$\mathcal{O}(r) = \{< e, e' > | r \in R^{-}(e) \wedge r \in R^{+}(e')\}$$

and for class attributes as:

$$\mathcal{O}(c) = \{e | c \in C(e)\}$$

Example. According to RDF graph in Fig. 4.2, three class attributes (i.e. $Movie(\alpha_1)$, $Person(\alpha_1)$, $Location(\alpha_1)$) and four relationship attributes (i.e. $bornIn(\alpha_1, \alpha_2)$, $starring(\alpha_1, \alpha_2)$, $directedBy(\alpha_1, \alpha_2)$, $spouse(\alpha_1, \alpha_2)$) can be defined. Accordingly, the object skeleton for, e.g., the $Person$ attribute is $\mathcal{O}(Person) = \{p_1, p_2, p_3\}$, while the object skeleton for, e.g., $starring$ includes $\mathcal{O}(starring) = \{< m_1, p_2 >\}$. Given an attribute and its object skeleton, a set of template variables are defined which are the actual probabilistic elements in the ground Bayesian network. For instance, the template variables for the $starring$ attribute includes only $\mathbf{X}_{\mathcal{O}(starring)} = \{starring(m_1, p_2)\}$. A set of template variables created that way about the entity $p2$ is shown in Fig. 4.3-a (some of the object skeletons are omitted here for brevity). Note that all the template variables derived from an attribute share the same domain of assignments (i.e. $Val(A)$) which is set to $\{0, 1\}$ for class and relationship attributes.

Finally, we take the words of an entity into account and define a dedicated attribute to instantiate every observed entity-word pair as template variables in the model. This leads to the following definition:

Definition 7 *(Word Attribute) A word attribute w is a binary-valued attribute $w : V_E \times V \rightarrow \{0, 1\}$ that assigns one to every entity and one of its word tokens.*

This is a generic definition which we apply to every tokenized word in the attributes of the entities to generate a random variable for every word. Object skeleton for w includes all the possible entity-word pairs from the data graph:

$$\mathcal{O}(w) = \{< e, v > | \exists a.(e, r, a) \in \mathcal{E} \wedge v \in words(a)\}$$

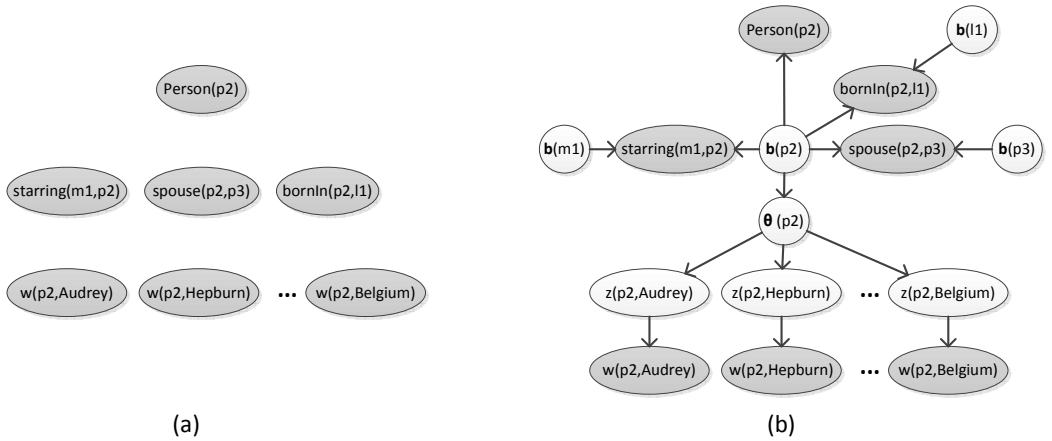


Figure 4.3: (a) The template variables initialized for the entity $p2$ (i.e. *Audrey Hepburn*) according to RDF graph in Fig. 4.2, (b) An illustration of the complete set of variables after introducing topical correlations around the entity $p2$ (the observed variables in gray)

where $words(a)$ is the set of tokenized words of the data value $a \in V_D$. Example template variables for the words of the entity $p2$ are shown at the bottom of Fig. 4.3-a.

4.2.1 Topical Correlations

Focusing on a particular entity, we consider that the observed variables from the data graph for an entity such as the ones derived from structural attributes (i.e. class and relationship) and word attributes are not completely independent, but probabilistically correlated. For example, the probability of observing the words "Brussels" or "Belgium" for the entity "p2" (i.e. "Audrey Hepburn") in Fig. 4.2 is supposed to be high given random variable of $bornIn(p2, l1)$ indicating "Audrey Hepburn is born in Belgium". However, representing such correlations directly in a PGM is very difficult, if not impossible, for two main reasons: First, the number of random variables is large resulting in a complex structure when we model a direct correlation among all of them. Additionally, such correlations partially exist – e.g. not every entity having a $bornIn$ relationship with $Belgium$ contains the word $Brussels$ in its attributes but some other word representing other cities of $Belgium$. In machine learning literature, this issue is dealt with the inclusion of hidden variables (i.e. random variables whose values are non-observed, but learned via inferencing) that can greatly simplify the structure of PGM, and reduce its complexity [122, p. 713].

In fact, topics are a good candidate to model such correlations between the structural attributes and word attributes. First, topics specify a low-dimensional representation of the textual data since in topic models the text is to be generated from a relatively small number of topics. We consider that textual data associated with the entities share the same set T of K latent topics denoted as $T = \{t_1, ..., t_K\}$. More importantly, the intuition of thematic clustering of the words within the topics is the key to represent the correlations among the structural and word variables. For example, assume for a moment a topic that assigns high probabilities to

the words relating to visual arts (e.g. movies, actress, director etc.). We normally expect that an entity related to such a topic has high probability of having a $starring$ relationship – e.g. $starring$ relationship between the entities $m1$ and $p2$ in Fig. 4.2.

We capture such associations between the topics and entities in a topic indicator attribute whose value is initially unobserved (hidden). The idea of including such hidden attributes in the probabilistic model plays a central role in our discussion of learning later in Sec. 4.2.3. For the moment, we note only that including topic-related attributes directly in the model allows us to make explicit the fact that all entities are associated to some topics and their words are sampled from a unique mixture of those topics as in the case of standard topic models.

Definition 8 *Given a set of topics $T = \{t_1, ..., t_K\}$, a topic indicator b of topic $t \in T$ is a binary-valued attribute $b_t : V_E \rightarrow \{0, 1\}$ such that for an entity e, $b_t(e) = 1$ indicates that the entity e has the topic t as its feature and zero otherwise. We denote the vector of all topic indicator variables of an entity as $\mathbf{b}(e) =< b_{t_1}(e), ..., b_{t_K}(e) >$.*

Topic indicator $\mathbf{b}$ acts as a placeholder in the context of an entity as a vector indicating which topics are present for that entity. Such a representation is useful as a random variable to capture further correlations between the topics and structural attributes of an entity. First we consider that the probability of observing an entity belonging to a class depends on its topic indicator vector $\mathbf{b}$. One way to model this is to define a template factor over $c(e)$ and $\mathbf{b}(e)$ by assuming that the probability of $c(e)$ can be written as a logistic sigmoid acting on a linear function of the topic indicator vector $\mathbf{b}(e)$ such that:

$$p(c(e) \mid \mathbf{b}(e)) = \sigma(\boldsymbol{\lambda}_c^T \mathbf{b}(e)) = \sigma\left(\sum_{k=1}^{K} b_{t_k}(e)\, \lambda_{ck}\right) \tag{4.1}$$

Here $\sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic sigmoid function that maps the values from $[-\infty, +\infty]$ to $[0, 1]$. $\boldsymbol{\lambda}$ is a global parameter shared among the entities which is represented as a $\mid V_C \mid \times K$ matrix. Each element λ_{ck} of $\boldsymbol{\lambda}$ represents the weight of correlation between the class c and topic t_k. Thus, the inner linear function of logistic sigmoid σ becomes the vectorial product of the class weights $\boldsymbol{\lambda_c}$ and topic indicator vector $\mathbf{b}(e)$. Here, logistic regression is a useful way of describing this correlation between one or more independent variables (e.g., topic indicator variables) and a binary response variable (e.g. class template variable), expressed as a probability, that has only two values, such as having the class or not.

Similarly, we model the probability of observing a relationship between two entities, e.g. $r(e_1, e_2)$, via logistic regression over their topic indicator vectors $\mathbf{b}(e_1)$ and $\mathbf{b}(e_2)$. A template factor over $r(e_1, e_2)$,$\mathbf{b}(e_1)$ and $\mathbf{b}(e_2)$ is defined as:

$$\begin{aligned} p(r(e_1, e_2) \mid \mathbf{b}(e_1), \mathbf{b}(e_2)) &= \sigma(\mathbf{b}(e_1)^T \boldsymbol{\omega}_r \mathbf{b}(e_2)) \\ &= \sigma\left(\sum_{k,l} b_{t_k}(e_1) b_{t_l}(e_2) \omega_{rkl}\right) \end{aligned} \tag{4.2}$$

Here the parameter $\boldsymbol{\omega}_r$ is a $K \times K$ matrix and every cell of this matrix indicates a weight

between two particular topics of that cell for a relationship r. In other words, for any given two entities such as the first one with a topic t_k and the second with a topic t_l, the weight of observing a relationship r between these two entities are given as the value of cell (k, l) of the matrix $\boldsymbol{\omega}_r$ denoted as ω_{rkl}.

Modeling both the classes and relationships of the entities by regressing over the topic indicator vectors has important implications. By doing so, we explicitly model the dependency between structural properties of the entity and its topical representation. Therefore, for those entities showing structural resemblances, we assume that their topics will also be similar. More importantly, the model parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ quantitatively capture the information of how much a topic is correlated to a particular structural property (e.g. class or relationships). More importantly, both template factors are only dependent on the topic indicator vectors of the corresponding entities. This observation leads to the following proposition:

Proposition 1 *Given a topic indicator vector of an entity, all of its template variables derived from class and relationship attributes are* conditionally independent.

Although a topic indicator vector is useful to determine which topics are present for an entity, it is not solely sufficient to train a topic model from which the actual words are generated. Following our discussion on topic models in Sec. 2.3.2, a topic proportion variable θ is introduced for this purpose to indicate the mixture of topics. In other words, topic proportions specify the distribution of the topics selected for the word attributes of an entity, while topic indicator vectors only specify the presence of those topics.

Definition 9 *A topic proportion θ is a vector-valued attribute $\theta : V_E \rightarrow \mathbb{R}^K_{\geq 0}$ such that, for an entity e, $\theta(e) =< \theta_1, ..., \theta_K >$ is per-entity topic distribution w.r.t. $\sum_{k=1}^{K} \theta_k = 1$. A topic-word assignment z is an attribute $z : V_E \times V \rightarrow T$ assigning each entity-word pair to a topic.*

However, unlike the standard topic models such as LDA [28], the topic proportions are only specified among the selected topics by the topic indicator vector of the entity, instead of a mixture of all topics. This introduces a sort of sparsity among the topics that the non-selected topics have no density in the topic proportions. This issue is dealt by parameterizing the distribution of topic proportions with the topic indicator vector. For this, we introduce a template factor $p(\boldsymbol{\theta}(e) \mid \mathbf{b}(e), \rho)$ between the topic indicator vector and topic proportions of the same entity e where ρ is the Dirichlet parameter that we discuss in the next subsection.

Definition 10 *Given a data graph $\mathcal{G}$, a* Topical Relational Model (TRM) *is defined as a tuple $\mathcal{M}_{TRM} = (\mathfrak{A}, \mathfrak{O}, \mathfrak{X}, \mathcal{P})$ where:*

- *For every class c and relationship r of $\mathcal{G}$, there exists attributes $c, r \in \mathfrak{A}$,*
- *$w, b_{t_1}, ..., b_{t_K}, \theta, z \in \mathfrak{A}$,*
- *$\mathfrak{O}$ is the set of object skeletons s.t., for every attribute $A \in \mathfrak{A}$, $\mathcal{O}(A) \in \mathfrak{O}$,*
- *For each attribute $A \in \mathfrak{A}$ and its object skeleton $\mathcal{O}(A) \in \mathfrak{O}$, there exists $\mathfrak{X}_{\mathcal{O}(A)} \in \mathfrak{X}$,*

- *P is the set of template factors.*

Theorem 1 *Any ground Bayesian network of TRM is valid (acyclic).*

Proof Sketch: It is easy to prove in two cases that any Bayesian network that conforms to our TRM construction is acyclic. First, because a local Bayesian network around an entity e is a directed tree in which the topic indicator vector $\mathbf{b}(e)$ is the root, it cannot contain a cycle. Consider the path $\mathbf{b}(e) \rightarrow \boldsymbol{\theta}(e) \rightarrow z(e,v) \rightarrow w(e,v)$ for any word v of the entity. Since v is the leaf node without any descendant, any element of the topic indicator vector $\mathbf{b}(e)$ is conditionally independent. Similarly, this can be also shown for the path $\mathbf{b}(e) \rightarrow c(e)$ of any class variable c of the entity. Second, since a relationship response variable between any two entities has no children, it cannot be a part of a directed cycle.

4.2.2 Generative Process

In the previous subsections we have defined a template-based construction of a TRM. In order to learn a TRM, our input contains a data graph that includes the relational structure and the underlying data of entities and attributes. Unfortunately, only some parts of such a model (entity words, concepts, and relationships) are directly observable from data while topic specific random variables need to be determined via inferencing. Thus our learning task involves to take the observed data of data graph $\mathcal{G}$ and use it to estimate the values of other parameters to obtain a posterior distribution. For this, we define the generative process here which specifies the joint distribution for TRM.

First we start with the template variables of topic indicator vector $\mathbf{b}$ that specifies a binary vector for each entity. We specify a *prior distribution* over the possible values in $\mathbf{b}$ in order to capture our initial uncertainty about the parameters. One such prior is the *Indian Buffet Process (IBP)* which is a non-parametric Bayesian process to generate latent features via a Beta-Bernoulli distributions [82, 85]. IBP assume that each entity possesses a topic k with probability π_k, and that the topic indicators are generated independently. Under this model, the probabilities of the topics are given as $\pi = \{\pi_1, ..., \pi_k\}$, and each π_k follows a beta distribution with hyperparameter α. Then each topic indicator value is sampled from a Bernoulli distribution as:

$$\begin{aligned} p(\pi_t \mid \alpha) &= Beta(\alpha/K, 1) \\ p(b_t(e) \mid \pi_t) &= Bernoulli(\pi_t) \end{aligned}$$

for any entity e. IBP is a non-parametric Bayesian process that can be utilized for both finite and infinite number of topics. For the purposes of this work, we set the number of topics to a fixed number of K, but this can easily be extended to infinite case as described in [67].

For topic proportions that we assign to each entity e, $\boldsymbol{\theta}(e)$, we set a Dirichlet distribution prior over the topics as in the case of other topic modeling approaches such as LDA [28]. However, instead of using uniform hyperparameters, we parametrize the Dirichlet distribution with topic indicator vector $\mathbf{b}(e)$ in order to limit the topic proportions to be distributed to only those topics selected by the topic indicator vector:

$$p(\boldsymbol{\theta}(e) \mid \mathbf{b}(e), \rho) \quad = \quad Dirichlet(\rho \mathbf{b}(e))$$

As the number of selected topics vary according to $\mathbf{b}(e)$, each entity will have a different density of topic proportions. However, according to the properties of Dirichlet distribution, the expectation of each topic proportion will only depend on the number of selected topics in $\mathbf{b}(e)$.

Proposition 2 *Given $f(\theta(e) \mid \mathbf{b}(e), \rho)$ as a $Dirichlet$ distribtion with parameters $\rho b_{t_1}(e), ..., \rho b_{t_K}(e)$, the expectation of $\theta_t(e)$ is $E[\theta_t(e)] = \frac{b_t(e)}{\sum_{t'} b_{t'}(e)}$.*

Based on these template factors, the following generative process is defined that leads to the joint probability distribution given in Appendix A.1 (The topic index z and word probabilities w are considered to be the same as LDA model):

1. For each topic $t = 1, 2, .., K$:
   (a) Draw topic probabilities $\pi_t \mid \alpha \sim Beta(\alpha/K, 1)$.
2. For each entity e:
   (a) For each topic t:
      i. Draw entity topic indicators $b_t(e) \mid \pi_t \sim Bernoulli(\pi_t)$
   (b) For each concept c:
      i. Draw $c(e)$ using Eq. 4.1
   (c) Draw topic proportions $\boldsymbol{\theta}(e) \mid \rho \sim Dir(\rho \mathbf{b}(e))$
   (d) For each word v of entity e:
      i. Draw topic index $z(e, v) \mid \boldsymbol{\theta}(e) \sim Mult(\boldsymbol{\theta}(e))$
      ii. Draw word $w(e, v) \mid z(e, v), \beta_{1:K} \sim Mult(\beta_{z(e,v)})$
3. For each pair of entities e_1, e_2:
   (a) For each relationship $r \in \{r \mid (e_1, r, e_2), (e_2, r, e_1) \in \mathcal{E}\}$:
      i. Draw $r(e_1, e_2)$ or $r(e_2, e_1)$ using Eq. 4.2

Figure 4.4 illustrates the graphical representation of the generative process for only two entities for brevity. In the next subsection, we "reverse" this process to apply the inference for discovering the distributions for latent variables.

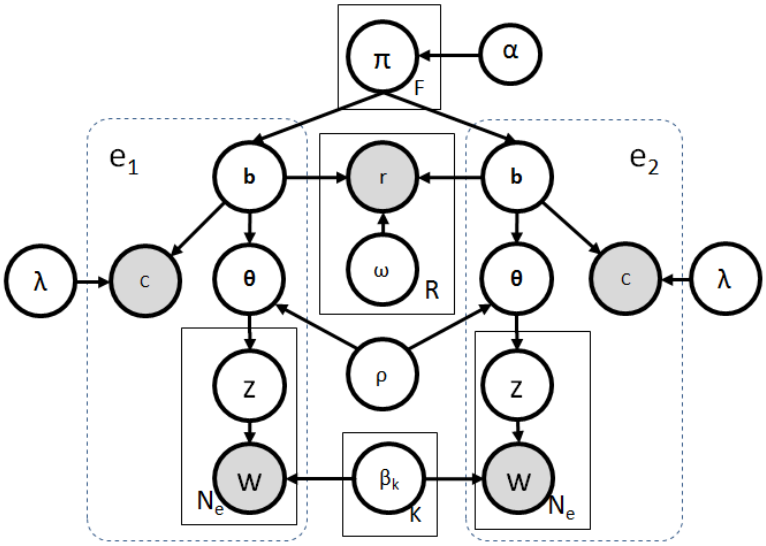


Figure 4.4: A graphical representation of two-entity segment of the generative process

4.2.3 Learning

We now move to the task of learning TRMs. To evaluate the merits of our approach, our goal is to learn the posterior distribution of the hidden topic-related variables $\mathbf{b}, \boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{Z}$ conditioned on the observed variables of data $\mathbf{w}, \mathbf{c}, \mathbf{r}$. However, using exact inference to learn such posterior distribution based on the joint distribution given in Appendix A.1 is intractable due to the hierarchy between the hidden topic-related variables. Therefore, we employ variational Bayesian learning to find an approximation of the posterior inference [112, 234]. Intuitively, the variational Bayesian method approximates the posterior distribution w.r.t. the distribution p by another distribution q which is restricted to belong to a family of distributions of simpler form than p. A simple way to obtain a tractable family of distribution is to use *mean field approximation* which assumes q to be a fully-factorized distribution over hidden variables indexed by some free variational parameters [122, Ch. 11]. Those parameters are then fit to be close to the true posterior, where closeness are measured by KL-divergence distance between p and q, $KL(q \parallel p)$. For TRM, we use the following fully-factorized distribution q, where the topic probabilities, indicators, proportions and assignments are considered independent:

$$q(\mathbf{b}, \boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{Z} \mid \boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{\tau}) = \prod_{t=1}^{K} \left[q(\pi_t \mid \tau_{t1}, \tau_{t2}) \prod_{e} q(b_t(e) \mid \nu_t(e)) \right] \prod_{e} \left[q(\boldsymbol{\theta}(e) \mid \boldsymbol{\gamma}(e)) \prod_{v, <e,v> \in \mathcal{O}(w)} q(z(e,v) \mid \boldsymbol{\phi}(e,v)) \right] \quad (4.3)$$

The parameters $\boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{\tau}$ are variational parameters for Bernoulli, Dirichlet, Multinomial, and Beta distributions, respectively. So borrowing the idea of energy functional from statistical physics, the following decomposition holds [152]:

$$\log p(\mathbf{c}, \mathbf{r}, \mathbf{w}) = \mathrm{KL}(q \parallel p) + \mathcal{L}(q) \tag{4.4}$$

where the term on the left hand side is called log marginal likelihood of observations and does not depend on q. Since $\mathrm{KL}(q \parallel p) \geq 0$, minimizing the KL-divergence is equivalent to maximizing the term $\mathcal{L}(q)$ by optimizing w.r.t. the distribution q. So when the KL-divergence vanishes (i.e. the distribution q equals to the true posterior p), then the maximum of the $\mathcal{L}(q)$ is reached. The term $\mathcal{L}(q)$ is called the variational lower bound on the log marginal likelihood and the learning can be represented as optimization problem of:

$$\{\tau, \nu, \gamma, \phi\} = \arg\max_{\{\tau,\nu,\gamma,\phi\}} \mathcal{L}(q) \tag{4.5}$$

where the full expression of $\mathcal{L}(q)$ is given in Appendix A.3. To optimize Eq. A.3 we use a *coordinate ascent* strategy on the variational parameters. The updates for variational parameters follow the standard recipe for variational inference with exponential family distributions in a conjugate setting except the update of the variational parameter ν [81]. We start with some choice of variational parameters $\{\tau, \nu, \gamma, \phi\}$ and iteratively update for all the entities until converge. Then for fixed values of the variational parameters, we maximize the lower bound with respect to the model parameters $\{\beta, \lambda, \omega\}$. This results in a *variational bayesian EM algorithm* with the following two steps:

1. (Variational E-Step) For all entities, find the optimizing values of the variational parameters $\{\tau, \nu, \gamma, \phi\}$. This is done as described in Algorithm 2.

2. (Variational M-Step) Maximize the resulting lower bound on the log marginal likelihood with respect to the model parameters $\{\beta, \lambda, \omega\}$. This corresponds to finding maximum likelihood estimates with expected sufficient statistics for each entity under the approximate posterior q which is computed in the variational E-step.

In particular, we show in Appendix A.4 that by computing the derivatives of the variational lower bound and setting them equal to zero, we obtain the following update equations. The variational parameters τ_{t1} and τ_{t1} are parameters of Beta distribution:

$$\tau_{t1} = \frac{\alpha}{K} + \sum_{e \in \mathcal{O}(\boldsymbol{b})} \nu_t(e) \tag{4.6}$$

$$\tau_{t2} = 1 + |\mathcal{O}(\boldsymbol{b})| - \sum_{e \in \mathcal{O}(\boldsymbol{b})} \nu_t(e) \tag{4.7}$$

The updates for variational multinomial ϕ_{wt} is identical to that in variational inference for LDA [28]:

$$\phi_t(e, v) \propto \exp\{\log \beta_{tv} + \Psi(\gamma_t(e)) - \Psi(\sum_{t'} \gamma_{t'}(e))\} \tag{4.8}$$

where $\Psi(.)$ is the digamma function (a digamma of a vector is vector of digammas) and $\phi_t(e, v) = q(z(e, v) = t)$. The variational Dirichlet parameters $\gamma_t(e)$ of topic proportions is:

$$\gamma_t(e) = \rho\nu_t(e) + \sum_{v,<e,v>\in\mathcal{O}(w)} \phi_t(e, v) \tag{4.9}$$

The contribution to the update in Eq. 4.9 includes the variational multinomial ϕ_{wt}, but also the variational parameter ν_t of the corresponding topic indicator b_t (i.e. $q(b_t(e) \mid \nu_t(e))$. This is the direct result of parameterizing the Dirichlet distribution with the topic indicator vector of each entity instead of using a non-informative prior α as in the LDA.

Finally, the update for the variational parameter $\nu_t(e)$ is given by:

$$\nu_t(e) = \frac{1}{1 + e^{-\vartheta}} \tag{4.10}$$

where ϑ is:

$$\vartheta_t(e) = \Psi(\tau_{t1}) - \Psi(\tau_{t2}) + \sum_{c\in C(e)} \vartheta_c + \sum_{r(e,e')} \vartheta_{r1} + \sum_{r(e',e)} \vartheta_{r2} + \vartheta_\gamma \tag{4.11}$$

The update in Eq. 4.11 has four different parts. The contribution from the Beta prior can be computed by $\Psi(\tau_{t1}) - \Psi(\tau_{t2})$. For each concept $c \in C(e)$ the contribution to the update is given by:

$$\vartheta_c = (1 - \sigma(\boldsymbol{\lambda}_c^T \boldsymbol{\nu}(e)))\lambda_{ct} \tag{4.12}$$

We make a distinction between the contribution from the relationships of the entity. If the entity is the source of a relationship $r(e, e')$, the contribution to the update in Eq. 4.11 is computed by:

$$\vartheta_{r1} = (1 - \sigma(\boldsymbol{\nu}(e)^T \boldsymbol{\omega}_r \boldsymbol{\nu}(e')))\omega_{rt.}\boldsymbol{\nu}(e') \tag{4.13}$$

or if the entity is the target of the relationship $r(e', e)$, the contribution of the relationship is given by:

$$\vartheta_{r2} = (1 - \sigma(\boldsymbol{\nu}(e')^T \boldsymbol{\omega}_r \boldsymbol{\nu}(e)))\omega_{r.t}\boldsymbol{\nu}(e') \tag{4.14}$$

and ϑ_γ is given by:

$$\rho\,(\psi(\gamma_t(e)) - \psi(\sum_{t'} \gamma_{t'}(e))) \tag{4.15}$$

In M-Step, we maximize the variational lower bound w.r.t. the model parameters β, λ, ω. This is equivalent to maximum likelihood estimation with expected sufficient statistics where expectation is taken w.r.t. the variational distribution q in Eq. A.2. The update for the topic parameter β is given by:

$$\beta_{tv} \propto \sum_e \sum_{v',<e,v'>\in\mathcal{O}(w)} \delta(v' = v)\phi_t(e, v) \tag{4.16}$$

This update is the same as the β update for LDA since the words of the entities are only conditionally dependent on topic parameter β and topic-word assignment z as in other topic

models.

In order to fit the parameters λ and ω of the logistic regression of Eq. 4.1 and 4.2 respectively, we employ gradient-based optimization. At each iteration we update these parameters using the following gradients:

$$\bigtriangledown_{\lambda_{ct}} = \sum_{e}(1 - \sigma(\boldsymbol{\lambda}_c^T \boldsymbol{\nu}(e)))\nu_t(e) \tag{4.17}$$

and

$$\bigtriangledown_{\omega_{rtt'}} = \sum_{e,e'}(1 - \sigma(\boldsymbol{\nu}(e)^T \boldsymbol{\omega}_r \boldsymbol{\nu}(e')))\omega_{rtt'}\nu_{t'}(e') \tag{4.18}$$

Note that these gradients cannot be used to directly optimize the parameters since they are only calculated for the positive observations of class and relationships. As negative observations (i.e. an entity is not a member of a class) is not explicitly indicated in RDF graph, we also apply a regularization penalty to along with parameter update procedures as described in Appendix A.5.

Algorithm 1: Variational Bayesian for TRM

```
Input: 𝒢 // Data graph
Input: β, λ, ω // initial set of parameters
Input: α, ρ // hyperparameters
  while (not converge) do
    // variational E-step
    ν, φ ← Compute-EStep(𝒢, β, λ, ω, α, ρ)
    // variational M-step
    for each topic t do
      Update β_t using Eq. 4.16
    end for
    for each class c do
      for each topic t do
        Update λ_ct using the gradient in Eq. 4.17
      end for
    end for
    for each relationship r do
      for each topics t,t' do
        Update ω_rtt' using the gradient in Eq. 4.18
      end for
    end for
  end while
Output: β, λ, ω
```

Algorithm 2: Compute-EStep procedure

Input: $\mathcal{G}$ // Data graph
Input: $\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\omega}$ // initial set of parameters
Input: α, ρ // hyperparameters
// initialize variational parameters
$initialize(\boldsymbol{\tau}, \boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\phi})$
while not converge **do**
  for each topic t **do**
    $\tau_{t1}, \tau_{t2} \leftarrow$ update using Eq.4.6 and Eq.4.7
  end for
  for each entity e **do**
    for each word w **do**
      $\boldsymbol{\phi}(e, v) \leftarrow$ update using Eq.4.8 for all t
    end for
    $\boldsymbol{\gamma}(e) \leftarrow$ update using Eq.4.9 for all t
    $\boldsymbol{\nu}(e) \leftarrow$ update using Eq.4.10 for all t
  end for
end while
Output: $\boldsymbol{\nu}, \boldsymbol{\phi}$

4.3 Experiments

4.3.1 Dataset

In order to evaluate the effectiveness of the proposed model, we used a subset of DBpedia[1] as dataset to train and test TRM. For training, we extracted 20094 entities from DBPedia mostly in the categories of People, Locations, Organizations, Visual Arts and Education. The resulting dataset includes 112 distinct concepts and 46 distinct relationships among the entities. The size of the vocabulary is 26109 unique words after highly frequent words are removed. The use of DBPedia mainly provides a heterogeneous data graph: Most of the entities belong to more than one class along with rich textual description. Text of each entity is represented by a bag of words that appeared in its attributes. All stop-words and highly frequent terms are removed and stemming is applied before training. In addition, we sample another dataset for testing which includes 4788 entities from 37 distinct concepts and 18 distinct relationships in order to evaluate the model. It should be noted that all the concepts and relationships in the test dataset are required to be a subset of the concepts and relationships in the training dataset in order to re-use the same model parameters learned from the latter.

4.3.2 Parameter setting

In the training phase, we set convergence criteria to terminate the iteration in Alg. 1 if the fractional decrease in the lower bound of the log-likelihood of the entire observed data in two

[1] http://dbpedia.org

successive iterations is less than 10^{-4}, or if the number of iterations exceeds than 100. This results in the variational EM to iterate until the convergence criteria is met. For the inner iteration of E-step in Alg. 2 which involves updating the variational parameters, the convergence criteria are when the fractional decrease in the lower bound of the log-likelihood of the data is less than 10^{-5} in two successive iterations, or when the number of iterations exceeds 20.

4.3.3 Topic analysis

A useful application of our approach is to understand the underlying nature of data. The topical correlations between the topic wise multinomial distributions for each type of entity induced by TRM provide an overview of this nature. We compare our approach with the topics learned via an unsupervised method of LDA. For this we train a LDA model using the experiment dataset regarding every entity as a document. Table 4.1 shows the top words of five selected LDA topics induced by running over the dataset. The model utilizes Mallet implementation of the Gibbs sampling procedure which was run until convergence (around 85 iterations) and the number of topics was set to 100.

For the visualization of TRM, we ran the model on the dataset and selected some representative topics from the output around some relationships and concepts. Figure 4.5 displays the top words of four selected topics using the learned $\boldsymbol{\beta}$ parameter. Words are ranked by the following formula:

$$score(v,t) = \beta_{tv}\left(\log\beta_{tv} - \frac{1}{K}\sum_{t'}\log\beta_{t'v}\right)$$

which finds words that are likely to be characteristic in a topic but also not so frequent in other topics. The figure also shows the topical correlations between the topics and classes and relationships from the data graph. The relationships between any two topics that give a high weight in its learned ω parameter is represented as an edge between those topics together with the unnormalized weight assigned. In addition, each class that assigns a high weight in its learned λ parameter to particular topics is also shown.

As observed in the figure, the structure of the data graph is highly influential while learning topic distributions since the topics capture the co-occurrence statistics only among the entities sharing similar data structure. This is in contrast to the topic distributions of the LDA model which is only based on the co-occurrence of words without considering an external structure. In fact, TRM enables a bias towards the correlation for the entities with similar structure to obtain similar topics and only capture co-occurrence statistics in that direction. For example, topic 1 on the top-left of Figure 4.5 has top words from the entities of type $Organization$ or $Company$ whereas topic 2 on the top-right is correlated more to the $Person$ class. As those topics are highly frequent among the entities of those classes, the assigned λ weights (shown in parentheses) of those classes to these topics are higher than the others. Another interesting property of TRM is to model those correlations as the mixture of topics instead of assigning one topic per class or relationship. For example, the class $Company$ is correlated to three different topics (e.g. topic 1, 2, and 4) in the figure which results from the fact that the entities of this class are highly related to those three topics.

The topics also reflect the topical correlations between the text and the relationships. A

Topic 1	film director born screenwrit produc actor american direct ndash writer decemb octob septemb januari novemb august juli june
Topic 2	british english film star taylor michael william john adapt elizabeth richardson powel plai richard jame redgrav henri david mari
Topic 3	swedish sweden bergman finnish stockholm finland ingmar municip afghanistan ingrid sj inhabit malm counti berg afghan moodysson ping liv
Topic 4	model magazin world fashion miss playboi commerci time beauti cover page coca cola photograph peopl food campaign celebr lopez
Topic 5	citi popul area largest capit locat river region metropolitan km center municip world provinc north port urban south million

Table 4.1: Five topics and their top-20 words trained by the LDA model on the experimental data.

directed edge from a source topic to a target topic indicates that it is very likely to observe a relationship between a source entity and target entity related to those topics respectively. For example, the $keyPerson$ relationship assigns a high weight in its ω parameter to the topic 1 as source and topic 2 as target. This indicates that $keyPerson$ relationship is highly correlated between the $Company$-related topic and $Employee$-related topic meaning that this relationship is highly frequent between the entities of these two types. Similar to the classes, different weights are also assigned to different pairs of topics by the relationships. For example, $successor$ relationship is highly common among the $people$ but also among the $companies$ which assigns positive weights (3.0 and 1.1) between the corresponding topics. Also note that a self-loop for $successor$ relationship in the figure indicates that both the source and target topics are the same. This indicates a weight to be in the diagonal of ω matrix of that relationship.

It is clear from topic analysis that the proposed TRM exploits both the structure and text within the data to learn more semantically coherent topics as well as a relational model to reveal the topical correlations between structure and text. This highly differs from the LDA topics shown in Table 4.1 which are solely based on word co-occurrence learned in an unsupervised fashion.

4.3.4 Log-likelihood of new data

In this task, we measure how well the TRM predict unseen data in terms of log-likelihood. The higher log-likelihood the model assigns to unseen data, the better is its predictive power and generalzability. Our experimental set-up is as follows. We first train TRM parameters using the entire set of training data. Using these learned model parameters (with the same number of topics), we perform inference on the test dataset and compute the variational lower-bound on the cumulative log-likelihood of words in the test dataset.

Figure 4.6 compares the performance of the LDA and TRM models on log-likelihood evaluation on the test dataset. It is clear that TRM is significantly better than the LDA. This results from the fact that the use of structure information of the new entities in the test dataset is helping to predict their topics and then to decide topical correlations between the text and the structure during inferencing. It is also observed that LDA slightly improves as more topics are added to the model since it is able to cluster the words in different topics. The number of topics

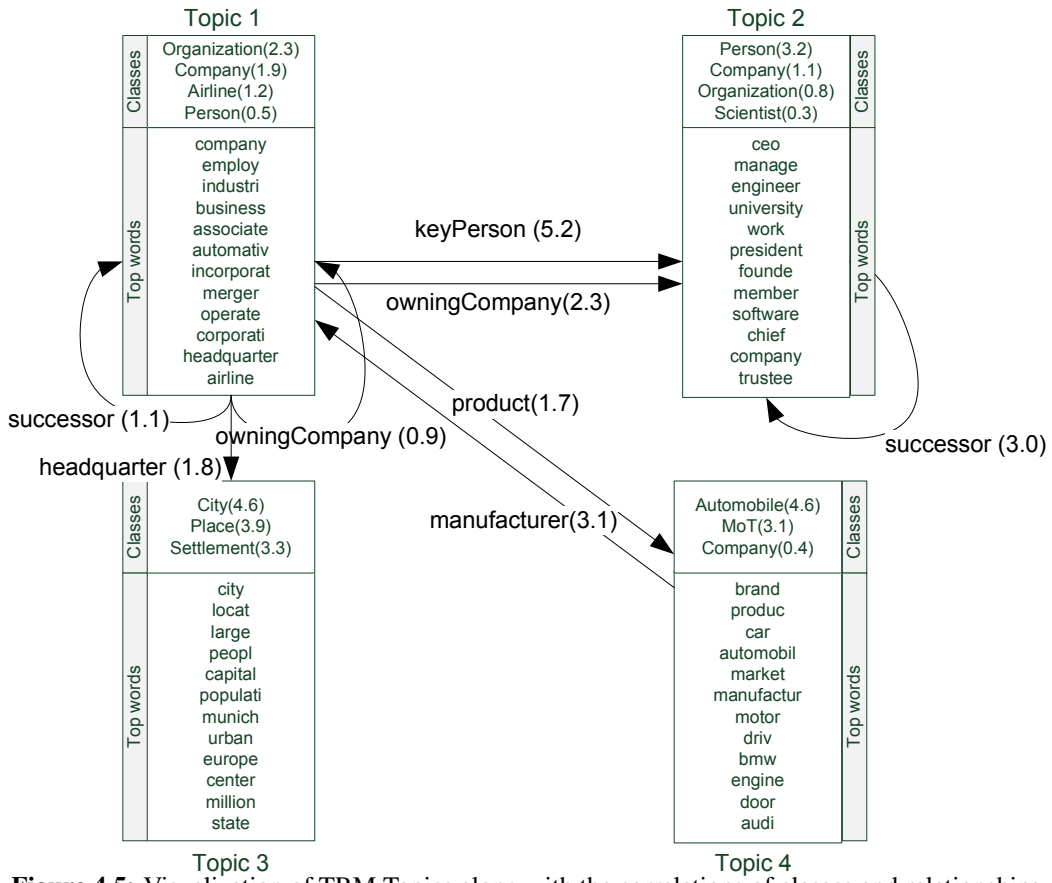


Figure 4.5: Visualization of TRM Topics along with the correlations of classes and relationships

in TRM is also helping to improve the log-likelihood to some extent but remains static after some point. This is mostly due to the fact that the topics are highly determined based on the structure and additional topics do not change topic assignments if there are sufficient number of topics to capture topic distributions.

4.4 Discussion

The work presented in this chapter is related to the fields of statistical relational learning (SRL) and topic models as introduced in Sec. 2.3.3.2 and 2.3.2 respectively. Early models of SRL such as probabilistic relational models (PRM) [75, 79] or Markov Logic Networks (MLN) [193] train a template-based probabilistic graphical models (based on Bayesian and Markov networks respectively) using the data structure as a template. These models however assume a flexibility on the structure within the PGM that either requires templates to be pre-given or learned via costly structure learning techniques. The use of latent variables in SRL is first proposed by [118] in Infinite Relational Models which clusters the entities into different latent classes and applies a nonparametric Bayesian inference to learn those classes. This is further extended by Xu et al. [248] that utilizes a Dirichlet process mixture model to propagate infor-

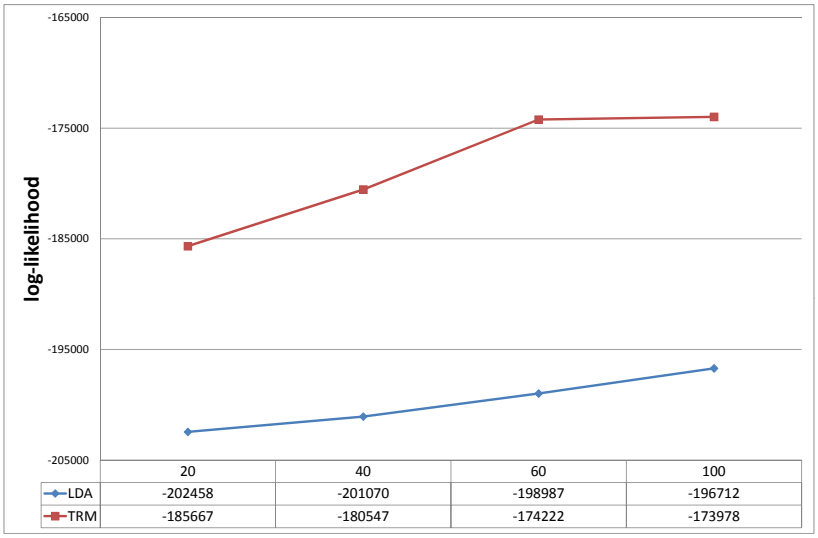


Figure 4.6: Likelihood performance of TRM and LDA models on the test dataset.

mation in the network of latent variables and to reduce the necessity for extensive structural learning. Our work follows this line of approaches, but significantly differs from these models in various points. Mainly we utilize a latent feature model of topic indicators to represent every entity as a feature vector instead of assigning it to latent classes of previous approaches. More importantly, previous approaches of SRL do not consider the text within the relational data. By capturing the topical correlations between the text and structure of the data via topic models we are able to train a topical relational model which is ideal for textually-rich relational data.

In addition, many variations of topic models have been proposed in the literature and shown to be useful for data analysis. Basically there are two principal approaches in topic models, namely PLSA [95] and LDA [28], which exploit co-occurrence patterns of words in documents to reveal semantically meaningful probabilistic topics. Recently, several studies are proposed to apply topic modeling to the network structures (e.g. social networks) such as NetPLSA [160], Pairwise-Link-LDA [171], Nubbi [42] and author-topic models [199]. The major distinction between these models and TRM is the fact that these models consider combining topic modeling with homogeneous networks, such as citation graphs and social network graphs, which only consider one or two types of entities connected with unlabeled relationships. Instead a data graph in RDF is a heterogeneous information network which is more complex than those homogeneous networks with unique characteristics of modeling class memberships and different relationship types. A more recent work to consider heterogeneous networks is proposed in [61] that applies a regularization framework to bias the topics according to the network structure. However, the proposed approach is still not generic enough to be extended to the RDF

setting since the framework requires new regularization parameters to be specified for every class present in the network and results in a different model when a new class is introduced in the network. In fact, the work in [61] is presented for a network of three classes (e.g. paper, venue and author). By directly incorporating the structure information to the same TPGM we are able to learn biased topics via topical correlations and provides a more generic approach as required by RDF-based, textually-rich relational data.

4.5 Conclusion

In this chapter we presented TRM as a novel model to learn from RDF datasets in order to extract topical correlations between structure and text. TRM exploits a template-based construction from RDF structure and learns a full probabilistic graphical model that captures the correlations with learned parameters. In the next chapter we apply the results of TRM to develop a new ranking model for document retrieval.

5

Semantic Relevance Models for Document Retrieval

In this chapter, we mainly study how semantic Web data can be used to infer users' information need and retrieve and rank documents in a standard IR setting. We propose an adoption of the relevance model (see Sec. 2.2.2.1) that instead of using pseudo relevance feedback documents, we employ semantic data to construct a query model. *Inferring the information need* from an often very short query is an important problem in Information Retrieval (IR). Various attempts to improve the query model have shown to be effective in numerous settings. Generally speaking, the common goal is to reach a better approximation of the need so that more information can be employed for retrieving and ranking documents. To this end, different avenues have been explored, including the use of *clusters* (each containing similar documents that are assumed to represent the same need) [146], *pseudo-relevance feedback (PRF) documents* [38, 133, 137, 151, 246] and *semantic resources* such as concepts and entities associated with documents [226]. A statistically disciplined and effective approach is for instance the Relevance Model (RM) [133], which predicts an enhanced query model from top-ranked PRF documents. However, the assumption that the top-ranked cluster or documents obtained from the initial query correctly represent the need is often invalid [38], resulting in cases where pseudo-relevance feedbacks degrade retrieval performance regardless of the retrieval method being used [222, 235]. Also the use of semantic resources is problematic in practice because it depends on the underlying machinery to extract semantic concepts from documents to provide high quality annotations. So far, it has been shown that searching with a conceptual representation hurts performance in the case of short documents [226] and there are no clear evidence of improvement in other cases [162].

However, improvement is possible with the use of *external resources* that are not assumed to be directly associated with documents. For instance, Wikipedia has been successfully exploited as a source for PRF [246]. Meanwhile, as discussed in Sec. 3.3.2, when the volume of data on the Web becomes much larger, we also consider this as a valuable source, which can potentially help to infer the information need. In particular, the amount of *semantic data* available as ontologies, RDF datasets, and Linked Data is increasing rapidly. Collectively, the amount of Linked Data on the Web alone comprises of hundreds of datasets (domain-specific

and domain-independent ones such as Freebase and DBPedia, the semantic data counterpart to Wikipedia) containing billions of RDF triples [24]. Every RDF triple is basically a statement, which describes an *entity* in terms of its *class*, *attribute* value or *relationship* to one another. Publicly available semantic data cover many topics from different domains, and might reasonably be assumed to reflect the diverse interests and information needs of users. In fact, a recent study of the Yahoo! Web query log has shown that over 70% of all queries contain a semantic resource (entity, type, relation, or attribute) [183].

To the best of our knowledge, only semantic resources that are associated with or embedded in documents have been used so far. There are however no studies investigating whether the large amount of *external semantic resources* on the Web that have no direct connections with documents, can help to improve IR tasks.

In this chapter, we introduce a framework for dealing with external semantic data for the task of document retrieval. While this framework opens various new ways to exploit external semantic data (e.g. for document modeling), we focus our attention on understanding the information needs (i.e. query modeling). For this, we propose an adoption of RM to arrive at a *semantic relevance model* (SRM) that helps to infer the needs in terms of available semantic entities. Although RM is mainly used as a way to expand queries, it can also be understood as a generic framework that allows relevance to be captured by a hidden representation space as previously discussed in Sec. 2.2.2.1 and 3.2.1. So far, this hidden representation space is assumed to be captured by documents. In this work, we employ a richer representation captured by entities of an external semantic dataset instead. For this, we construct SRM from these semantic data by selecting entities for a given query. While PRF is also used here, entities matching the queries are obtained, instead of documents in the collection, i.e. we obtain a richer representation space by using the query to sample from semantic data (semantic relevance sampling). Additionally, instead of directly using the selected entities by the resulting SRM for query expansion, we employ the Topical Relational Model (TRM) as introduced in Chapter 4 to infer the relevance as mixtures of topics. TRM provides more smoothed probabilities of words around an entity and also allows to utilize the structural information around the entity such as the classes or relationships involved. Using TRM as a generative model, we are able to rank more semantically coherent documents to the actual information need of the user.

The contributions of this work are as follows: (1) we study the use of external semantic data as a source for inferring the information need. (2) Instead of using PRF documents, we propose methods to sample from semantic data to estimate a more fine-grained relevance model based on entities. In the experiments using TREC collections, we show that this SRM can improve the baseline RM because focusing on terms specifically generated from relevant entity topics helps to avoid the negative effect of non-relevant terms that can appear in the more coarse-grained PRF documents. (3) Previous approaches map the query to documents and exploit their associated semantic resources for inferring the needs. By mapping the query directly to semantic resources and using SMTs estimated for them to provide an approximation of the need, we do not rely on high-quality document annotations. This is crucial for large-scale ad-hoc retrieval scenarios where such annotations cannot be assumed to be always available. In particular, the use of semantic resources is not limited to the ones directly associated with documents but naturally applies to any dataset comprising of entities and relations. Thus, this

work paves new ways for exploiting the abundant availability of relational databases and data warehouses in enterprises and data on the Web in order to improve document retrieval. (4) We also study how the extracted topical correlations in TRM can be exploited to obtain better term weighting for a specific entity. For this we propose a supervised weighting scheme to determine the importance of structural information around the entity (classes or relationships) directly on document retrieval performance.

The remainder of this chapter is organized as follows: Sec. 5.1 further elaborates preliminaries for the work presented in this chapter. Sec. 5.2 introduces the SRM framework and discusses the specific techniques to the abovementioned contributions. Experimental results are reported in Sec. 5.3, and we conclude in Sec. 5.4.

5.1 Problem Definition and Preliminaries

5.1.1 Information Need and Relevance Models

Various attempts have been made to infer the information needs. The *cluster hypothesis* has been studied, which postulates that similar documents match similar information needs and finding the right cluster and using its documents as a mean to approximate the information need can improve retrieval effectiveness [91, 146]. Clearly, how the right cluster(s) can be automatically identified is the core problem. Clustering has been performed statically over the entire collection or dynamically on documents obtained for the given query [91, 223]. Studies confirmed that the hypothesis is especially valid for query-specific approaches, showing that using an optimal cluster can help to improve retrieval performance [91]. A recent approach based on the language modeling framework, which performs consistently across large collections, is based on smoothing the document models using models of the clusters they come from [146].

A different direction that involves direct manipulation of the information need representation is *query expansion* [38, 133, 137, 151, 246] in that terms associated with relevant clusters are used in addition to query terms. Query expansion is mainly done based on top-ranked PRF documents. The way RM [133] has been implemented corresponds to this idea because PRF documents are used to infer relevance and to compute the query model. However, it is a rather massive expansion strategy, which actually replaces the query with a distribution over the entire vocabulary. As a generic framework, RM is based on the generative relevance hypothesis that assumes for a given information need, queries and documents relevant to that need can be viewed as random samples from the same hidden model of the information need (i.e. the RM). Thus, for this approach to be effective, the main factor becomes how accurately we can approximate this RM. Formally, given r is a sample of the hidden information need representation, the RM is defined as:

$$RM_{\mathbf{r}}(v) = P(v \mid r_1, .., r_m) = \frac{P(v, r_1, .., r_m)}{P(r_1, .., r_m)}$$

Here, $v \in V$ is a word from the vocabulary V and $r = \{r_1, .., r_m\}$ is the decomposition of the information need representation. In the implementation of this model, query terms in Q

are used as an approximation of this sample r and PRF documents obtained using Q, serve as artifacts of the hidden information need representation. In particular, using a set F of PRF documents and assuming that query terms $q_i \in Q$ are exchangeable, Lavrenko and Croft proposed independent and identically distributed (i.i.d.) sampling to obtain the RM as follows:

$$RM_F(v) \approx P(v \mid Q) = \sum_{D \in F} P(D)P(v \mid D) \prod_{q_i \in Q} P(q_i \mid D)$$

Given a collection of documents C and the vocabulary of terms V, the score of a document $D \in C$ is based on its cross-entropy from the relevance model RM_F:

$$-H(RM_F \parallel D) = \sum_{v \in V} RM_F(v) \log P(v \mid D)$$

where $P(v \mid D)$ is defined as:

$$P(v \mid D) = \lambda_D \frac{n(v, D)}{\mid d \mid} + (1 - \lambda_D)P(v \mid C)$$

Here, $n(v, D)$ is the count of the word v in the document, $\mid d \mid$ is the document length, and $P(v \mid C)$ is the background probability of v.

Clearly, this approximation of the RM relies on the assumption that document models, either obtained from clusters or via PRF, correctly capture the information needs and thus, are used as the main artifacts for sampling. However, documents are rather coarse-grained models that might contain terms not related to the needs. There are cases showing that PRF documents actually degrade retrieval performance regardless of the retrieval method being used [222, 235]. Recent work focused on improving this approximation by using techniques such as cluster-based resampling [137], learning-to-rank relevant documents [188], large external corpora [63], or by exploiting term positions [151].

Besides documents, there are also approaches, which consider statistics of the entire collection, *semantic resources* such as concepts and entities [162, 198, 214, 226] and *external sources* such as Wikipedia [246]. Using semantic resources is mainly based on expanding the queries with concepts associated with PRF documents [162], or by replacing the entire query with a conceptual representation [226]. Either way, there is a need for knowledge extraction techniques that help to extract concepts and entities from texts. As opposed to this, recent approaches making use of Wikipedia do not require an extraction mechanism to bridge the semantic gap, but directly map queries to external documents, i.e. Wikipedia entity pages [246], which serve as textual representations of entities to be exploited for query expansion. Although the use of Wikipedia entity pages to estimate the information need is more focused and works better than some of the document-based PRF approaches, there are still problems: first, entity pages are diverse, often describing different topics in different sections. Many terms in the page may not be directly related to the entity. Besides, Wikipedia is still relatively small in the sense that its entity pages may not cover all possible information needs and contents of the document collection. Due to these, in [246], it is reported that this approach only works well for some types of queries (e.g. entity and ambiguous queries).

Our work leverages the cluster hypothesis (and RM) in the sense that the information need is inferred by finding the right entities and relations, which correspond to the notion of cluster (samples of the RM). By interpreting the needs in terms of entities and relations, it aims to leverage the increased availability of external semantic resources. However, it does not rely on documents to be annotated with these resources but is rather similar to the idea of considering documents as representations of entities [246]. Instead of using external documents (e.g. entity pages in Wikipedia), we employ PRF to derive virtual documents from the collection, which may correspond to entities, entity types, entity attributes and relation between entities. Further, as opposed to retrieving entity pages and using them directly for query expansion [246], our approach retrieves virtual semantic documents, and employs an extension of RM to infer a query model from this semantic data sample. This model is in fact a mixture of topics that are related to the semantic resources in this sample.

5.1.2 Topic Models for IR

Over the last two decades, different latent topic models have been proposed to address the *vocabulary mismatch problem* of IR that results from the discrepancy between the documents and query keywords [28, 95]. A topic in this sense is a group of different words that occur in a similar context. Thus, a query and a document represented in such a lower-dimensional space can still have high similarity even if they do not share a word. In [238], Wei and Croft successfully showed that using LDA [28] to model a document as a mixture of topics performs better than the one topic (cluster) assumption. LDA is a state-of-the-art unsupervised learning technique for extracting topics from documents. Basically, in this retrieval model, a query $\mathbf{q} = q_1, ..q_{|q|}$, is generated from a document $\mathbf{d}$ using the following process: A multinomial distribution $\boldsymbol{\theta}^d$ over topics is selected for $\boldsymbol{d}$. Then, a latent topic z is selected with probability $P(z \mid \boldsymbol{\theta}^d) = \theta_z^d$. Finally, each $q \in \mathbf{q}$ is generated with probability $P(q \mid \boldsymbol{\beta}, z)$, i.e. the probability of word q being observed through repeated sampling from words in topic z. Using this generative model, the probability of generating the query $\mathbf{q}$ from the document $\mathbf{d}$ is defined as:

$$P(\mathbf{q} \mid \mathbf{d}) = \prod_{q \in \mathbf{q}} \sum_{z} P(q \mid \boldsymbol{\beta}, z) P(z \mid \boldsymbol{\theta}^d)$$

LDA-based topic model actually differs from its predecessor model Probabilistic Latent Semantic Analysis (PLSA) in that it places Dirichlet priors over the parameters of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. Experiments showed that this helps to overcome the problem of overfitting.

Instead of learning topics and the model parameters in a completely unsupervised manner, recent work uses predefined topics. Ramage et al. assume a one-to-one correspondence between LDA topics and user tags and focus on learning word-tag distributions [190]. Also ontology concepts have been used as topics [44]. To establish the connection between these predefined topics and the collection, documents were assumed to be annotated with a set of topics (tags) [190], or topics (concepts) to be associated with a set of words [44].

STM follows this line of work by assuming the correspondence of topics to some semantic resources. However, this correspondence is not one-to-one such that concepts, entities, and

relations might be associated with a mixture of topics. In fact, STM is more similar to the original LDA in that latent topics are used. Thus as opposed to previous work [44, 190], no explicit connection between topics and the collection has to be assumed. The difference to LDA is that topics model virtual documents corresponding to semantic resources.

5.2 Semantic Relevance Model

Following Lavrenko's generative theory of relevance, SRM assumes that the query and documents are samples from a hidden representation space where the actual relevance is modeled. However, while artifacts of this space correspond to collection documents in the original model, we interpret them as referring to entities. As discussed, highly critical to the quality of any RM is the choice of the representation of the relevance in the hidden space and also the way of how the queries and documents are generated from that representation. Instead of using the set of PRF documents F for it, we replace them by entities in order to create a more focused and condensed representation of relevance. Intuitively speaking, we aim to construct a model, which captures relevance in terms of semantic resources, but at the same time, has the generative power to output terms representing relevant queries and documents. Since the initial representation of relevance to user's information need is unknown, we can make an estimation from a given query q. Analogous to RM, the SRM based on a PRF set $F = \{e_1, ..., e_k\}$ of entities selected for query q can be defined as:

$$SRM_F(v) \quad = \quad \sum_{e \in F} P(e)P(v \mid e)P(e \mid q) \tag{5.1}$$

where the prior probability $P(e)$ is assumed to be uniform and can be ignored. While the last term on the right hand side weights each entity e in the PRF set w.r.t. the query terms, the middle term $P(v \mid e)$ assigns a probability to any word v based on the entity e. Based on this, the score of a document d is calculated using negative cross-entropy between the SRM and document model as:

$$S_{SRM}(q, d) = \sum_{v \in V} SRM_F(v) \log P(v \mid d) \tag{5.2}$$

where $P(v \mid D)$ is defined as:

$$P(v \mid D) = \lambda_D \frac{n(v, D)}{\mid d \mid} + (1 - \lambda_D)P(v \mid C)$$

Here, $n(v, D)$ is the count of the word v in the document, $\mid d \mid$ is the document length, and $P(v \mid C)$ is the background probability of v.

The performance of SRM is highly determined by two principal factors: (1) the entities in PRF set F considered to be relevant and (2) the generative model used for these entities to assign probabilities to any word v in $P(v \mid e)$. We now discuss how the given query is used to

obtain the PRF set F, and how to estimate the probability $P(v \mid e)$ in detail.

5.2.1 Selecting Relevant Entities

A variety of techniques can be used for selecting entities relevant to a given query q. For the purposes of this work, we employ all attributes A_e associated with an entity such as $name$, $comment$, etc., which can be found in the data graph. Attribute texts are treated as a smoothed model θ_e of a set of terms, which together, form yet a 'virtual document" representation of every semantic resource. For utilizing these sources, we employ the standard language modeling framework and the query-likelihood model [179]. That is, top-k entities are selected based on language models $P_{LM}(v \mid \theta_e)$ built from their attribute text, and included in the resulting PRF set F in order to represent the relevance for the query. The interpolation with corpus-specific probability is controlled by linear interpolation using the parameter λ_e such that the ranking of entities based on query likelihood is defined as

$$P(q \mid e) = \lambda_e \prod_{v \in q} P_{LM}(v \mid \theta_e) + (1 - \lambda_e) \prod_{v \in q} P(v \mid C) \tag{5.3}$$

In order to easily retrieve the entities, a special inverted index is created which regards every entity as a document and terms are extracted from the smoothed model of attributes. Based on this, $P(e \mid q)$ in Eq. 5.1 can be represented as a distribution over the sample space limited by query q:

$$P(e \mid q) = \frac{P(q \mid e)}{\sum_{e' \in F} P(q \mid e')} \tag{5.4}$$

That is, the weight $P(e \mid q)$ is the normalized query-likelihood scores obtained in the initial retrieval phase. Therefore, we can say that the relevance model represents a group of the top-k entities combining the language models by the arithmetic mean weighted by the initial search results over entities.

5.2.2 Term weighting using TRM

Following the selection of relevant entities, we need to estimate the core probability $P(v \mid e)$ in our SRM model in Eq. 5.1 in order to assign accurate term weights for selected entities. Term weighting is a classical IR problem that has been long studied in various contexts. A trivial way to estimate $P(v \mid e)$ can be to use the text associated with the entity and construct a multinomial distribution using the term frequencies in the text. However, such an approach would not serve the main purpose of term weighting which is to distinguish important terms relevant to an entity from the irrelevant ones. This is mostly due to the fact that any two terms that are equally represented in the context of an entity may not be equally important when the structural information (class and relationship) or topics of the entity are considered. For example, *"actress"* and *"humanitarian"* are the words that appear in the description of *Audrey Hepburn* entity ($p2$ in Fig. 5.1-a) and are equally represented, but *"actress"* is a more important term for that entity when we consider that *Audrey Hepburn* has played in many

movies and also more related to the topic of movies. In particular, we consider the following points to distinguish important terms while estimating the probability $P(v \mid e)$:

- Topical importance is highly critical in term weighting to distinguish the importance. In fact, the words that are highly rated according to the topics of the entity are assumed to be weighted high by that entity. In TRM, topics are learned from the underlying data and often correspond to meaningful semantic themes presented in the corpus. As entities are also related to different topics with different mixture of topic proportions as learned by TRM, we utilize this information for weighting the terms.
- Structural information of the entity is also as important as the entity text. In RDF data graph, entities can be member of different classes, or have relationships to different entities. Since TRM also learns the topical correlations between these structural elements and the topics, we can easily exploit this information to weight terms correlated with the structural information of the entity.

In order to reflect this variety of information in the probability $P(v \mid e)$, we represent it as linear weighted combination of some basis probabilities $P_\phi(v \mid e)$ each of which has a weight w_ϕ[1] as:

$$P(v \mid e) = \sum_{\phi \in \Phi} w_\phi P_\phi(v \mid e) \tag{5.5}$$

Every basis probability $P_\phi(v \mid e)$ weights the terms according to different criteria while its weight specifies its importance. Fig. 5.1-b illustrates this intuition of decomposing $P(v \mid e)$ into n basis probabilities. Here, the number n of basis probabilities varies for every entity since the structural information in the data graph changes from one entity to another. Mainly, we exploit four unique types of evidence about the entity for term weighting: The first source of evidence is the text in its attributes which lists important words associated with the entity. In this simple probability, we utilize the maximum likelihood estimation of the term by directly using its frequency in the entity text to obtain the probability $P_{\phi ML}(v \mid e)$. As another source of evidence, instead of directly deriving term weights using this text as it is, we assign a probability to each term according to their importance in the topics of the entity. Since in TRM we learn topic proportions $\boldsymbol{\theta}(e)$ for each entity, we can exploit this information to calculate attribute-specific term weight. denoted as $P_{\phi_a}(v \mid e)$, across all topics as follows:

$$P_{\phi_a}(v \mid e) = \sum_t \mathbb{I}\left[< e, v > \in \mathcal{O}(w)\right]\ p(v \mid \beta_t)\ p(\theta_t \mid \mathbf{b}(e), \rho) \tag{5.6}$$

Here $\mathbb{I}\left[< e, v > \in \mathcal{O}(w)\right]$ is the indicator function that checks whether v is included in the attribute text of the entity e, and $p(v \mid \beta_t)$ is the probability of the word in a specific topic and equals to the learned TRM parameter $\beta_{t,v}$. However, the probability of the topic for the entity, $p(\theta_t \mid \mathbf{b}(e), \rho)$, is not directly learned in TRM. Instead we use variational methods in Sec. 4.2.3 to approximate these probabilities with fully-factorized distribution q, so we can replace

[1]Note that weights of the basis probabilities are different from the weights assigned to each term by the entity

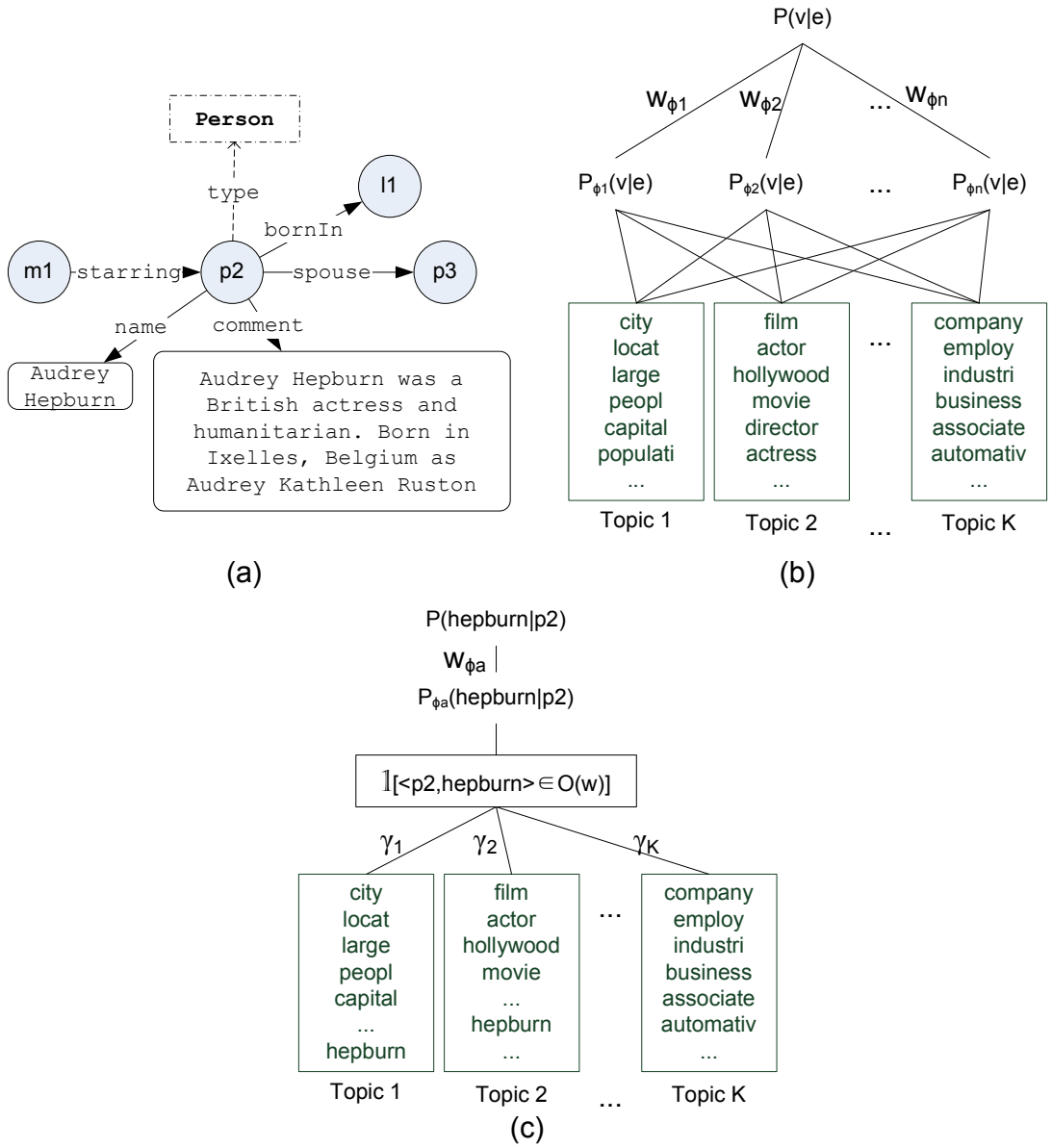


Figure 5.1: (a)Information about the entity of Audrey Hepburn in the RDF graph. (b) Decomposition of term weighting probability $P(v \mid e)$ into basis probabilities. (c) Attribute-based term weighting of the word *hepburn* for the entity $p2$

the term $p(\theta_t \mid \mathbf{b}(e), \rho)$ with its expectation w.r.t. the q as follows:

$$\begin{aligned} P_{\phi_a}(v \mid e) &\approx \sum_t \mathbb{1}\left[< e, v > \in \mathcal{O}(w)\right]\ p(v \mid \beta_t)\ \mathbb{E}_q\left[p(\theta_t \mid \mathbf{b}(e), \rho)\right] \\ &= \sum_t \mathbb{1}\left[< e, v > \in \mathcal{O}(w)\right]\ \beta_{t,v}\ \gamma_t(e) \end{aligned} \tag{5.7}$$

where $\mathbb{E}_q\left[p(\theta_t \mid \mathbf{b}(e), \rho)\right]$ is replaced with variational parameter $\gamma_t(e)$. Basically in this probability, term weighting depends on thematic importance of the term in particular topics and also the importance of the topics for that entity. Fig. 5.1-c illustrates this intuition to weight the term *hepburn* for the entity $p2$. The term gets higher weights only if it is important for the topic and if the topics are regarded important for the entity.

Another source of evidence that we employ for term weighting is the classes that the entity has in the data graph. In fact, RDF does not restrict the number of classes that an entity can belong to. As discussed in Sec. 4.2.1 we already associate classes with topics using $\boldsymbol{\lambda}$ parameter to find important topics. This information is useful to obtain important terms for each class and we introduce a basic probability $P_{\phi_c}(v \mid e)$ for every class c of the entity to incorporate class-specific terms into the core probability $P(v \mid e)$. $P_{\phi_c}(v \mid e)$ is given by:

$$P_{\phi_c}(v \mid e) = \sum_t p(v \mid \beta_t)\ \widetilde{\lambda}_{ct} = \sum_t \beta_{t,v}\ \widetilde{\lambda}_{ct} \tag{5.8}$$

This probability is similar to attribute-specific $P_{\phi_c}(v \mid e)$ except now we do not restrict words to the attribute text and also use normalized class-topic weights $\widetilde{\lambda}_{ct}$ of the corresponding TRM parameter. The underlying intuition here is to introduce terms describing a class to the entity in order to exploit this type of further semantics. For example, the entity $p2$ is of type of the class $Person$ that will assign higher probabilities to the terms related to a person from the underlying topic distributions.

The last source of evidence that we consider for term weighting is the relationships that an entity has. In particular, the topical correlations between any particular relationship r and the topics in TRM are captured as a $K \times K$ matrix ω_r which assigns a weight between any source and target topic (e.g. see Fig. 4.5). We utilize this information to derive the term weights from a basis probability $P_{\phi_r}(v \mid e)$ of every unique relationship r of the entity as follows:

$$P_{\phi_r}(v \mid e) = \sum_t p(v \mid \beta_t)\ \widetilde{\omega}_{rt} = \sum_t \beta_{t,v}\ \widetilde{\omega}_{rt} \tag{5.9}$$

Here, $\widetilde{\omega}_{rt}$ is normalized weight of the topic t for relationship r and calculated as follows for source entities:

$$\widetilde{\omega}_{rt} = \frac{\sum_{t'} \omega_{rtt'}}{\sum_{t'} \sum_{t''} \omega_{rt't''}}$$

and, for target entities:

$$\widetilde{\omega}_{rt} = \frac{\sum_{t'} \omega_{rt't}}{\sum_{t'} \sum_{t''} \omega_{rt't''}}$$

The set $\Phi = \{\phi_{ML}, \phi_a, \phi_{c_1}, ..., \phi_{c_k}, \phi_{r_1}, ..., \phi_{r_l}\}$ (where $k = |C(e)|$ and $l = |R(e)|$) denotes different evidences that we consider for each entity e in Eq. 5.5. Although we exploit only these four types of evidence for term weighting, the Eq. 5.5 is generic enough to incorporate more complex sources of evidence such as term weighting over a path in the data graph.

5.2.3 Optimizing Entity Weights

Although TRM is useful to construct basis probabilities $P_\phi(v \mid e)$, it does not say much about the corresponding weights w_ϕ of these probabilities. However, in the final term weighting $P(v \mid e)$, these weights are crucial because they indicate which semantic information (attributes, classes or relationships) is useful to improve the search performance. In order to achieve this we propose a supervised learning setting in which the weights can be optimized by directly measuring the retrieval performance. However training data directly addressing the entities is difficult to obtain. Instead we propose to select a subset of queries $\mathcal{Q}_e \subset \mathcal{Q}$ that are assumed to be relevant to a particular entity by using Eq. 5.4 and use existing IR datasets in order to optimize the weights based on their effectiveness on improving the retrieval performance. For this we consider a special retrieval setting that the PRF set F in Eq. 5.2 only contains the entity for which we want to optimize the weights and derive a new scoring function that is limited to that entity as follows:

$$S_e(q, d) = \sum_{v \in V} \sum_{\phi \in \Phi} w_\phi P_\phi(v \mid e) P(e \mid q) \log P(v \mid d) \tag{5.10}$$

Since there is only one entity in the PRF set, $P(e \mid q)$ is assumed to be one and ignored. Using S_e we can directly measure the performance of the entity in the retrieval setting by using widely-accepted IR measures such as average precision (AP) or normalized discounted cumulative gain (NDCG) (see Appendix B). For this, we define a performance measure as a function $\mathrm{E} : \mathcal{S} \rightarrow \mathbb{R}_{[0,1]}$ which takes a scoring function $S \in \mathcal{S}$, returns a value on the interval 0.0 to 1.0, and thus is comparable across queries. Based on E we adopt a loss function from a logistic regression error [23, pp. 337] between any two scoring function as follows:

$$\mathrm{L}(S_1, S_2) = \log(1 + \exp(k(-\mathrm{E}(S_1) + \mathrm{E}(S_2))))$$

where $k > 0$ is a constant used for adjustment. This loss function is not so crisp as the other loss functions such as hinge loss, or 0-1 loss, but instead monotonically decreasing as the difference $\mathrm{E}(S_2) - \mathrm{E}(S_1)$ increases. Fig. 5.2 depicts a plot of the loss function w.r.t. difference between the performance measures. As the second scoring function S_2 performs better than the first one, the loss decreases. In order to evaluate the performance of the entity we compare the entity specific scoring function given in Eq. 5.10 with a standard query likelihood score S_q for the original query q [179]. Although with one entity e in the PRF set, the scoring function S_e may not perform as well as the scoring function S_q for every query. However, the loss function can still measure the effectiveness if the performance of S_e approaches to the performance of S_q. We set the second parameter S_2 to S_e and the first parameter S_1 to S_q, so the maximum loss is obtained when $\mathrm{E}(S_e) = 0$ and $\mathrm{E}(S_q) = 1$ (i.e. $\mathrm{E}(S_e) - \mathrm{E}(S_q) = -1$) while the minimum

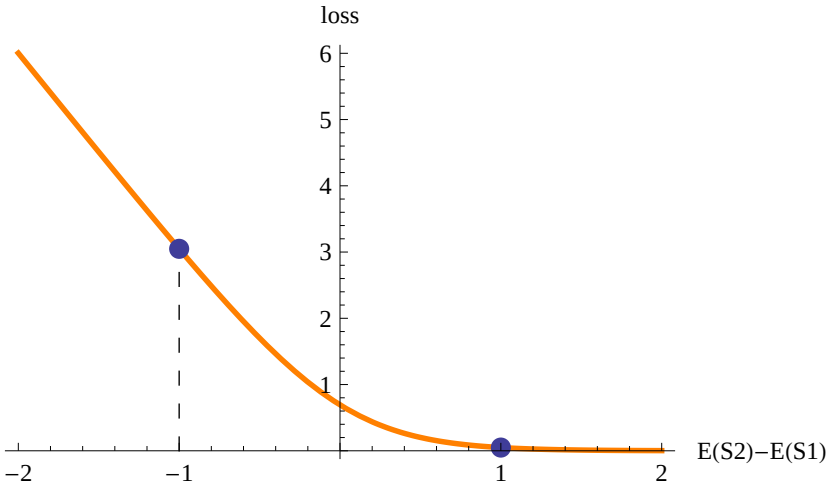


Figure 5.2: Logistic regression loss function for performance measure $\mathrm{E}(S_2) - \mathrm{E}(S_1)$. The points show the maximum and minimum values of possible loss for $E(S) \in [0, 1]$ (k=3).

loss is obtained for $\mathrm{E}(S_e) = 1$ and $\mathrm{E}(S_q) = 0$ (i.e. $\mathrm{E}(S_e) - \mathrm{E}(S_q) = +1$). The points in Fig. 5.2 indicate these maximum and minimum loss, respectively. Also note that loss decreases fastest in the second region (i.e. $\mathrm{E}(S_e) < \mathrm{E}(S_q)$) giving more emphasis on closing the gap between $\mathrm{E}(S_e)$ and $\mathrm{E}(S_q)$ during optimization. Then, our goal is to minimize the overall loss for all queries in the training data by optimizing the entity weights. This leads to the following optimization problem:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{|\mathcal{Q}_e|} \sum_{q \in \mathcal{Q}_e} \log(1 + \exp(k(-\mathrm{E}(S_q) + \mathrm{E}(S_e)))) \tag{5.11}$$

Here, we assume each query to be uniform and used $\frac{1}{|\mathcal{Q}_e|}$ to indicate that all queries are equally important. A number of optimization techniques can be applied to solve this problem, and we employ gradient descent with random restarts [98]. That is, the algorithm to minimize the loss starts from a random weights $\mathbf{w}$, and at each iteration, updates each weight in $\mathbf{w}$ in the opposite direction of the gradient – i.e. the partial derivative of the loss function w.r.t. each weight w_ϕ. This procedure is only to find the local optimum of the lost function given a certain initial weights. Then, we randomly re-initialize the weights and do another round of optimization (i.e. random restarts). Finally we regard the best local optimum as a global optimum value of weights. According to the chain rule, the partial derivative of the loss function w.r.t. each weight w_i can be written as:

$$\frac{\partial \mathrm{L}(S_q, S_e)}{\partial w_\phi} = \frac{\partial \mathrm{L}(S_q, S_e)}{\partial \mathrm{E}(S_e)} \frac{\partial \mathrm{E}(S_e)}{\partial w_\phi} \tag{5.12}$$

Our loss function $\mathrm{L}(S_q, S_e)$ is continuous and differentiable so we can easily obtain the

first term of the partial derivative above as follows:

$$\frac{\partial \mathrm{L}(S_q, S_e)}{\partial \mathrm{E}(S_e)} = k \frac{\exp(k(-\mathrm{E}(S_q) + \mathrm{E}(S_e)))}{1 + \exp(k(-\mathrm{E}(S_q) + \mathrm{E}(S_e)))} \tag{5.13}$$

For the second term, however, we need a performance measure $\mathrm{E}(S_e)$ that is differentiable. The standard performance measures in the IR such as AP, or NDCG are position-based and thus non-continuous and non-differentiable. That's why we cannot directly take the derivative and use it in the optimization problem here. Instead, we take an approach of approximating the performance measure E as introduced in [186] which presents a reformulation of the major IR measures to make them continuous and differentiable. Specifically, we use the NDCG measure as E here, but a similar approach can also be applied to AP. According to [186], the NDCG measure can be re-formulated as:

$$NDCG = N_n^{-1} \sum_d \frac{2^{r(d)} - 1}{\log(1 + \pi(d))}$$

where $r(d)$ is equal to one for relevant documents and zero for irrelevant documents, N_n is a constant to make the value of NDCG on the interval 0.0 and 1.0, and $\pi(d)$ is position function that can be represented as a function of scores as:

$$\pi(d) = 1 + \sum_{d', d \neq d'} \frac{\exp[-\alpha(S_e(q, d) - S_e(q, d'))]}{1 + \exp[-\alpha(S_e(q, d) - S_e(q, d'))]}$$

Here, $\alpha > 0$ is a scaling constant which is set to 100 in our experiments. The position function $\pi(d)$ actually compares the score of each document with the other documents and uses logistic function as an approximation to make it continuous and differentiable. Applying the chain rule once again, we derive the gradient for the second term as:

$$\frac{\partial \mathrm{E}(S_e)}{\partial w_\phi} = \frac{\partial NDCG}{\partial w_\phi} = N_n^{-1} \sum_d \frac{\partial \frac{2^{r(d)}-1}{\log(1+\pi(d))}}{\partial \pi(d)} \frac{\partial \pi(d)}{\partial w_\phi} \tag{5.14}$$

where

$$\frac{\partial \frac{2^{r(d)}-1}{\log(1+\pi(d))}}{\partial \pi(d)} = -\frac{2^{r(d)} - 1}{(\log(1 + \pi(d)))^2} \frac{1}{(1 + \pi(d))} \tag{5.15}$$

and

$$\begin{aligned} \frac{\partial \pi(d)}{\partial w_\phi} \quad = \quad & -\alpha \sum_{d', d \neq d'} \frac{\exp[\alpha(S_e(q, d) - S_e(q, d'))]}{(1 + \exp[\alpha(S_e(q, d) - S_e(q, d'))])^2} \\ & \left(\frac{\partial S_e(q, d)}{\partial w_\phi} - \frac{\partial S_e(q, d')}{\partial w_\phi} \right) \end{aligned} \tag{5.16}$$

Additionally, the scoring function S_e is also differentiable and we can take the derivative

of Eq. 5.10 as:

$$\frac{\partial S_e(q,d)}{\partial w_\phi} = \sum_{v \in V} P_\phi(v \mid e) \log P(v \mid d) \tag{5.17}$$

Substituting Eq. 5.17 into Eq. 5.16, and Eq. 5.16 and Eq. 5.15 into Eq. 5.14, we obtain the second term in Eq. 5.12. Finally, we get the gradient for w_ϕ, i.e. $\bigtriangledown w_\phi$, by substituting Eq. 5.13 and Eq. 5.14 into Eq. 5.12 over all the queries as:

$$\bigtriangledown w_\phi = \frac{1}{|\mathcal{Q}_e|} \sum_{q \in \mathcal{Q}_e} \frac{\partial \mathrm{L}(S_q, S_e)}{\partial w_\phi}$$

At each iteration i, the weights are updated with the gradient as $w_\phi^{i+1} = w_\phi^i - \gamma \bigtriangledown w_\phi^i$ where γ is step-size constant. Algorithm 3 depicts the pseudo-code of the optimization algorithm. It expects a set of training queries, associated document and relevant judgments as input. In addition, a fully trained TRM the base probabilities $P_\phi(v \mid e)$ are given as additional input. It performs R number of restarts. In each iteration, the algorithm 1) randomly initialize the weights, 2) executes gradient descent until converges, 3) computes the gradient of loss function over the queries, and 4) updates the weights accordingly using a step-size of γ. Also note that the resulting weights of each restart is added to the output set $\mho$ and finally returns those weights that has minimum loss calculated as the objective function of optimization problem in Eq. 5.11.

5.3 Experiments

We will now discuss the experiments we performed to study the performance of our proposed approach.

5.3.1 Dataset

We conducted experiments on five data sets taken from TREC: the Associated Press Newswire (AP) 1988-90 with queries 51-150, Wall Street Journal (WSJ) 1987-92 with queries 51-100 and 151-200, Financial Times (FT) 1991-94 with queries 301-400, San Jose Mercury News (SJMN) 1991 with queries 51-150, and Web Collection (WT10g) with queries 451-550. Queries are taken from the "title" field of the TREC topics. Relevance judgments are obtained from the pool of top documents returned by various retrieval systems from previous TREC conferences. Queries that have no relevant documents for a specific collection have been removed from the query set for that collection. Statistics of the collections and query sets are given in Table 5.1. Version 3.0 of the Terrier IR Platform [174] is used for indexing and retrieval tasks. In addition, Porter's stemming and stop word removal are applied to all collections.

As semantic data, we use the same dataset from DBPedia [7] as described in Sec. 4.3 to train TRM with 20094 distinct entities. Most of the selected entities are in the categories Cities, Countries, People, Music, Film and Science.

Algorithm 3: Optimize Entity Weights

Input: training queries $\mathcal{Q}_e$, associated documents D, and relevance judgments $\{r(d)\}_{d \in D}$
Input: step size γ, number of restarts R
Input: $P_\phi(v \mid e)$ for each $\phi \in \Phi$ of entity e
  initialize $\mathrm{E}(S_q)$ for every $q \in \mathcal{Q}_e$
  for r=1 to R **do**
    randomly initialize the weights w_ϕ^0
    initialize output set $\mathcal{O}$ of weights
    $i \leftarrow 0$
    while (not converge) **do**
      $\bigtriangledown w_\phi^i \leftarrow 0, \forall w_\phi$
      for each query $q \in \mathcal{Q}_e$ **do**
        compute $\mathrm{E}(S_e)$ w.r.t. the weights w_ϕ^i
        compute the gradient $\frac{\partial \mathrm{L}(S_q, S_e)}{\partial w_\phi}$ using Eq. 5.12 w.r.t. $\mathrm{E}(S_e)$ and $\mathrm{E}(S_q)$
        $\bigtriangledown w_\phi^i = \bigtriangledown w_\phi^i + \frac{\partial \mathrm{L}(S_q, S_e)}{\partial w_\phi}$, $\forall w_\phi$
      end for
      $\bigtriangledown w_\phi^i = \frac{1}{|\mathcal{Q}_e|} \bigtriangledown w_\phi^i, \forall w_\phi$
      $w_\phi^{i+1} = w_\phi^i - \gamma \bigtriangledown w_\phi^i, \forall w_\phi$
      $i \leftarrow i + 1$
    end while
    add w_ϕ^i to $\mathcal{O}$
  end for
Output: return the weight $w_\phi \in \mathcal{O}$ with minimum loss

Table 5.1: Dataset Statistics

Collection	Contents	Docs.Count	Queries (TREC Topics)	# of valid Queries
AP	Associated Press Newswire 1988-90	242918	51-150	99
FT	Financial Times 1991-94	209705	301-400	95
SJMN	San Jose Mercury News 1991	90257	51-150	94
WSJ	Wall Street Journal 1991-94	173252	51-100 & 151-200	100
WT10g	TREC Web Collection	1692096	451-550	100

5.3.2 Parameters

There are several parameters that need to be determined in our experiments.

For the baselines, we use the Dirichlet prior $\mu \in \{500{,}750, 1000, 1500, 2000, 2500, 3000\}$ and the number of feedback documents $F_D \in \{5, 10, 25, 50, 75, 100\}$, and the number of expansion terms $e \in 10, 25, 50, 75, 100$.

For TRM, the maximum number of topics is set to $K = 100$. We performed 100 iterations for outer loop and 20 iterations for inner loop to obtain the posterior distribution. The initial values of the TRM parameters (e.g. ω,λ,β,α, ρ) are set at random, and then updated according to the mechanism described in Sec. 4.2.3.

For SRM, we set the parameter of the loss function $k = 3$ and the linear interpolation parameter $\lambda_e \in \{0.1, 0.2, ..., 0.5\}$ for scoring entities. It has been shown that making the document smoothing parameter λ_D dependent on the document length yields superior performance [255]. Therefore, we set $\lambda_D = \frac{\rho}{|D|+\rho}$, where ρ is the average document length for each collection. The number of entities and the number of expansion terms is same as the RM baseline. For both the baselines and the implementation of our approach, we sweep over these parameters' values to find the best configuration. That is, we compare the upper bound performances of these systems.

5.3.3 Methods

We test the SRM in comparison to four baselines. As a standard baseline, we employ document-based retrieval with a *query-likelihood language model* (LM), using a Dirichlet prior for smoothing. To test the performance of the baseline pseudo-relevance feedback model, we use RM as described in [133]. In addition to these baselines, we also use *Relevance Model with Wikipedia* (RMW), which is presented in [246] utilizing Wikipedia articles as the external corpus (Wikipedia April 2011 dump). Finally, the approach presented in [63] is used as *Relevance Model with External Corpus* (RME) that utilizes a large external corpus to improve RMs. We select the GOV2 dataset as the external corpus for that. We exclude the LDA-based language model approach presented in [238] since it has already been reported to perform poorer than RM.

5.3.4 Training

In order to optimize entity weights we train the model using the optimization algorithm presented above. In order to find a subset $\mathcal{Q}_e$ of queries for each entity, we need to select from a large pool of queries so that sufficiently relevant queries can be found. For this we use a separate dataset from TREC 2009 Millions Query Track which includes a large set of 40,000 queries most of which have the corresponding relevance judgments. A subset of queries are selected for each entity using the Eq. 5.4. We used top-50 queries according to the query-likelihood score between query and entity. For dataset, we used *ClueWeb09-CatB* which consists of 50 million pages collected from the Web between Jan. 2009 and Feb. 2009. This training is conducted as offline process and the training dataset is not included in the evaluation since the weights are optimized specifically for this dataset for the fairness of the results.

Table 5.2: Comparison of Language Model (LM), Relevance Model (RM) and Semantic Relevance Model (SRM). The evaluation measure is average precision. Stars * and ** indicate statistically significant improvements over the LM and RM baseline, respectively. We use the paired t-test with significance at p<0.05

	LM	**RM**	**SRM**	**chg% over LM**	**chg% over RM**
AP	0.2094	0.2655	0.2889	37.96*	8.81**
FT	0.2606	0.2812	0.3140	20.49*	11.66**
SJMN	0.2134	0.2567	0.2842	33.17*	10.71**
WSJ	0.2833	0.3142	0.3327	17.43*	5.89**
WT10g	0.1832	0.1973	0.2204	20.30	11.70**

Table 5.3: Comparison of Relevance Model with Wikipedia (RMW), Relevance Model with External Corpus (RME) and Semantic Relevance Model (SRM). The evaluation measure is average precision.

	RMW	**RME**	**SRM**	**chg% over RMW**	**chg% over RME**
AP	0.2592	0.2695	0.2889	11.45*	7.19**
FT	0.2887	0.2944	0.3140	8.76*	6.65**
SJMN	0.2641	0.2673	0.2842	7.61*	6.32**
WSJ	0.3105	0.3285	0.3327	7.15*	1.27
WT10g	0.2052	0.1879	0.2204	7.40*	17.29**

5.3.5 Results

Retrieval results on the five datasets for the first two baselines (LM and RM) are presented in Table 5.2. The SRM method shows better performance than these baselines on all test collections with 8.81%, 11.66%, 10.71%, 5.89% and 11.70% improvement over the RM baseline. Especially, for FT and WT10g, the improvement is large in comparision to the improvement between LM and RM. This is because there is a large overlap between the TREC topics 301-400, 451-550 and the entities in the employed dataset. Besides, Table 5.3 shows the comparision of results for the baselines RMW and RME. RMW provides results comparable to the standard RM, whereas RME performs better than RM for the newswire collections. Our method provides better performance in all collections, except for WSJ, on which RWE also demonstrates similar results.

A critical observation for the SRM, however, is that the improvement per query is not so uniform as in the case of the RM. That is, SRM performs significantly well for some queries, while degrades to the baseline scores for some others. This is mostly due to the fact that the information overlap between the information need (i.e. query) and the available semantic data plays a crucial role. For example, for the TREC topic-400, the query *Amazon rain forest*, the

SRM method selects *Amazon river*, *Amazon rainforest*, *Brazil* and *Guyana* as top entities. This gives more emphasis to the features related to the *Location* and *country* that in turn, promotes the terms *brazil* and *rio* as well as the term *environment*. For this query, the document "FT922-4344" is a relevant document that ranks low in the baselines LM and RM since the document does not contain the exact query terms *Amazon rain forest* (and apparently, also does not contain the terms in the expanded RM query). Using SRM, this document is closely related to the top entities resulting in better precision for that query. The average precision of this query improves from 0.5088 to 0.7243.

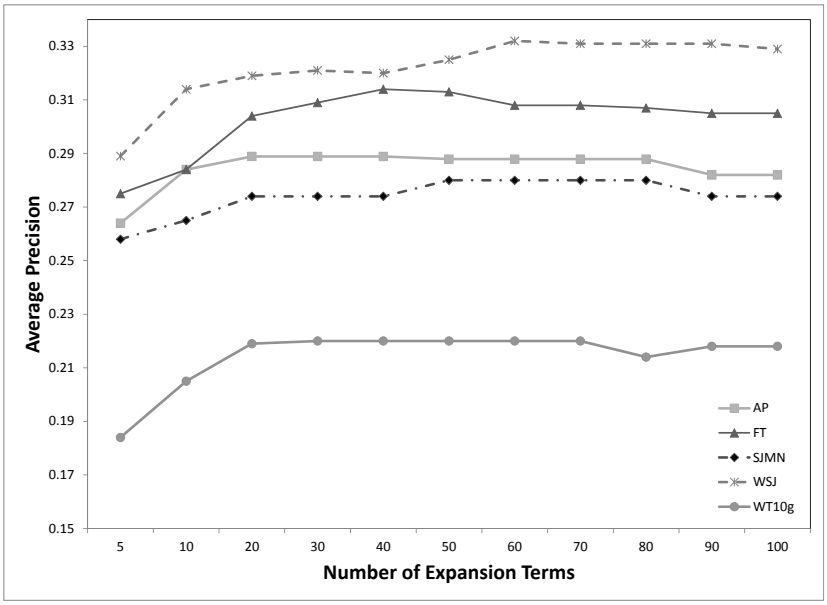


Figure 5.3: Sensitivity to the number of expansion terms in different collections

This, in fact, shows the added value of the SRM over the baseline methods as it performs relatively well when there is background knowledge in semantic resources that can be incorporated. This is particularly true for queries that can be mapped to entities in the DBPedia dataset. DBPedia, in fact, puts a special emphasis on the entities and relations concerning *Person*, *Location*, *Movies*, *Companies*, etc. However, for queries that refer to more abstract entities, the SRM method fails to locate the correct entities and degrades to the baseline scores. For example, for the TREC topic 362 *human smuggling*, the performance of the SRM is average. There are simply no data in Wikipedia, which correspond to this topic, and thus, our approach was not able to infer the correct information need. Clearly, this result is not surprising as already pointed out in the motivation of our work: we aim to exploit semantic data to obtain a more fine-grained, topical understanding of the needs. However, this is only possible if these needs are "covered" by the available semantic data, i.e., the ones that correspond to what are captured by the data. This experiment suggests that some TREC queries are not covered in this sense. Yet, given the amount of data is increasing, more sophisticated strategies for combining data from different sources to improve the coverage is possible on top of our work.

We further analyze the sensitivity of average precision w.r.t. the number of expansion terms in Figure 5.3. The number varies from 5 to 100. We observe that an optimal result is reached at different numbers of expansion terms for different collections. For WT10g and AP collections, the optimal results do not change significantly after 20 terms. The optimal results are reached at 40, 50, and 60 for FT, SJMN, and WSJ collections, respectively. It seems that with terms weighted according to the SRM model, performance is good using highly weighted terms, whereas it slightly drops as we reach 100.

5.4 Conclusions

We proposed a solution for exploiting a large and ever increasing amount of semantic data for document retrieval on the Web. We employ TRM as a generative model to capture the entities as mixtures of topics, and sample both from text and structured data to obtain a semantic relevance model that captures the query semantics in terms of entity-specific mixtures of topics. Inferring the information needs using the semantics of external datasets helps to obtain a more fine-grained understanding and to avoid expanding the query with terms not relevant to the need. In comparison with the baseline, which uses more coarse-grained documents for query expansion, large improvements could be achieved, given the queries are about entities that are covered by the external semantic data. While this was not true for several TREC queries used in the experiments, recent studies done by Yahoo! showed that queries of this type are indeed common on the Web. Also, this work is highly applicable to real-life enterprise settings, where most queries are likely to be covered by the ubiquitously available databases and data warehouses. So far, we focused our attention on inferring the information needs while the framework we proposed paves many other ways for exploiting external semantic data.

6

Semantic Relevance Model for Data Retrieval

In this chapter, we focus on the keyword search on structured data which has gained a lot of interest as keywords have proven to be an intuitive mean for accessing information and the amount of available structured data increases rapidly, especially in the realm of Web databases. Keyword search helps to circumvent the complexity of structured query languages, and hide the underlying data representation. Without knowledge of the query syntax and data schema, users can obtain complex structured results, including tuples from relational databases, XML data, data graphs, and RDF resources [2, 90, 114, 225]. As opposed to standard keyword search where retrieval units are single documents, results in this setting may encompass several resources that are connected over possibly very long paths (e.g. joined database tuples, XML trees, RDF resources connected over paths of relations).

Although much attention has been focused on finding efficient techniques to process keyword queries and to retrieve the structured data in these settings, only a small number of dedicated work can be found on the ranking of results and understanding their relevance to the user information need. One main direction is the use of Information Retrieval (IR) inspired TF-IDF ranking [143, 150]. Another direction is proximity search where the goal is to minimize the distance between data elements matching the keywords [2, 114]. For computing this weight of paths (distance), a PageRank-based metric is often incorporated for including the node prestige [114]. While high-quality results have been reported in the ad-hoc evaluations of these approaches, a recent study [47] that specifically focuses on benchmarking the effectiveness of keyword search ranking strategies has revealed serious problems: in contrast to previously published results, there is no ranking strategies that is clearly superior, and effectiveness results are much worse than those reported when using a principled approach for assessing relevance and a broader spectrum of queries.

The major shortcomings of previous work can be summarized as follows: the minimal distance heuristic behind proximity search is rather convenient for the efficient computation of results but does not directly capture relevance. The adoption of IR-based ranking proposed so far is also problematic because unlike document ranking, the score of a result in this setting is an aggregation of several resources' scores. Combining resources with high "local

scores" however, does not always guarantee highly relevant final results [150]. Recent evaluation results [47] suggest that the proposed normalization methods [143, 150] are not effective in dealing with this issue. More importantly, previous work implicitly assumes that relevance is completely captured by the keyword query that is mostly short and ambiguous. We consider this assumption to be too strong especially in this setting, where users might not be aware of the underlying structure and terminology of the database and thus, cannot specify "complete queries".

In this chapter, we make the following contributions: (1) we propose the use of Relevance Models (RM) for dealing with the ranking of structured results in keyword search. RM has been used for document ranking [133], which is a probabilistic model that directly captures the notion of relevance behind documents and queries. Primarily, it is employed as a form of query expansion, where relevance (and the query model) is constructed based on the set of documents obtained from an initial query run, i.e., via pseudo-relevance feedback (PRF). While it has shown to be effective in dealing with document ranking, the use of RM has not been studied in this keyword search setting before. (2) We adopt this model, with the goal of obtaining a representation more fine-grained than the original RM that can also exploit the structure of the PRF results. That is, instead of relying on the possibly short and ambiguous query, we use a model of relevance derived from both the content and structure of PRF results. (3) We introduce smoothing strategies that exploit the specific structure and semantics of the underlying data to obtain better estimates of probabilities that make up the RM. (4) We proposed a relevance-based ranking function that employs this adopted RM for ranking standalone as well as aggregated structured results. In particular, we show that this ranking function does not only provide a principled way for the aggregation of "local scores", but also, it is monotonic. This is a crucial design issue because only monotonic functions are compatible with existing top-k techniques proposed for the efficient computation of keyword search results. (5) As opposed to previous ad-hoc experiments, we employ the general framework recently proposed for evaluating keyword search ranking strategies [47]. To the best of our knowledge, this is the first work that – based on a standardized evaluation – largely and consistently outperforms all existing systems across datasets and information needs in terms of precision, recall and MAP. The content of this chapter is partially published in some of our previous publications. In [21], RM-based ranking model for data retrieval is presented. Additionally, the application of the proposed approach to search structured environmental data is presented in [1, 16].

Structure. We introduce the problem of computing and ranking structured results for keyword queries in Section 6.1. Then, our approach for ranking is discussed in detail in Section 6.2. We also present how the proposed ranking scheme can be used in top-k query processing to retrieve the structured results in Section 6.3. Section 6.4 contains results of the experiments. We make discussion of our work in comparison with the related works in Section 6.5 and conclude in Section 6.6.

6.1 Keyword Search on Structured Data

In this section, we provide the definition of our keyword search problem and present an overview of existing approaches.

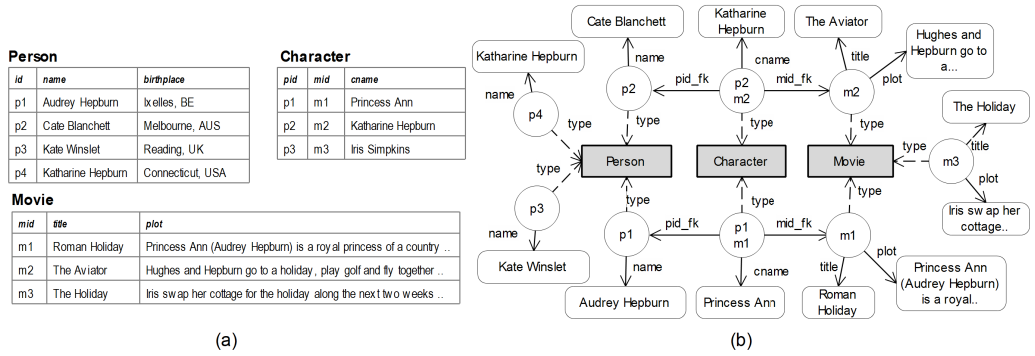


Figure 6.1: a) An example database from IMDB and b) its corresponding data graph (partially shown).

6.1.1 Problem Setting

Keyword search approaches have been proposed for dealing with different kinds of data, including relational, XML and RDF data. Generally speaking, the underlying data can be conceived as a directed labeled data graph $G = (V, E)$, where V is a disjoint union ($V = V_R \uplus V_A$) of resource nodes (V_R), and attribute nodes (V_A), and $E = E_F \uplus E_A$ represents a disjoint union of relation edges also called foreign key edges (E_F) that connect resource nodes, and attribute edges (E_A) that link between a resource node and an attribute node. In the following, we denote all attributes of the resource $r \in V_R$ as $\mathcal{A}(r)$, the attributes reachable from r via the edge e as $\mathcal{A}(r, e)$, and all the outgoing edges of r as $\mathcal{E}(r)$. This model closely resembles the graph-structured RDF data model (omitting special features such as RDF blank nodes). The intuitive mapping of this model to relational data is as follows: a database tuple captures a resource, its attributes, and references to related resources in the form of foreign keys; the column names correspond to edge labels. Example data and its corresponding data graph is shown in Figure 6.1.

The user query Q is a set of keywords $(q_1, ..., q_m)$. A keyword matches a resource node r if it matches any of the attribute nodes $\mathcal{A}(r)$, or any of the edges $\mathcal{E}(r)$. An answer to the user query Q is a minimal rooted directed tree that can be found in the data graph, which contains at least one matching resource for every keyword in Q. This is commonly referred to as a Steiner tree [2, 114]. In this work, we use the term *Joined Resource Tree* (JRT) to make clear that an answer is a joined set of resources represented as $JRT = \{r_1, \ldots, r_n\}$. In recent work, subgraphs have also been considered as keyword answers [225], instead of trees. While this difference in semantics requires more advanced mechanisms for result computation, we will show that this aspect is orthogonal to the problem of *keyword search result ranking* studied in this chapter: given a user query, the goal here is to rank Steiner trees (or graphs) according to the user's perceived degrees of *relevance*. In other words, we want to produce a ranking of keyword search results that corresponds to the degree to which they *match the user information needs*. Unlike previous approaches which employ ad-hoc notions of relevance, and conducted different (and rather ad-hoc) experiments to assess this relevance, we aim to

provide a principled approach to modeling and dealing with relevance, and will follow the general evaluation framework for evaluating ranking strategies as proposed recently [47].

6.1.2 Keyword Search Result Computation

Clearly, the main difference of keyword search on structured data and the traditional keyword search on documents is that instead of one single document, a Steiner tree may encompass several resources (e.g. database tuples, documents, RDF resource descriptions) that are connected over a possibly very long path of foreign key relationships.

There are *schema-based approaches* implemented on top of off-the-shelf databases ([97, 143, 150]). Basically, a keyword query is processed by mapping keywords to elements of the database. Then, candidate networks are computed from the schema, which represent valid join sequences (Steiner tree templates) that can be used to connect computed keyword elements. Formal structured queries are derived from candidate networks that finally are evaluated using the underlying database engine. The main advantages of this approach is that the power and optimization capabilities of the underlying database engine can be fully utilized for computing structured results.

The *schema-agnostic approaches* ([90, 114, 139]) operate directly on the data. Since they do not rely on a schema, the applicability of these approaches is not limited to structured data. For instance, schema-agnostic approaches have been proposed for dealing with semi-structured RDF data [225] as well as the combination of structured, semi-structured and unstructured data [139]. Also here, keyword elements have to be identified first. Then, Steiner trees are iteratively computed by exploring the underlying data graph that is mostly held in memory. For the query "Blanchett Aviator" for instance, this graph is simply the path between $p2$ and $m2$ in Fig. 6.1. Various kinds of algorithms have been proposed for the efficient exploration of data graphs, which might be very large (e.g. bidirectional search [114]).

While this large body of work on result computation is in principle orthogonal to the problem of ranking, there is one critical issue to be considered. Namely, existing approaches employ top-k processing techniques to focus only on the best results in order to terminate as early as possible. In particular, top-k processing terminates when the score of the k-best result obtained so far is guaranteed to be at least as high as the maximum score that can be achieved with the remaining candidates. The score of a result (i.e., a JRT), is typically defined as an aggregation of scores of the individual resources. The upper bound guarantee that is necessary for top-k can be ensured, when the overall score monotonically increases as more resources are added to the result during the process. In other words, existing top-k techniques assume the ranking function to be monotonic, a requirement we aim to satisfy so that our proposal can be used in combination with this previous work.

6.1.3 Keyword Search Result Ranking

The main idea behind ranking is to (1) assign each resource r in the JRT a score $Score(r)$, and then to combine the individual scores using a monotonic aggregation function $agg(\cdot)$ to obtain the final score $Score(JRT)$. Most commonly used is the IR-style ranking function adopted from TF-IDF based ranking. For instance, Discover [143] uses the sum of individual TF-IDF

scores obtained via pivoted normalization weighting [211][1]:

$$\begin{aligned} Score(JRT) &= \sum_{r \in JRT} Score(r), \\ Score(r) &= \sum_{v \in r,Q} weight(v,r) * weight(v,Q), \\ weight(v,r) &= \frac{ntf}{ndl} * idf, \\ ntf &= 1 + ln(1 + ln(tf)), \\ ndl &= (1-s) + s * \frac{dl}{avgdl}, \\ idf &= ln\frac{N}{df+1}, \end{aligned} \tag{6.1}$$

where ntf is the normalized term frequency (the normalized number of occurrences of v in r), df is the document frequency (the number of r containing the term v), N is the total number of resources in the collection, idf is the inverse document frequency, dl is the length measured as the number of terms contained in r, $avgdl$ is the average length in the collection, s is a constant usually set to 0.2, and ndl is the normalized document length.

In fact, Discover [143] adopts this weighting via four different normalization methods. The goal is to obtain customized (1) idf and (2) ndl values, and to introduce (3) inter-document weight normalization as well as (4) Steiner tree size normalization. This is motivated by the fact that each column text has different vocabularies and thus shall be treated as a single collection, a Steiner tree is actually an "aggregated resource" and thus requires additional normalization beyond the resource level, and similar to document length, the size of the tree shall have an effect on relevance.

The last factor is very specific to this keyword search setting. Closely related to this Steiner tree size metric is the length of "root to matching nodes" paths [90], or "leaf to center nodes" paths [225]. All these metrics aim to capture the goal of what is known as proximity search, i.e., to minimize the distance between search terms (matching resource nodes) in the data graph. Applying this heuristic yields higher scores to those Steiner trees that are more compact. Intuitively speaking, the assumption here is that trees with more closely related matching nodes more likely capture the intended information need.

Besides IR-style scores defined for matching nodes (IR), and scores representing the distances between nodes in the result (proximity), the third category of scores commonly used is node prestige derived via PageRank [114]. Further, while most scoring functions are designed to be monotonic, examples can be constructed where the resulting ranking contradicts human perception [150]. This gives rise to non-monotonic aggregation functions [150] that however, preclude the large body of existing query processing algorithms.

[1] Although the original ranking scheme in [143] is presented using the original IR-based document-specific terminology, it is applied to structured data retrieval by considering the textual entries in the structured data as some sort of documents. We also follow the same terminology here, but distinguish that a *document* refer to textual data in a structured context.

We note that these existing ranking strategies capture ad-hoc and often debatable intuitions of what is supposed to be relevant. Traditionally, IR ranking (i.e., the probability ranking principle [236]) aims to maximize performance by ranking documents based on the posterior probability that they belong to the relevant class. That is, an explicit notion of relevance is employed, and specific approaches vary in their attempt to estimate word probabilities in this class. While the discussed TF-IDF normalizations [143] – and other existing IR-style ranking strategies [150] – intuitively make sense for the introduced examples, they lack a general account for relevance, i.e., it is not clear whether the resulting weights correspond to word probabilities in the relevant class. Also, while the distance-based heuristic used for proximity search is convenient from the computational point of view (as the keyword search problem can be reduced to the shortest-path problem [90]), it does not directly capture relevance in this sense.

An extensive evaluation of existing ranking strategies [47] suggests that no existing scheme is always best for search effectiveness (contradicting previous ad-hoc evaluations). Proximity search that incorporates node prestige tends to be slightly more effective than IR-style ranking. Overall, the authors found that previously reported results were "abnormally well", which they attribute to non-standard relevance definitions coupled with very general queries. Further, non-monotonic ranking yields no appreciable difference such that the derived recommendation is "cheap ranking schemes should be used instead of complex ones that require completely new query processing algorithms". In light of these results, we will now present a principled approach to keyword search ranking that enables the reuse of out-of-the-box techniques for result computation.

6.2 Relevance Based Ranking

In this section, we present an IR-style ranking strategy. Instead of TF-IDF ranking and the ad-hoc normalization of weights proposed previously, we employ a principled method based on the use of language models that represent documents and queries as multinomial distributions over terms. In particular, we advocate the use of RM [133], aiming to directly capture the relevance behind the user query and documents.

6.2.1 Relevance Model

As a generic framework, RM is based on the generative relevance hypothesis that assumes for a given information need, queries and documents relevant to that need can be viewed as random samples from the same underlying generative model. Formally, an RM is defined as the function

$$RM_{\mathbf{R}}(v) = P(v \mid r_1, .., r_m) = \frac{P(v, r_1, .., r_m)}{P(r_1, .., r_m)}, \tag{6.2}$$

where R captures the notion of relevance. The selection of R is highly critical for the effectiveness of the RM. Using a set F of PRF documents (i.e., documents obtained using the query Q) as an approximation of R, and assuming that query terms $q_i \in Q$ are independent, Lavrenko

and Croft defined RM as

$$RM_F(v) \approx P(v \mid Q) = \sum_{D \in F} P(D)P(v \mid D) \prod_{q_i \in Q} P(q_i \mid D). \tag{6.3}$$

Given a collection of documents C and the vocabulary of terms V, the score of a document $D \in C$ is based on its cross-entropy from the relevance model RM_F, defined as

$$H(RM_F \parallel D) = \sum_{v \in V} RM_F(v) \log P(v \mid D), \tag{6.4}$$

where $P(v \mid D)$ is defined as

$$P(v \mid D) = \lambda_D \frac{n(v, D)}{\mid d \mid} + (1 - \lambda_D)P(v \mid C). \tag{6.5}$$

Here, $n(v, D)$ is the count of the word v in the document, $\mid d \mid$ is the document length, and $P(v \mid C)$ is the background probability of v.

Intuitively speaking, relevance is modeled as a distribution over words obtained from PRF documents. This can be seen as a kind of query expansion. However, the difference is that instead of adding a few additional terms to the query, the relevance model here assigns a probability value to every term in the vocabulary V. Given the relevance and document as language models, cross-entropy is used as a similarity measure. Note that while constructing the document language model, the background collection probability is used for smoothing, which can be seen as a method for normalizing the document-specific probability of a term v.

6.2.2 Edge-specific Relevance Model

Given a user query $Q = (q_1, ..., q_m)$, we follow the RM approach to construct RM from a set of artifacts that is assumed to be close to the query. To achieve this, we use a *keyword index* over the data graph that indexes resources along with their attributes. Conceptually, this index can be seen as a query-resource map used for evaluating the multi-valued function $f : Q \rightarrow 2^{V_R}$. A standard inverted index is used for its implementation: every node $r \in V_R$ is treated as a document and document terms correspond to labels of attribute nodes $a \in \mathcal{A}(r)$. Further, we store edge labels in the index such that every term actually carries the information $(r, e, a) \in V_R \times E_A \times V_A$. After an initial run of the query Q over the keyword index, we obtain a PRF set of resources $F_R = \{r_1, ..., r_n\}$. Based on this, we construct an edge-specific relevance model (ERM) [21] for each unique attribute edge e as

$$RM^e_{F_R}(v) = \frac{\sum_{r \in F_R} P_{r \xrightarrow{e} a}(v \mid a) \prod_{i=1}^{m} P_{r \xrightarrow{*} a}(q_i \mid a)}{\sum_{r' \in F_R} \prod_{i=1}^{m} P_{r' \xrightarrow{*} a}(q_i \mid a)}, \tag{6.6}$$

where $P_{r \xrightarrow{e} a}(v \mid a)$ represents the probability of observing a word v in the edge-specific attribute a (i.e., the attribute that is connected to r over e) and $P_{r \xrightarrow{*} a}(v \mid a)$ is the probability of observing the word in all the attributes of r. In fact, $P_{r \xrightarrow{*} a}(v \mid a)$ can be rewritten in the form

$words$	RM_q^{name}	$RM_q^{character}$	RM_q^{title}	RM_q^{plot}
hepburn	0.30	0.17	0.02	0.11
holiday	0.02	0.03	0.5	0.1
audrey	0.14	0.02	0.01	0.06
katharine	0.12	0.12	0.02	0.03
princess	0.01	0.01	0.02	0.1
roman	0.01	0.01	0.26	0.03
...	...	...	...	...

Table 6.1: Example ERM for the query "Hepburn Holiday"

of $P_{r \xrightarrow{e} a}(v \mid a)$ as

$$P_{r \xrightarrow{*} a}(v \mid a) = \sum_{e \in \mathcal{E}(r) \subseteq E_A} P_{r \xrightarrow{e} a}(v \mid a) P(e \mid r), \tag{6.7}$$

where $P(e \mid r)$ denotes the weight of edge e among all the edges of r. In particular, we estimate this by considering the length of the attributes of the edge as

$$P(e \mid r) = \frac{\sum_{a \in \mathcal{A}(r,e)} |a|}{\sum_{e' \in \mathcal{E}(r) \subseteq E_A} \sum_{a' \in \mathcal{A}(r,e')} |a'|}. \tag{6.8}$$

A trivial way to estimate the edge-specific attribute probabilities, $P_{r \xrightarrow{e} a}(v \mid a)$, is to consider every attribute as a document and to use a maximum likelihood estimation, $P^{ML}_{r \xrightarrow{e} a}(v \mid a)$, which is proportional to the count of the word v in an attribute a. However, such an estimation is problematic because it would assign zero probabilities to those words not occurring in the attribute. This may result in an underestimation of probabilities of the missing words. We will discuss later in Section 6.2.4 how to address this by applying a structure-based smoothing method to $P_{r \xrightarrow{e} a}(v \mid a)$.

Example. Consider the query "Hepburn Holiday" on the example data graph in Fig. 6.1. The keyword index returns the PRF resources $F_R = \{m1, p1, p4, m2, p2m2, m3\}$, each matches one or more query keywords. Based on this, we construct an ERM as illustrated in Table 6.1. Note the probabilities of the words vary for different edges. For example, "hepburn", and "audrey" are important words for the $name$ or $plot$ attributes, whereas "holiday" is given more emphasis as a movie *title*.

Compared to RM (equation 6.3), ERM (equation 6.6) takes the specific structures of the results into account. ERM is thus a more fine-grained approximation of relevance. It can be considered as analogous to form-based keyword search on databases in which the user is required to enter different keywords to different fields of the form. By processing the PRF set F_R of an initial query run, we are able to exploit the structure of the returned resources. This for instance, may capture attribute types such as *name*, *city*, *comment* etc. without any user intervention. Structures that can be incorporated might emerge from different resources with varying types of attributes.

6.2.3 Edge-Specific Resource Model

The basic retrieval unit here is a resource (a tuple) since the final results are obtained by combining resources (joining tuples) into an aggregated result. In order to calculate a final aggregated score and utilize an efficient top-k algorithm, we need to be able to assign each resource a score using our ranking mechanism. In order to achieve that, we construct a resource model that assigns probabilities to the word in the vocabulary w.r.t. a given resource r. This is similar to the document model in standard IR approaches. The difference is that also structure information (the edges) is exploited for the estimation of word probabilities. We define an edge-specific resource model as

$$RM_r^e(v) = (1 - \lambda_r) P_{r \xrightarrow{e} a}(v \mid a) + \lambda_r P_{r \xrightarrow{*} a}(v \mid a). \tag{6.9}$$

Here, the probability of a word for a specific edge e of r is approximated as smoothed probabilities controlled by λ_r. Actually, the parameter λ_r is critical for our ranking to indicate the weight of the edge-specific attributes. A small λ_r means less smoothing and more emphasis on the edge-specific attribute (more emphasis on the terms of the attribute), and more emphasis on the terms of the entire resource (terms in all attributes) otherwise.

The score of a resource is then calculated based on the cross-entropy between the relevance model obtained for the query (RM_{F_R}) and the resource model (RM_r) as

$$Score(r) \quad = \quad \sum_{e \in E} \alpha_e \sum_{v \in V} RM_{F_R}^e(v) \log RM_r^e(v), \tag{6.10}$$

where α_e is a parameter that allows us to control the importance of different edges in the scoring.

6.2.4 Smoothing

Note that the core probability of both RM_{F_R} and RM_r is $P_{r \xrightarrow{e} a}(v \mid a)$. Smoothing is a well-known technique to address data spareness and improve the accuracy of language models, which we apply to obtain a better estimate for $P_{r \xrightarrow{e} a}(v \mid a)$. Traditional smoothing methods mainly use the global collection probabilities. In our case, the global collection simply comprises all attributes in the database. One deficiency of such a global smoothing is that it does not reflect the structure information that is available in this case. As an example, the smoothing probability of the word "Washington" in the *name* attribute "Denzel Washington" should not consider the probability of that word in the *city* attribute, since "Washington" can be highly frequent as a *city* but not as a *name*.

A general framework for smoothing is presented in [161], which consists of the tuple $\{S, f_u, \tilde{f}_u, w(u), w(u,v)\}$, where $S = (V_S, E_S)$ is the smoothing graph, f_u is the smoothed probabilities of the vertexes $u \in V_S$, $\tilde{f}_u$ is the non-smoothed (initial) probabilities of $u \in V_S$, $w(u)$ captures the weights indicating the importances of $u \in V_S$, and $w(u,v)$ stands for the weights of the edges $(u,v) \in E_S$.

Different instantiations of this framework result in different smoothing strategies. We adopt this by using the data graph G for S. The goal is to estimate the edge-specific attribute prob-

abilities $f_u = P_{r \xrightarrow{e} a}(v \mid a)$. While $w(u)$ is set to be uniform, $\tilde{f}_u$ is computed via maximum likelihood as $P^{ML}_{r \xrightarrow{e} a}(v \mid a) = \frac{n(v,a)}{|a|}$, where the nominator is the count of v in a and the denominator denotes the length of a. Then, we obtain the smoothed probability

$$P_{r \xrightarrow{e} a}(v \mid a) = (1 - \lambda_a) P^{ML}_{r \xrightarrow{e} a}(v \mid a) + \lambda_a \sum_{a' \in \mathcal{N}(a)} \frac{w(a, a')}{Deg(a)} P_{r \xrightarrow{e} a}(v \mid a')$$

where $\mathcal{N}(a)$ is the neighborhood of a and $Deg(a) = \sum_{a' \in \mathcal{N}(a)} w(a, a')$. This notion of neighborhood in the case of structured data is illustrated in Fig. 6.2. More formally, given $a \in \mathcal{A}(r, e)$, a' is said to be in the neighborhood of a, $a' \in \mathcal{N}(a)$, iff:

- a and a' share the same resources, i.e., $a' \in \mathcal{A}(r, e')$ for all edges $e' \in \mathcal{E}(r) - \{e\}$ (*Type 1*),
- resources of a and a' are of the same type, i.e., $a' \in \mathcal{A}(r', e)$ where $(r, type, c), (r', type, c) \in E$ (*Type 2*),
- resources of a and a' are connected over a foreign key, i.e., $a' \in \mathcal{A}(r', e')$, where $(r, e'', r') \in E_F$ (*Type 3*).

The weight of these three types of attributes ($i \in \{1, 2, 3\}$) are determined separately as

$$w_i(a, a') = \frac{\gamma_i + \sigma(sim(a, a'))}{\gamma_i + 1} \quad \forall i \in \{1, 2, 3\} \tag{6.11}$$

Here, $\sigma(.)$ is a sigmoid function, $sim(a, a')$ is the content-based similarity of the two attributes, and $\gamma_1, \gamma_2, \gamma_3$ are the control parameters for each type. A commonly used instantiation of $sim(a, a')$ is the cosine similarity that we also employ in our experiments. We use control parameters to control the importance of similarity that we determine experimentally in subsection 6.4.5. Control parameters, $\gamma_1, \gamma_2, \gamma_3$, are set experimentally indicating the importance of each type for smoothing. In our experiments, we evaluate their robustness by assigning different values to these parameters in the range of $\gamma_i \in [0, 1]$. For low values of γ_i (e.g. close to 0), the smoothing is mostly controlled by the similarity of two attributes. This works well for *Type 2* attributes where the number of attributes are relatively large and extensive smoothing can harm the performance. In addition, for high values (around 1), more smoothing is performed regardless of similarity. This indicates, in our running example, that in the probability of *name* attribute of resource *p1* (e.g. *Audrey Hepburn*), the words such as *ixelles*, *belgium* (*Type 1*), *princess*, *ann* (*Type 2*) are assigned with non-zero probabilities, as they are considered to be relevant to that attribute. Fig. 6.2 illustrates these three types of smoothing on the local structure of an attribute a.

6.2.5 Ranking

Ranking in this setting is concerned with JRT, which is a joined set of resources. We provided equation 6.10 for ranking individual resources. For ranking a set of resources, the same scoring

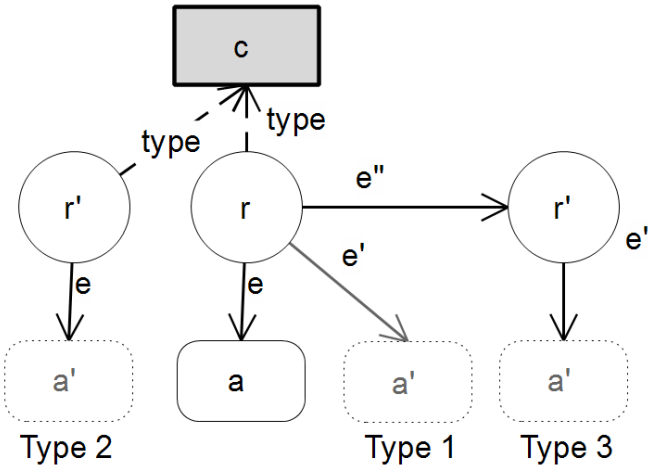


Figure 6.2: The neighborhood of attribute a.

function is applied. However, in this case, $RM_r^e(v)$ in equation 6.10 actually stands for the set of resources $r_m \in JRT$, defined as

$$RM_{JRT}^e(v) = \sqrt[m]{RM_{r_1}^e(v) \dots RM_{r_m}^e(v)}. \tag{6.12}$$

A critical issue for the efficient computation of results is that such an aggregated scoring function, which combines scores from more than two resources, is monotonic:

Definition 11 (Score Monotonicity) *Let Q be the query, and $JRT = \{r_1, \dots, r_n\}$ and $JRT' = \{r'_1, \dots, r'_n\}$ be two results to Q. An aggregated scoring function is* monotonic *if it satisfies the following condition: if $Score(r_i) \leq Score(r'_i)$ for all $1 \leq i \leq n$, then $Score(JRT) \leq Score(JRT')$.*

We now show that the proposed ranking satisfies the following theorem:

Theorem 2 *The scoring function defined in equation 6.10 is monotonic with respect to the aggregation of resources defined in Equation 6.12.*

Proof Sketch:

$$
\begin{aligned}
& Score(JRT) \\
= & \sum_{e \in E} \alpha_e \sum_v RM^e_{F_R}(v) \log \sqrt[m]{RM^e_{r_1}(v) \dots RM^e_{r_m}(v)} \\
= & \frac{1}{m} \sum_{e \in E} \alpha_e \sum_v RM^e_{F_R}(v) \left[\log RM^e_{r_1}(v) + \ldots + \log RM^e_{r_m}(v) \right] \\
= & \frac{1}{m} \sum_{e \in E} \alpha_e \sum_v RM^e_{F_R}(v) \log RM^e_{r_1}(v) + \ldots + \\
& \sum_{e \in E} \alpha_e \sum_v RM^e_{F_R}(v) \log RM^e_{r_m}(v) \\
= & \frac{1}{m} (Score(r_1) + \ldots + Score(r_m))
\end{aligned}
$$

Clearly, $\frac{1}{m}(Score(r_1) + \ldots + Score(r_m))$ is monotonic.

Example. Continuing with our example query "Hepburn Holiday", the keyword search algorithm firstly obtains the matching resources $F_R = \{m1, p1, p4, m2, p2m2, m3\}$. Based on this, several JRTs are constructed in a bottom-up top-k fashion. They are found based on exploring foreign key paths between these elements. Table 6.2 shows two of these paths (between p_1 and m_1, and between p_2m_2 and m_2) and two example JRTs resulting from these. During this computation, the scoring function defined in equation 6.10 is used both to compute the individual scores *Score(r)* (e.g. of p_1 and m_1) and *OurScore*, which indicates the combined score in Table 6.2 (e.g. of $\{p_2, m_2\} \rightarrow m_2$). Note the difference of our ranking scheme and the ones using proximity and TF-IDF. Proximity alone would assign $\{p_2, m_2\} \rightarrow m_2$ the highest score simply because it is the most compact answer. The two results tie in terms of TF-IDF scores. Our approach ranks $p_1 \leftarrow \{p_1, m_1\} \rightarrow m_1$ first, recognizing that "Hepburn" is an important term for $name$ and "Holiday" is an important term for the $title$ of movies.

JRT	$r \in JRT$	$tf_{hepburn}$	$tf_{holiday}$	$Score(r)$	*TF-IDF*	*Proximity*	*ERM*
$p_1 \leftarrow \{p_1, m_1\} \rightarrow m_1$	p_1	1	0	0.36	3.0	3	0.22
	$\{p_1, m_1\}$	0	0	0.10			
	m_1	1	1	0.18			
$\{p_2, m_2\} \rightarrow m_2$	$\{p_2, m_2\}$	1	0	0.12	3.0	2	0.19
	m_2	1	1	0.26			

Table 6.2: Comparison of TF-IDF, proximity and our ERM scores for the query "Hepburn Holiday".

6.3 Query Processing with Relevance Models

In this section, we present how our proposed ranking model can be used within a top-k query processing algorithm in order to obtain the best relevant results for a user's query. Instead of directly computing the results from the keyword elements, we assume that a number of structured queries are pregiven. Similar to candidate networks in the schema-based approaches,

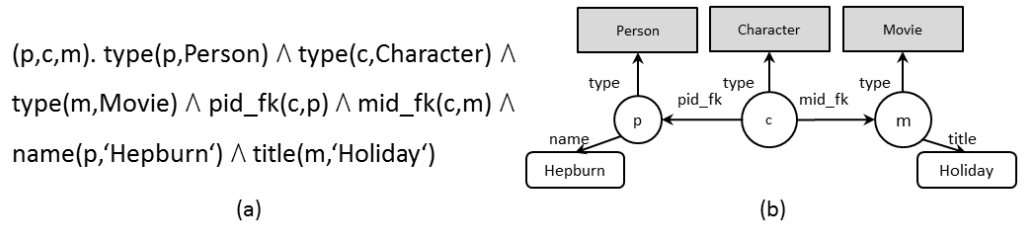


Figure 6.3: a) An example conjunctive query and b) its corresponding query graph representation.

in the previous work [225] a number of conjunctive queries are generated by exploring the so-called summary graph. In fact, the work presented in [225] also employs a TF-IDF and proximity based ranking to calculate the top-k queries. However, we consider that ranking in this case is not as important as ranking the final results because the summary graph is relatively small in comparison with the size of actual data and only a small number of queries are generated which makes relevance based ranking irrelevant in that case. Instead we consider that given those generated queries, called *conjunctive queries*, we complement the previous work in the sense that it considers the query processing as a black-box and assumes an RDF query processing, while we explicitly calculate top-k results with a relevance-based ranking. The input to our algorithm are conjunctive queries which are defined as follows in [225]:

Definition 12 *A conjunctive query is an expression of the form* $(x_1, ..., x_k).A_1 \wedge ... \wedge A_r$*, where* $x_1, ..., x_k$ *are called query variables (those which will be bound to yield an answer), and* $A_1, ..., A_r$ *are query atoms. These atoms are of the form* $P(v1, v2)$*, where* P *is called predicate, v1, v2 are either variables or constants.*

Example. Fig. 6.3 shows an example conjunctive query with three variables and its corresponding graph representation. In the atoms, two query variables (e.g. p, m) are bound to the keyword elements (e.g. 'Hepburn', 'Holiday'), while the other (e.g. c) is unbound. Variables are also associated with the *type* information indicating their classes.

We make a distinction between the bound and unbound query variables denoting them as $nonFree(CQ)$ and $free(CQ)$ of a conjunctive query CQ, respectively. Then, our aim is to bind the resources from the data graph to those variables which satisfy the expression of the conjunctive query given as atoms. We use a special index structure apart from the keyword index described above and our goal is to minimize the number of index accesses necessary to compute the relevant results. There are two types of accesses in the literature of top-k query processing [101]: *Random access (RA)* and *sorted access (SA)*. In an RA, we know the value for at least one of the variables and using the predicates in the query we ask for all the other resources that satisfy the predicate in the data graph. An SA, on the other hand, returns the resource with highest score that has not been accessed so far from a list of resources that satisfy the conditions of the query.

In the state-of-the-art of top-k query processing, an SA is only possible if a relation is indexed according to a specific attribute value – e.g. all the tuples in the *Person* table are sorted according the *age attribute*. Since we calculate a query-dependent score for every resource

using the relevance model, such a materialization is not possible at indexing time. However, at the query time, the resources to be bound to non-free query variables (e.g. variables with keyword variables) are retrieved by querying the specific attributes of the resources with the given keyword. For example, for the query given in Fig. 6.3 we first retrieve all *Person* resources with the query 'Hepburn' in their *name* attribute and all *Movie* resources with the query 'Holiday' in their *title* attribute. Then each set of resources is materialized (i.e. sorted) according to their relevance score calculated based on the Eq. 6.10. Fig. 6.4 shows three sets of resources corresponding to the query variables of the conjunctive query in Fig. 6.3. For the variables p and m, both SA and RA are possible as the resources are sorted after an initial retrieval. This assumption cannot be made for the variable c which is free and the set of resources initially corresponds to all *Character* entities in the data graph which is infeasible to score and sort. That's why a RA strategy is more suitable for the set of resources of a free variable.

We denote the set of resources that can be bound to a query variable x_i as R_i. Then for each query variable, we specify an upper bound score, $uscore(x_i)$. If x_i is non-free, the upper bound score maintains a resource-specific score $uscore(x_i) = score(r_i)$, where r_i is the last accessed resource in R_i. If x_i is a free variable, the upper bound score cannot be explicitly calculated since the scores of all possible resources in its set (i.e. R_i) is never completely calculated and it cannot be determined which the most relevant resource is. To approximate this upper bound score, we query R_i of a free query variable x_i with an expanded query from the edge-specific relevance model (i.e. using the top words of relevance model) in order to retrieve the resources relevant to the query. Then we set the upper bound score of the variable to the score of the top resource. For example, we set the score of variable c to the score of resource $p2m2$ (i.e. $uscore(c) = score(p2m2)$) since it is the closest resource to the query according to its attributes. Now, we can define *result candidate* which partially or completely specify a query result as follows:

Definition 13 *Let $G = (V, E)$ be a data graph and CQ be a conjunctive query where $x_1, ..., x_k$ are query variables. A result candidate c is vector of the form $< r_{x_1}, ..., r_{x_k}, score(r_{x_1}), ..., score(r_{x_k}) >$ where:*

- *each r_{x_i} is either a resource r bound to variable x_i (i.e. $r \in V_R$) or $*$ (i.e. unbound),*
- *for each r_{x_i}, there is a relation edge $type(r_{x_i}, C)$ in G for some class C,*
- *$score(r_{x_i}) = score(r)$ if $r_{x_i} = r$ or $score(r_{x_i}) = uscore(x_i)$ if $r_{x_i} = *$*

A result candidate is complete if for all r_{x_i}, $r_{x_i} \neq *$.

Note that a result candidate does not require all variables to be bound to a particular resource and allows r_{x_i} to be unbound a new symbol $*$. A result candidate with all its variables unbound is denoted as c_*. We also use a binding operator over result candidates, denoted by $c' = (c, x_i \rightarrow r)$, to indicate that c' is created from c by binding r_{x_i} with the resource r. For example, consider the set of resources in Fig. 6.4. A result candidate c_1 can be created from c_* as follows:

$$c_1 = (c_*, p \rightarrow p1) = (< *, *, *, 0.20, 0.11, 0.19 >, p \rightarrow p1) =< p1, *, *, 0.20, 0.11, 0.19 >$$

Based on the definition of the result candidate, an upper bound score can be calculated as follows:

$$uscore(c) = \sum_{r_{x_i} \in c, r_{x_i} = r} score(r) + \sum_{r_{x_i} \in c, r_{x_i} = *} uscore(x_i) \tag{6.13}$$

Algorithm 4: Top-k Algorithm

Input: CQ // Conjunctive query
Input: $R_i, .., R_k$ // The sets of resources
Input: K // The number of results
$O = \emptyset$ // Sorted set of results
C // Priority queue of result candidates
$\tau = \infty$ // Threshold
$C.push(c_*)$ // Priority queue of result candidates
while ($\mid O \mid < K$ **and** C is not empty) **do**
  $c \leftarrow C.pop()$
  if $c\ is\ complete$ **then**
    $O = O \cup \{c\}$
    // check if candidate has better score,if not do not process it
  else if $uscore(c) < \tau$ **then**
    $C.push(c)$
    // create a new candidate with the next sorted access
    $i = PS.chooseVariable()$
    $r \leftarrow$ next resource from SA on R_i
    $c' \leftarrow (c_*, x_i \rightarrow r)$
    $C.push(c')$
    $\tau \leftarrow BS.updateThreshold()$
  else
    // candidate has more potential than threshold
    choose last bound variable i of c
    $j \leftarrow$ choose adjacent variable to i in CQ
    $E_j \leftarrow \{\cup r_j\}$ where $r_j \leftarrow$ RA on R_j
    for all r_j in E_j **do**
      $c' \leftarrow (c, x_j \rightarrow r_j)$
      $C.push(c')$
    end for
  end if
end while
Output: O

The Algorithm 4 shows the pseudocode of the top-k algorithm which takes a conjunctive query, the sets of resources of query variables, and K (the number of results). We define an empty set of sorted results which is a placeholder for the final results and a priority queue C of result candidates which is sorted according to the upper bound score. Initially, there is only one

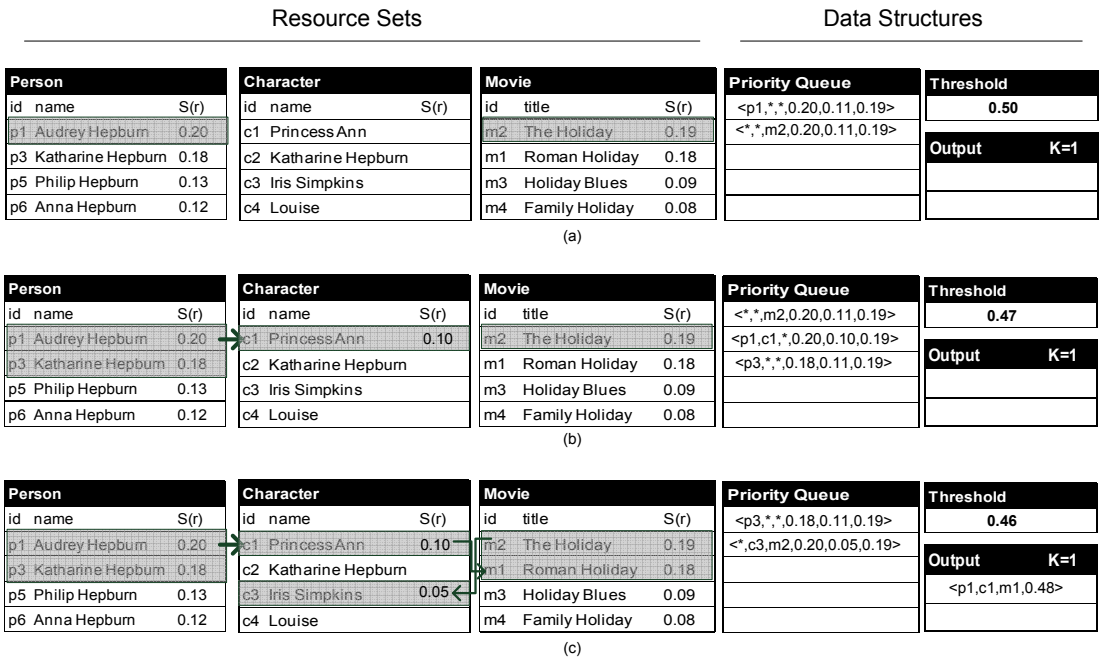


Figure 6.4: Illustration of the Algorithm 4 for the example query: a) Status after two SA to Person and Movie resource sets (grayed resources are accessed), b) after the first RA to Character resource set, and c) Algorithm terminates after a complete result candidate with a score larger than the threshold is found as output.

result candidate, i.e. c_*, whose variables are unbound. The threshold τ is initialized to infinity. The algorithm iterates in a loop until top-K results are found or there is no result candidate in the queue to be processed.

In each iteration, it picks the best candidate from the queue C and adds it to the output set if it is complete (i.e. first *if* condition). If not, it checks whether the upper bound score of the candidate is less than the threshold (second condition). If so, it does not process the candidate, adds it back to the queue. Then we pick a non-free variable by the function $PS.chooseVariable()$ which returns the next non-free variable in a round-robin fashion. In fact, here we adopt the formalism of the *Pull Bound Rank Join* template [205] with two deterministic components, namely, a pulling strategy (PS), and a bounding scheme (BS). At each iteration, PS chooses a variable x_i whose resource set is to be accessed with a SA. The original study in [205] showed a round-robin pulling strategy to be instance-optimal, picking an equal number of resources from the resource sets of non-free variables. For example, if PS chooses variable p in our example, then in the next iteration it picks variable m and thus the sorted access performed on both resource sets are kept in balance. Based on the selected variable, a SA is performed on its resource set to obtain the next resource and a new candidate is created by applying the binding operator on c_*. Broadly speaking, in this part of the algorithm we progress in a vertical direction which means that we get a resource from the next sorted set of resources and create a new result candidate result with that resource. This new candidate is unbound in all variables except the one chosen by the pulling strategy. Then a bounding scheme (BS) calculates the

new threshold indicating the best possible score that can be achieved by using the remaining resources.

When the upper bound score of the candidate is greater than the threshold τ, then we assume that the candidate has more potential than the remaining resources to be bound and we continue to bind more variables the resources using a RA strategy. First, it chooses the last bound variable and then retrieves another variable adjacent to it based on the conjunctive query CQ. In particular, the algorithm finds all the relationships predicates (e.g. $P(x_i, x_j)$ or $P(x_j, x_i)$) where the variable is a parameter, and instantiates it with the resource r bound to the variable x_i in the result candidate and retrieves the other resource r_j that satisfies the predicate according to the data graph. In fact, there can be more than one resource r_j so we take a union of those resources (i.e. E_J). Then the algorithm applies the binding operator on the adjacent variable to bind it with every new resource in E_j and for each one of those a new result candidate is added to the priority queue. The algorithm proceeds until K results are found and returns the output set O as a final result.

6.4 Experiments

Due to ad-hoc style evaluation, results of previous keyword search ranking were found to be abnormally well. Aiming at making results more conclusive and comparable, we exactly follow the framework for evaluating keyword search ranking strategies proposed recently [47]. For completeness, we will now summarize the data, queries and experimental settings proposed for this kind of evaluation.

6.4.1 Datasets

We use three different datasets, two of which are derived from two popular and large websites (Wikipedia and IMDb). IMDb data proposed for keyword search evaluation [47] is actually a subset from the original IMDb database, containing information about more than 180.000 movies. This is because several keyword search systems require data to be completely loaded into memory, and thus cannot scale to large datasets. The complete Wikipedia contains more than 3 milion articles. For the same reason, only a selection of more than 5500 articles was used. All tables unrelated to articles or users were excluded, and the PageLinks table was augmented with an additional foreign key to explicitly indicate referenced pages. The third dataset is MONDIAL, which represents the counterpoint to the other two because compared to them, it is smaller in size but more complex in terms of structure. It captures geographical and demographic information from the CIA World Factbook, the *International Atlas*, the TERRA database, and other web sources. The relational version for PostgreSQL was downloaded. Wikipedia contains more text than IMDb, which in turn, has more text than MONDIAL. In other words, while MONDIAL can be seen as a structured database, Wikipedia is rather a structured document collection.

6.4.2 Queries

Clearly, performance may vary widely across information needs for the same document collection. Traditionally, fifty information needs are regarded as the minimum for evaluating retrieval systems. Accordingly, 50 queries were proposed in [47] for each dataset to cover distinct information needs that vary in complexity. The maximum number of terms in any query is 7 and the average number of terms is 2.91. Among these different queries, there are two important types that were investigated in detail. There are the (1) *"TREC-style" queries* which are Wikipedia topics most similar to those encountered at TREC. Terms of these queries are present in many articles, yet most of those articles are not relevant. For instance, the query "smallpox vaccination" asks for information about the one who discovered/invented the smallpox vaccine. Finding out which articles are relevant and how they vary in the degree of relevance is the problem here. (2) The other type comprises *"single-resource"* queries which ask for exactly one resource. It has been reported that this type of queries constitute the most common type of query posed to existing search engines. For instance the query "rocky stallone" asks for the film in which the actor Sylvester Stallone plays the character Rocky.

6.4.3 Measuring the Degree of Relevance

Relevance is assessed based on the specified information needs. Binary relevance judgments are used such that all relevant results are equally desirable. Firstly, SQL queries are constructed to capture the information needs. Then, one single expert judges all results that can be obtained for these queries. Three metrics are employed to compare systems: (1) We use the number of *top-1 relevant results*, which is defined as the number of queries for which the first result is relevant. (2) *Reciprocal rank* is simply the reciprocal of the highest ranked relevant result for a given query. This measure aims to capture the quality of the top-ranked results. As the third metric, we use *average precision*, which also takes the order into account. For a query, it is defined as the average of the precision values calculated after each relevant result is retrieved (and assigning a precision of 0.0 to any relevant results not retrieved). Mean average precision (MAP) averages this single value across queries to obtain one single measure across different levels of recall and different types of information needs. These metrics are calculated based on the top-50 results returned by each system.

Table 6.3 provides a summary of the statistics of the data, the query workload, and the relevant results for each dataset.

6.4.4 Baseline Systems

We compare the results against systems, which have shown to provide best results in the previous evaluation [47]. These are *BANKS* [2] and *Bidirectional* [114], which represent the category of ranking strategies that make use of proximity and node prestige. The IR-style (TF-IDF based) ranking is implemented by *Efficient* [97], *SPARK* [150] and Covered Density [47] (*CD*). *SPARK* is the one that features a non-monotonic function. It has been found [47] that for single-resource queries, proximity in combination with node prestige perform well (compared to IR-style ranking). This is because this scheme prefers the smallest result that satisfies the query

Dataset	Size	Rel.	Tuples	|Q|	$\|\bar{q}\|$	$\|\bar{R}\|$
Mondial	9	28	17,115	50	2.04	5.90
IMDb	516	6	1,673,074	50	3.88	4.32
Wikipedia	550	6	206,318	50	2.66	3.26

Table 6.3: Characteristics of the three datasets and query workload. Size in MB, number of relations and tuples, total number of queries |Q|, average number of terms per query $|\bar{q}|$, and average number of relevant results per query $|\bar{R}|$.

(i.e., it prefers less complex JRTs referring to only one single resource) while IR-style ranking scheme prefers larger results that contain additional instances of the search terms. *BANKS* and the like are the best ones on the Mondial dataset, outperform the IR approaches on the IMDb dataset, and tie for the second most effective on Wikipedia. While this type of systems also provides reasonable effectiveness for the TREC-style queries, they are outperformed by systems implementing IR-style ranking. In particular, *Efficient* shows the best MAP result among all systems – for the set of Wikipedia topics used in the experiment. Further, SPARK reported results that were not supported by previous findings, suggesting that the use of non-monotonic ranking function may not be beneficial after all.

6.4.5 Results

The goal of the experiment is to find out how the proposed ranking (*RM-S*) compares to the best systems on TREC-style as well as single-resource queries. We will firstly discuss the overall results, and then investigate these two types of queries in more detail.

6.4.5.1 Overall Results

Figure 6.5 summarizes the overall effectiveness of each system across different information needs in terms of MAP values. We see that the overall effectiveness varies considerably across the three datasets. On average, the best three systems are *Bidirectional*, *CD* and *RM-S*. In particular, our proposed ranking *RM-S* shows very convincing results. It consistently outperforms all systems across datasets. MAP values are above 0.78 and for the MONDIAL dataset, it is even over 0.9. For the IMDb dataset, MAP is close to 0.3 higher then the second best system. These results are very encouraging, given the experiments have been conducted in a standardized way using real world datasets and a large set of queries. Also very important for practical purposes is the fact that the best systems including *RM-S* do not require non-monotonic ranking (as advocated by SPARK), and thus can be used in combination with state-of-the-art approaches for the efficient computation of keyword search results.

6.4.5.2 Single-Resource Queries

Figure 6.6 shows the mean reciprocal rank for each system for queries where exactly one resource is relevant. Performance of systems vary for this type of queries. As already reported

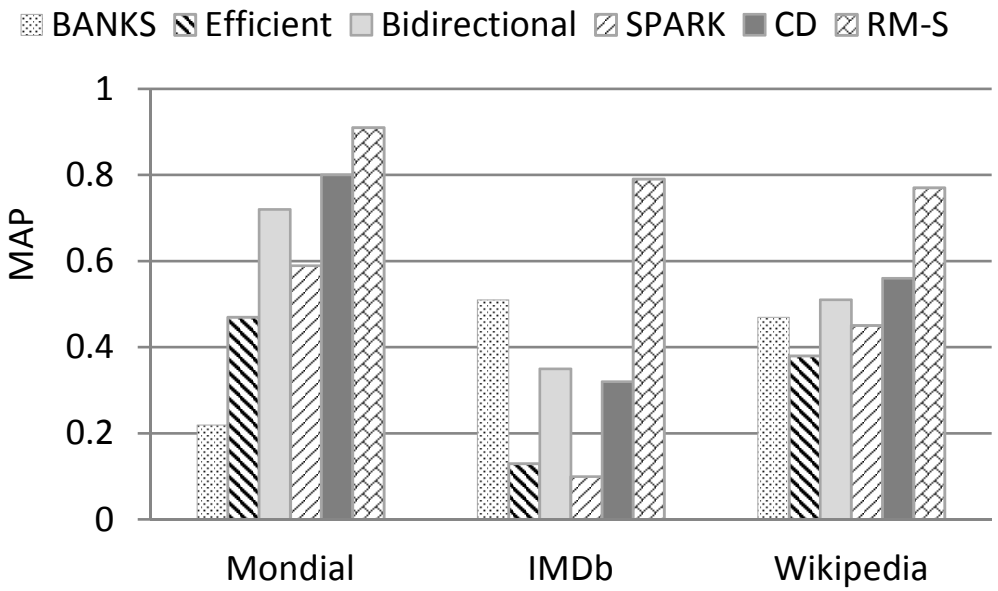


Figure 6.5: MAP across systems and datasets.

in the previous study, *Bidirectional* and *BANKS* perform relatively well, outperforming the standard TF-IDF based systems such as *Efficient* and *SPARK* on average. *BANKS* achieves poor performance on the MONDIAL dataset but very good performance on the IMDB dataset, while *Bidirectional* and *CD* exhibit more consistent performance. Our approach clearly shows best performance. It outperforms other approaches across all datasets, with mean reciprocal rank values being consistently above 0.91. We found out that TF-IDF based ranking tends to prefer complex results, which contain a large number of mentions of query terms. Thus, there are high rank results, which contain a large number of resources (each possibly containing mentions of several query terms). These results are however not relevant in this case because the queries target a single resource. In particular, the JRT size normalization (inspired by the document length normalization used in TF-IDF based ranking) [143] introduced to address this issue seems to be not as effective[1] as the more aggressive proximity-based scheme [2], which simply focuses on minimizing the tree size and finding compact results.

In light of these arguments, it seems surprising that *RM-S*, which does not incorporate this tree-size heuristic at all, achieves best results even in this case. Based on the query and the PRF results, it constructs a model of relevance and performs ranking entirely based on this model. Thus, while *RM-S* can accommodate for and exploit the possibly varying structures of the PRF results, it does not directly assume that more compact results are necessary better. We believe the key here is that *RM-S* is able to make better estimates of the structure (i.e., the set of

[1]The performance of [143] is not shown because it is similar to *Efficient* in concept but worse in performance. It is also based on TF-IDF but employes additional normalization strategies to accommodate for differences in the keyword search setting.

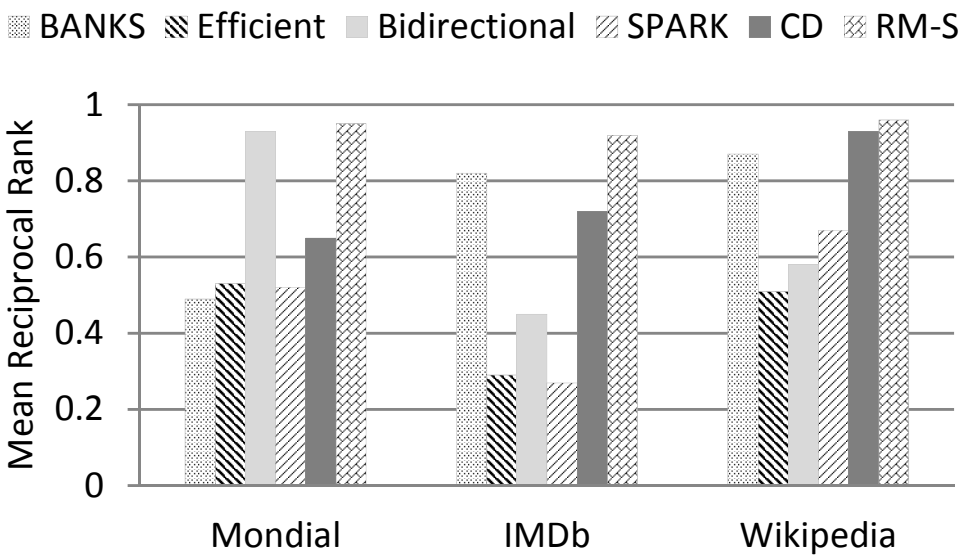


Figure 6.6: Reciprocal rank for single-resource queries.

attributes) and the content (i.e., the terms) that make up relevant results (independent of their size). For instance for the IMDb query "rocky stallone", TF-IDF based ranking finds many movies but the one with Silvester Stallone as actor and Rocky as character is not the highest ranked one because the term "rocky" and "stallone" appear more frequently in the other movies. Also here, proximity based ranking fails because there is one movie with a character name that matches "rocky stallone". This one is ranked best because its JRT size is smaller compared to the JRT of the result with Rocky as character and Stallone as actor. In this case, *RM-S* is able to find PRF results where "stallone" appears as term in the attribute $actor$ and "rocky" appears in the attribute $character$. Based on the resulting ERM, it successfully makes the guess that relevant results should have the attribute $actor$ with terms such as "stallone" and "sylvester", and the attribute $character$ with terms like "rocky" and "balboa".

6.4.5.3 TREC Queries

The results for this type of queries are shown in Figure 6.7 and Figure 6.8. While Figure 6.8 shows the overall performance, Figure 6.7 provides a breakdown into different levels of precision and recall. These results suggest that ranking schemes based on proximity and prestige do not perform well. *BANKS* is the best one in this category with precision close to 0.9 at the lowest recall level. Its precision however drops precipitously at higher recall levels. Expectedly, TF-IDF ranking performs better, with *Efficient* being the one with best performance in terms of MAP, and most stable performance across the entire precision-recall curve. In this category, *SPARK* performs best at low levels of recall. However, its precision drops sharply as recall

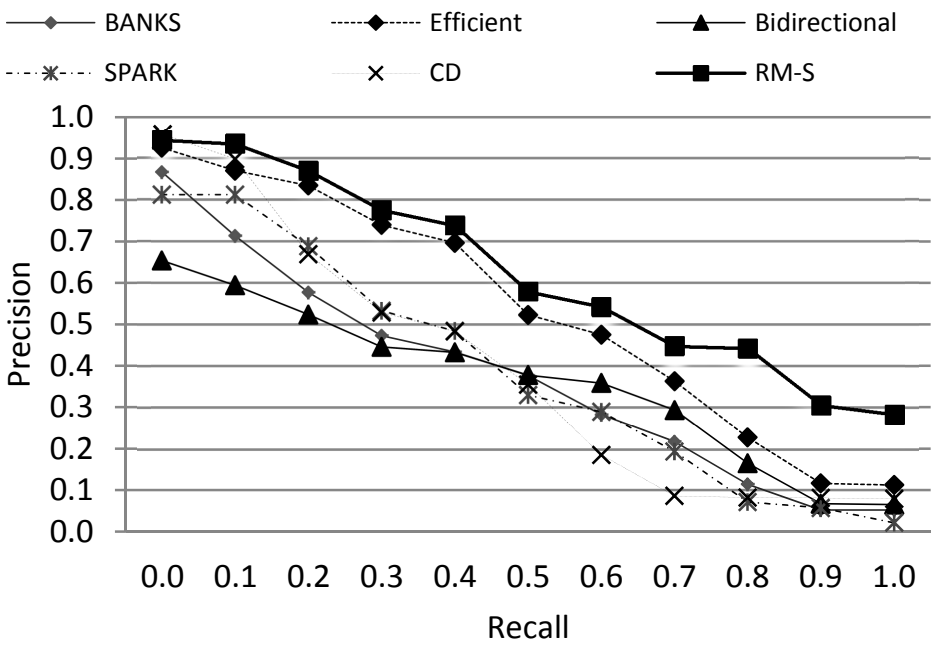


Figure 6.7: Precision-recall for TREC-style queries on Wikipedia.

increases over 0.1.

Also for this type of queries, our approach yields good results, providing best MAP (0.62) and stable performance over the entire precision-recall curve. It closely matches the performance of *SPARK* at the lowest level of recall, and consistently outperforms all approaches at higher recall levels. Up to recall level of 0.7, its performance is similar to the one of *Efficient*. It however provides much more stable performance at recall levels above that. In fact, we expect our approach to provide better recall (at the same level of precision) because using the relevance model is conceptually similar to query expansion. Terms and attributes, which make up relevant results are not limited to the ones specified in the query. Thus, relevant results can be determined even when they do exactly contain the query terms. For the query "smallpox vaccination" for instance, the resulting relevance model also contains terms such as "louis", "pasteur", "1794" and "virus". Surprising is however the fact that this does not come at the expense of precision. We believe that this is due to the fine-grained structure of the relevance model we employ, which reduces noises in the query expansion process. For instance, results are only deemed relevant when they contain the term "louis" in the attribute $inventor$.

6.4.5.4 Parameters

There is a number of parameters that we set experimentally based on the effectiveness of the ranking results. In particular, the PRF set of documents is $\mid F_R \mid \in \{5, 10, 25, 50, 75\}$, and the smoothing parameters for the edge-specific attribute properties $P_{r \xrightarrow{e}_{a}}(v \mid a)$ are $\lambda_r \in \{0.1, 0.2, ..., 0.9\}$ and $\lambda_a \in \{0.1, 0.2, ..., 0.9\}$. We sweep over values for finding the best

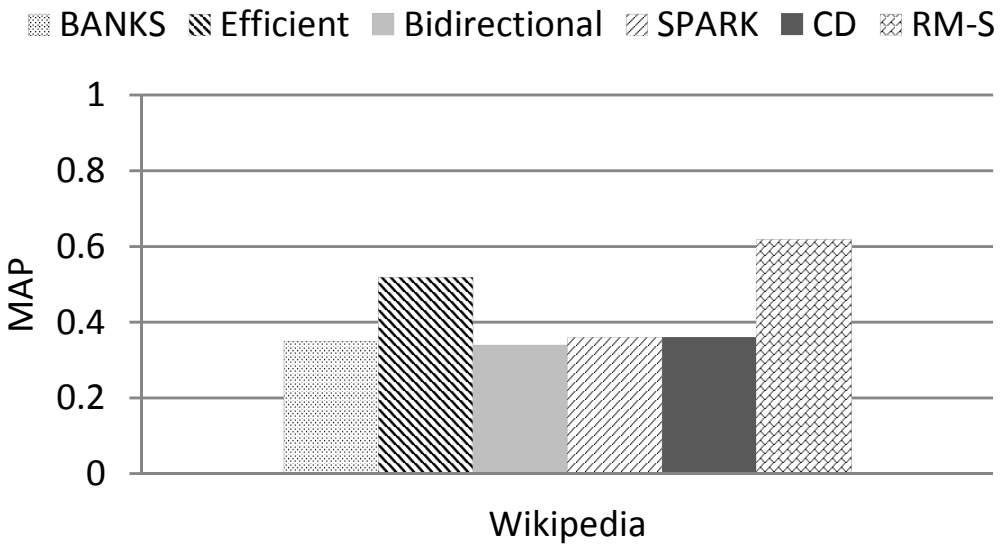


Figure 6.8: MAP for TREC-style queries on Wikipedia.

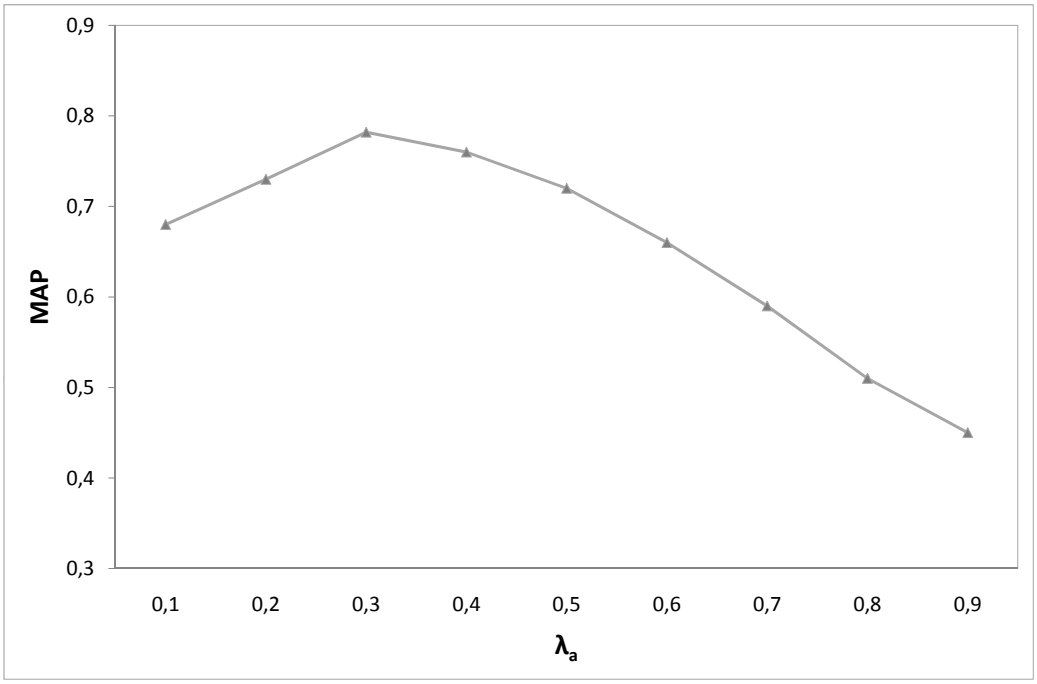


Figure 6.9: Sensitivity to smoothing interpolation parameter λ_a on Wikipedia.

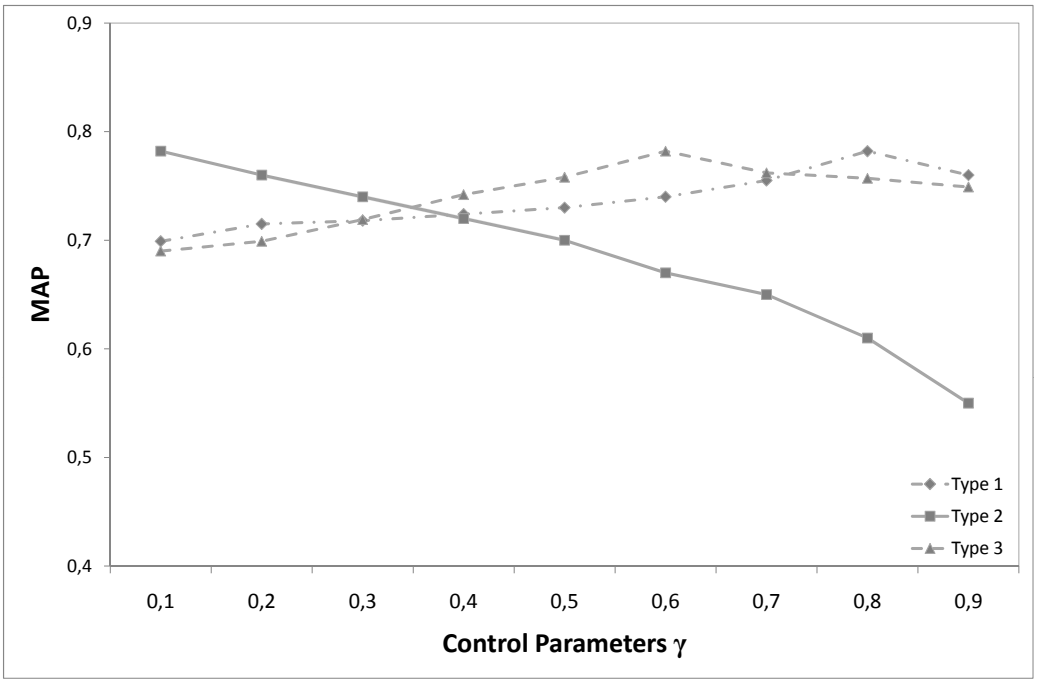


Figure 6.10: Sensitivity to control parameters of smoothing on Wikipedia ($\lambda_a = 0.3$).

configuration of these parameters. Further, α_e is set to be uniform in all the experiments.

We analyze the sensitivity of MAP w.r.t. the smoothing parameter λ_a and control parameters $\gamma_1, \gamma_2, \gamma_3$ of Equation 6.11. We first fix the control parameters and show in Figure 6.9 how MAP changes according to the value of λ_a for the Wikipedia dataset. We see that MAP values are relatively high at low levels of smoothing, i.e., are best for λ_a in the range between 0.2 and 0.5. This means that while smoothing based on the structure of the data helps to improve performance, it should not be overemphasized. For a fixed $\lambda_a = 0.3$, we investigate the sensitivity of each control parameter by fixing the other two (based on best performance such that fixed values are $\gamma 1 = 0.8, \gamma 2 = 0.1$ and $\gamma 3 = 0.6$). As illustrated in Fig. 6.10, best performance can be achieved when more emphasis is put on Type 1 and Type 3 and less on Type 2.

6.5 Discussion

Finding and ranking relevant resources is the core problem in the Information Retrieval community, for which different approaches have been investigated. The model we use here originates from the concept of language models [179], which have been proposed for modeling resources and queries as multinomial distributions over vocabulary terms, and for ranking based on the distance of the two models (e.g. using KL-divergence [256] or cross entropy [133] as measures for distance). More precisely, the foundation of our work is established by Lavrenko et al., who propose relevance-based language models to directly capture the relevance behind document and queries. Further, the structure of results as well as queries have been incorpo-

rated into language models. For instance, combinations of language models constructed from fields of documents have been proposed for structured document retrieval [257], and combinations of attribute-level language models have been employed for the retrieval of complex Web objects [172]. Also, structure information has been exploited for constructing structured relevance models [136] (SRM). This is the one mostly related to ERM. The difference is that while SRM consists of models derived from the structure specified in the query, ERM is derived from results obtained from unstructured keyword search. Also, the goal of SRM is to predict values of empty fields, whereas ERM targets the ranking of keyword search on structured data. In this setting, scores have to be combined from several resources (tuples), using a monotonic aggregation function. Thus, while ERM is similar to SRM in concept, the way it is constructed and how it is used are different.

The smoothing method we use represents an instantiation of an existing framework [161]. We adopt it to the case of structured data and show how the local neighborhood of resources can be exploited to obtain smoothed estimations of the edge-specific attribute probabilities. This technique is clearly different to existing work that instead, uses the local corpus structure of documents (e.g. document clusters, neighbors etc.) [124, 218].

Throughout the chapter, we discussed existing approaches including various adoptions of IR-style ranking [143, 150] that focus on the problem of keyword search on structured data. However, we note that this is the first work that investigates the use of language models. Instead of employing questionable normalizations and heuristics, we provide a principled approach that directly models the relevance behind queries and results.

Searching the structured data is also very important for data integration [36] or peer-to-peer (P2P) networks since there are a number of sources found on the Web. However, the semantics of the data is explicitly tied to the underlying data model and a generic ranking model is definitely needed for this purpose. Different techniques are applied in this regard to find, query and join the structured search results from various sources [36, 157] with the applications on the different domains such as to search healthcare [17, 18, 19, 65] or tourism data [113]. We consider that our work also provides a useful ranking scheme orthogonal to the techniques in these domains to be applied in a generic way to improve the search relevance. In this regard, we have successfully applied the technique presented here for a domain-specific scenario in the environment domain to provide an intuitive access to the environmental data as an addition to the generic datasets we used in the experiments [1, 16].

6.6 Conclusion

Keyword search on structured data is a popular problem for which various solutions exist. We focus on the aspect of keyword search result ranking, providing a principled approach that employs language models to capture results, queries and the relevance behind them. A recent study has shown that existing heuristics and normalizations proposed for this problem exhibit good results only in the previous ad-hoc experiments, but fail to deliver consistent performance across different information needs and datasets, and especially, do not deliver stable performance across the precision-recall curve (low precision at higher recall levels). Through a standardized evaluation, we show that our approach delivers superior results, largely

outperforming all existing systems in terms of precision, recall and MAP. Further, we formally show that the ranking function is monotonic. This is of great value in practice, enabling the proposed ranking scheme to be used in combination with state of the art approaches for the efficient computation of results.

7

Learning-to-Rank with Semantic Kernels

In this chapter, we turn our focus to a discriminative way of ranking by using the semantic information available as RDF data to decide whether a document is relevant to a query or not. In particular, Learning-to-rank (LTR) as discussed in section 2.3.4 is an emerging approach in IR that learns a ranking function from a given training data. The effectiveness of any LTR approach highly depends on the features extracted from the training data, and is usually constructed based conventional features such as query-document scores, metadata, or document-specific annotations. Thus, a typical ranking function in LTR can be considered as a statistical model of relevance discriminating the documents with particular features to be relevant to a query from the other documents.

In the following, we present how the semantic information available from an external RDF data can be used to annotate the documents and act as semantic features of those documents for training an LTR model. To achieve this, we first present a novel learning algorithm specifically developed to operate on RDF data, called Semantic Kernel Machines (SKM) [20]. SKM is extending Support Vector Machines (SVM) combined with a multiple kernel that concurrently apply an embedded feature selection and classifier training. Then we present how SKM can be adopted and used for LTR by employing the RDF data as semantic features of the query-document pairs. In our experiments, we show that SKM is a powerful approach for both classification of the RDF data and ranking of the documents for a given query.

7.1 Semantic Kernel Machines

The amount of *semantic data* captured in ontologies or made publicly available as RDF Linked Data is large and ever increasing on the Web. Generally, semantic data (as well as relational and XML data) can be captured using a graph-structured model where nodes may stand for resources, objects, entities, classes of entities etc., and edges may capture their properties, or relations between them. RDF data for instance, consist of resources and literals that can be modeled as nodes, and RDF triples correspond to edges. An example data graph capturing movie related information is given in Fig. 4.2.

Given their practical relevance, the problem of learning from semantic data has gained attention. A standard learning problem that is the subject of this paper is *classification*. In this setting, we have a training dataset $\mathcal{D} = \{(x_1, y_1), .., (x_n, y_n)\}$ with data point samples $x_i \in \mathcal{X}$ and their labels $y_i \in \{+1, -1\}$. A small training dataset according to our example graph in Fig. 4.2 might be $\mathcal{D} = \{(RomanHoliday, +1), (TheAviator, -1), (TheHoliday, -1)\}$, and a classifier learned from such training set to predict "Audrey Hepburn" movies. The intuition here is that the data points of the training data is embedded in an RDF graph as shown Fig. 4.2 and semantic data around this data point can help us to train a classifier for prediction.

This problem can be considered as a standard classification by representing the RDF data in a propositional (feature-based, attribute-value) form and applying one of the techniques described in Sec. 2.3.1. However, existing techniques to propositionalization are difficult to apply in the setting where the data is represented as large, connected RDF graph [123]. In classical data mining, propositionalization removes all relations from the data set and a common form of propositionalization in databases is denormalization, where multiple tables in a database are joined into a single one. If any of the relationships in the original schema are one-to-many or many-to-many, then records are repeated many times in the join, which can skew the marginal distribution of an attribute. This is the case in our setting in which RDF semantics donot restrict the relationship cardinality and one-to-many or many-to-many are highly common. For example, a movie (e.g. *Roman Holiday*) in the Fig. 4.2 can have many *starring* relationships (e.g. *Audrey Hepburn*, *Gregory Peck* etc.), as well as an actress (e.g. *Audrey Hepburn*) stars (i.e. has starring) in many movies (e.g. *Roman Holiday*, *Breakfast at Tiffany's*). When joined, the frequency of particular attributes would depend on how often movies or actors have those attributes. This problem is known as relational autocorrelation in data mining [105].

The presence of autocorrelation provides a strong motivation for using relational learning techniques which are briefly described in Sec. 2.3.3. For learning over such structured data (e.g. graphs, networks, relational databases), this line of work, broadly speaking, includes Inductive Logic Programming (ILP) [167], and Statistical Relational Learning (SRL) [80]. In the context of classification, *SRL* is particularly relevant and adoption of existing approaches to the case of semantic data has been proposed recently [119, 192]. More related to our work is also the line of work on using kernels and *Support Vector Machines* (SVM). Essentially, SVM learning can be conceived as operating on simple vectors of real numbers representing features of the data points $x_i \in \mathcal{D}$ in a vector space, and in the case of classification, the goal is to find a hyperplane in this space that separates points in $\mathcal{D}^-$ from points in $\mathcal{D}^+$. SVM learning in combination with kernels proved to be flexible in dealing with data of different types. This is because to find the hyperplane we only need to access the dot product of two given data points, and in principle, a kernel function for computing that can be designed for data of any kinds. Exploiting this idea, specific kernels for semantic data have been proposed: Edges representing certain kind of relationships [29], paths [100] as well as complex DL axioms [71] can be considered. Compared to SRL-based approaches which aim to construct a graphical model over the entire set of data, SVM only finds the discriminative points [206], i.e. support vectors, in the training data and thus, is a promising candidate for dealing with large amounts of semantic data on the Web.

Further, the authors in [29] made clear that a combination of several kernels can be used

to capture different, "useful" subset of features, and proposed a simple weighted additive combination. However, the crucial questions of (1) which features to use for constructing the base kernels (*feature selection*), (2) which ones to use in the combined composite kernel and how to set their relative importances, i.e. *multiple kernel learning* (MKL), are left open. In fact, not only SVM but also the mentioned SRL approaches [119] require features to be chosen manually by an expert. This is a difficult task especially in the semantic data setting because a data point x_i might have a large set of features. In fact, the entire data graph may be relevant as all its constituent nodes are somehow connected with the node pairs represented by x_i.

A more practical approach that can deal with these challenges is to automatically select some high level features and then, to search for related features in the data, and progressively incorporate only those features into the composite kernel, which help to improve the classification performance. Aiming at this intuition, we provide the following contributions in this work:

- We propose a framework called *Semantic Kernel Machines* for learning from graph-structured data such as RDF. This framework extends classical SVM and its adoptions to semantic data to incorporate feature selection into the problem of learning multiple kernels on semantic data.
- We show how this framework can be instantiated by adopting the ILP view of *searching for features* as the problem of learning of hypotheses that best capture training data,
- Recently, MKL methods [8, 189, 230] have been proposed, showing promising results for the automatic computation of an optimal composite kernel as a combination of base kernels. However, they cannot be directly applied to this more complex "combined" problem, where the search for valid hypotheses and the training of the composite kernel are two dependent processes that need to be solved concurrently. We propose a *co-evolutionary genetic algorithm* (GA) to solve this optimization problem.

Based on our previous publication [20] we introduce our framework in Section 7.1.1 , where we start with the clause kernels, then discuss how they can be used for learning in the MKL setting. We show how the resulting kernel machines can be optimized using a co-evolutionary genetic algorithm in Section 7.1.4.

7.1.1 Framework for Semantic Kernel Machines

In this section, we propose a framework called *Semantic Kernel Machines* for learning from graph-structured data such as RDF. While previous work proposed base kernels [29, 58, 71] and their linear combination [29] that have to be predefined for a given dataset manually, the framework presented here involves learning a non-linear combination of kernels that are automatically constructed from relevant features sets computed on the fly.

Given a data point, i.e. a triple, all connected triples (the entire dataset in the extreme case) might be considered as relevant features. For learning relevant features, we propose to adopt the ILP view of Sec. 2.3.3.1 that considers a hypothesis as an intensional description of data, i.e. the feature set of some given data points. Denoted $H = \{c_1, \ldots, c_n\}$, it is basically a set

of clauses $h(\overrightarrow{v}) \leftarrow b_1(\overrightarrow{v_1}), .., b_m(\overrightarrow{v_m})$, where h and b_i are predicates representing a clause's head and body, and $\overrightarrow{v_i}$ are either variables or constants. For example, a clause indicating all users from the *United Kingdom* who like *comedy* movies can be stated as follows:

$$c_1 : likes(?u, ?m) \leftarrow loc(?u, UK), genre(?m, comedy)$$

Note that the clause head can be conceived as some data points and the body represents their features. Based on this view, the proposed framework mainly includes (1) the search for relevant clauses to be included in the hypothesis to construct the base kernels, namely clause kernels, (2) and the formulation of the learning as an MKL problem.

7.1.2 Clause Kernels

A clause or a clause kernel, respectively, represents one relevant subset of the feature space as captured by the hypothesis. Every clause contains a set of predicates, which stand for dimensions of the feature subset. We propose to use an R-convolution kernel [88] for every clause to construct base kernels on dimensions of different feature subsets. We define a clause kernel $K_c(x_1, x_2)$ over each pair of data points $x_1, x_2 \in X$. Following the R-convolution kernels, we define a decomposition relation $R_c^{-1}(x)$ over the data point x for a clause c as follows:

- Given x, the substitution $\theta = \{v/x\}$ instantiates the variable v of c with x to generate instantiated clause as $c\theta$, and
- given the instantiated clause, another substitution $\theta' = \{v_1/b_1, \cdots, v_m/b_m\}$ instantiates the remaining variables v_i in $c_b\theta$ with bindings b_i,
- then the θ' is a R-decomposition relation for c denoted $R_c^{-1}(x)$ iff $G \models c_b\theta\theta'$ where G denotes the data graph.

In the RDF context, entailment is evaluated simply via graph pattern matching, i.e., $G \models c_b\theta\theta'$ if $c_b\theta\theta'$ is a binding to the query pattern $c_b\theta$ such that it is a subgraph of G.

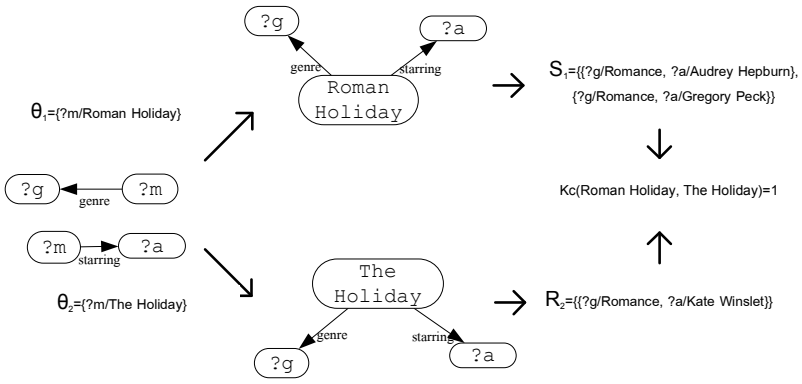


Figure 7.1: An example calculation of clause kernel.

Intuitively speaking, we need to instantiate clause variables for each data x. For example, for a clause

$$c_1 : movie(?movie) \leftarrow starring(?movie, ?a), genre(?movie, ?g)$$

a substitution $\theta = \{?movie/Roman\_Holiday\}$ results in the instantiated clause:

$$p_1 : movie(Roman\_Holiday) \leftarrow starring(Roman\_Holiday, ?a), genre(Roman\_Holiday, ?g)$$

.

Given two points (x_1,x_2), a kernel can be calculated by instantiating c this way to obtain $c\theta_1$ and $c\theta_2$, and showing how similar x_1 and x_2 are, based on the resulting instantiated clauses. A trivial evaluation of this similarity is to consider the instantiated clauses $c_b\theta_i$ as graph patterns to be evaluated on the data graph. For this, given G, we define a result set as $S_{c_b\theta} = \{\langle b_1, .., b_m\rangle \mid \theta' = \{v_1/b_1, .., v_m/b_m\} \wedge G \models c_b\theta\theta'\}$. Based on two result sets, a kernel function can be defined as:

$$k(c_b\theta_1, c_b\theta_2) =\mid \{\langle b_i, b_j\rangle \mid b_i \in S_{c_b\theta_1} \wedge b_j \in S_{c_b\theta_2} \wedge b_i = b_j\} \mid$$

Given $c_b\theta$ as a set of feature dimensions, the similarities of two data points are measured by retrieving values of those dimensions represented by variables in $c_b\theta$, and check if these points agree on these values. The resulting clause kernel is a R-convolution kernel in the sense that $R_c^{-1}(x)$ decomposes the feature set captured by c into dimensions, and subkernels k_i are used to operate on these dimensions, i.e., $k_i(x'_1, x'_2)$ where $x'_1 \in R_c^{-1}(x_1)$ and $x'_2 \in R_c^{-1}(x_2)$. An example illustrating this is given in Fig. 7.1. In this case, the two data points agree only on the value $Romance$ for the dimension $genre$.

7.1.3 Learning Kernel and Parameters

We consider learning a kernel $K : X \times X \rightarrow \mathbb{R}$ that is expressed as a sum of clause kernels K_c constructed from clauses $c \in H$, where $d_c \in \mathbb{R}_+$ and $\sum_{c\in H} d_c = 1$:

$$K(x, x') = \sum_{c\in H} d_c K_c(x, x') \tag{7.1}$$

To solve this MKL problem, a two-steps alternating optimization algorithm has been proposed where firstly, the SVM coefficients $\{\alpha_i\}_{i=1}^n$ are optimized with fixed $\mathbf{d}$, and the weight vector $\mathbf{d}$ is then updated with fixed $\{\alpha_i\}_{i=1}^n$ [189]. In [230], a gradient descent algorithm is used for the update of $\mathbf{d}$, which we will use in the next section. The dual form of this problem for the first step is:

$$\max_{\substack{\alpha \\ \sum_{i=1}^n \alpha_i y_i = 0,\ C \geq \alpha_i \geq 0}} -\frac{1}{2}\sum_{i,j=1}^n \alpha_i\alpha_j y_i y_j K(x_i, x_j) + \sum_{i=1}^n \alpha_i \tag{7.2}$$

The resulting $\boldsymbol{\alpha}^*$ are used in the second step to optimize the same objective function w.r.t. the weight vector $\mathbf{d}$:

$$\max_{\substack{d \\ \sum_{c\in H} d_c = 1,\ d_c \geq 0}} -\frac{1}{2}\sum_{i,j=1}^n \alpha_i^*\alpha_j^* y_i y_j K(x_i, x_j) + \sum_{i=1}^n \alpha_i^* \tag{7.3}$$

This two-step MKL formulation assumes that the base kernels in K are fixed. However, in order to

include feature learning, the set of hypothesis H cannot be assumed to be predefined and fixed in terms of clauses. Rather, we propose to include the search for hypothesis and inclusion of new clause kernels K_c as an additional step. This is analogues to dynamic feature construction in ILP approaches [182], where given the search space $\mathcal{H}$, we start with the most general clause, and by the iterative refinement of clauses, we aim to find a hypothesis that covers all positive examples and no negative ones. At every iteration t, an operator $\rho(c')$ is executed on a selected clause c' from a hypothesis H^t to perform refinement in two ways:

1. It picks a predicate $b_i(\overrightarrow{v_i})$ from c' and finds in the RDF graph another predicate b_j that is connected to b_i.
2. It replaces a free variable in c' with a ground term (e.g. label of an RDF resource).

The iterative application of $\rho(c')$ results in a refinement graph having the most general clause at the root and other nodes represent refined clauses. An example is shown in Figure 7.2.

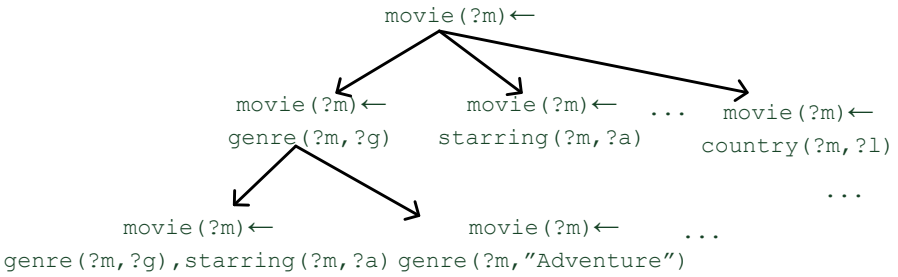


Figure 7.2: A fragment of the example refinement graph

Basically, incorporating feature learning into MKL can be done as follows: at every iteration t, we generate hypotheses via the refinement of previous ones, and for every hypothesis H obtained so far, we apply the two-step MKL optimization to solve the SVM coefficients and weight vector. After the optimization is done, we obtain for every H a kernel prediction function of the form:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \sum_{c \in H} d_c K_c(x_i, x) \tag{7.4}$$

Then, we can use the prediction error on the classification problem to select the best one, e.g. using the hinge loss function $l(y, f(x)) = max(0, 1 - y * f(x))$. Clearly, this is an expensive procedure because optimal parameters have to be computed for all possible candidate hypothesis, even though many of them turn out to be non-optimal, i.e., yield high prediction error.

7.1.4 Co-evolutionary Optimization

We aim to optimize this process by means of evolutionary GA. Basically, GA iteratively applies crossover and mutation on the fittest solutions to breed new refined solutions. Compared to the previous optimization, this is rather an approximate way of finding the optimal solutions, as only the fittest one are considered. In particular, we propose to focus only on the fittest hypotheses and the fittest SVM coefficients instead of computing all optimal coefficients for all hypotheses. This is achieved by using a co-evolution technique [70] that gives the opportunity of breeding more than one type of species in one population. At every iteration, new hypotheses are generated, and their weights and SVM coefficients are trained simultaneously in 2 steps: (1) select fittest hypotheses and coefficients to perform *crossover*, (2) and perform *mutation* to further optimize and refine the results.

Chromosome Coding. In our case two different but dependent subpopulations are to be trained, as sketched in Fig. 7.3. Each individual I_i^H in the first subpopulation represents a unique hypothesis H_i with a chromosome of $|H_i|$-dimensional real-valued vector $(d_1^i, .., d_{|H_i|}^i)$ indicating the weights of the clauses of H_i. During co-evolution, crossover and mutation result in new hypothesis individuals, which comprise new and refined clauses. Therefore, individuals' chromosomes may have different lengths. However, once a particular hypothesis individual is introduced in the subpopulation, its chromosome length (and accordingly its clauses) does not change over time. This is to ensure that the idea of MKL can be applied to our problem, i.e. iterative parameter optimization is performed for a fixed hypothesis. The second subpopulation is meant to train the SVM coefficients I_j^S called SVM individuals with the chromosomes $(\alpha_1^j, .., \alpha_n^j)$.

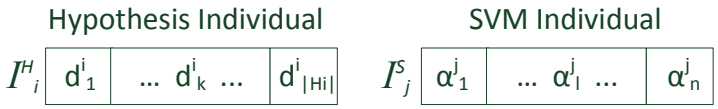


Figure 7.3: Coding of hypothesis and SVM individuals

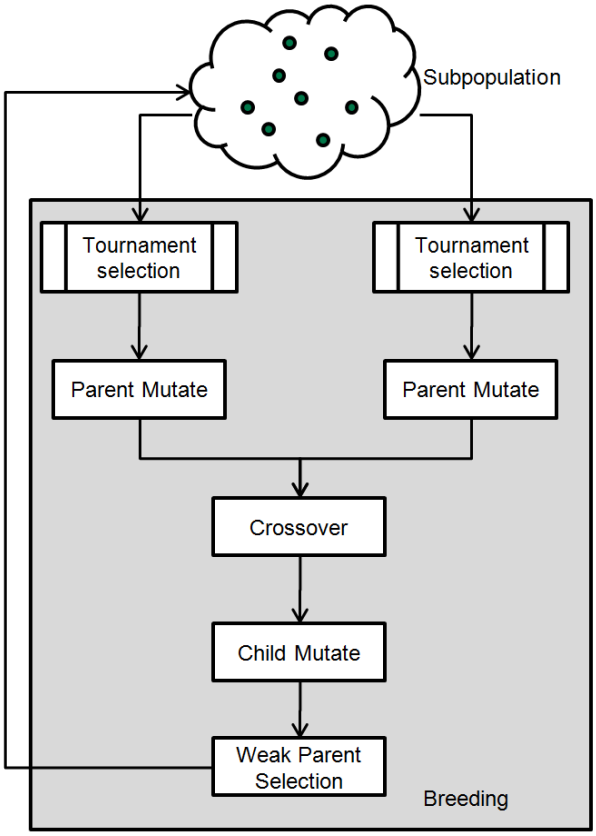


Figure 7.4: The breeding process of hypothesis individuals

Initialization. For the subpopulation of SVM individuals, a random number of the support vectors is chosen and these vectors are given random numbers between 0 and 1 to initialize each chromosome.

The remaining genes of each chromosome is set to 0. For hypothesis individuals, a refinement graph is firstly constructed by calling $\rho(c')$ on the most general clause c' as discussed before. Then randomly selected clauses (only one from each leaf-to-root path) and generated values are used to instantiate a hypothesis individual and its weight vector.

Crossover. For breeding new individuals at every iteration, the fitness has to be evaluated for some existing individuals. Based on a "optimal" SVM individual from the other subpopulation, a hypothesis individual is evaluated via equation 7.3. Vice versa, the fitness of an SVM individual is calculated using equation 7.2, based on a "optimal" hypothesis individual. For finding this "optimal" individual in an approximate way, we use standard tournament selection, i.e. use a few individuals (25%) chosen at random from the other subpopulation and in this case, choose the one that yields the highest fitness value. This tournament selection is also applied for finding the two fittest ones to form the offspring generation. This crossover results in a new individual created based on exchanging values of the two parent vectors. Since hypothesis individuals may have chromosomes with different lengths, we introduce zero weights when needed to ensure that the length of parent vectors are aligned. We employ uniform crossover by randomly generating binary masks of size m (aligned $|H_i|$) for mixing parent vectors. Crossover probability is set high (0.9). The resulting weight vector $\mathbf{d}$ and coefficients $\boldsymbol{\alpha}$ are normalized to satisfy the constraints in equation 7.3 and 7.2.

Mutation The mutation operator for hypothesis individuals applies in two different ways: In *parent mutate*, parameters of individuals from previous generations shall be further optimized as they may continue to serve as candidates for fitness selection. Thus, for the "fit" parent individuals chosen previously during the crossover, we update the weight vector $\mathbf{d}$ reflecting the gradient descent, using the scheme previously proposed for MKL [230]. For this, we take a simple differentiation of the dual function in 7.3 w.r.t. every gene d_c as:

$$\frac{\partial J}{\partial d_c} = -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i^* \alpha_j^* y_i y_j K_c(x_i, x_j) \quad \forall c \in H \tag{7.5}$$

and apply an updating scheme such as $d_c \leftarrow d_c + \gamma \frac{\partial J}{\partial d_c}$, where γ is the step size chosen according to Armijo rule as suggested in [230]. Such an update allows the optimization of the weight vector when the individual is not eliminated over the generations.

In child mutate, for every new individual, a clause c in its hypothesis is randomly chosen and replaced with two refinements. This mutation ensures that new hypotheses are incorporated into the learning via the refinement graph allowing to increase the size of hypothesis and also to apply a general-to-specific refinement. However, the refinement of c may yield a large set of clauses $\{c_1, .., c_k\}$. In order to select the two best ones, we apply Kernel Target Alignment [53] to determine how each clause kernel K_{c_i} is aligned with the ideal kernel Y. Note that the ideal kernel Y is calculated over the given labels y_i associated with data points in the training dataset, i.e. $Y = yy^T$, where $y = (y_1, ..., y_n)^T$. A simple measure of alignment between the clause kernel K_{c_i} and Y is provided by the Frobenius inner product $\langle ., . \rangle_F$ defined as

$$\langle Y, K_{c_i} \rangle_F = \sum_{y_i = y_j} K_{c_i}(x_i, x_j) - \sum_{y_i \neq y_j} K_{c_i}(x_i, x_j) \tag{7.6}$$

The mutation operator for SVM individuals follows the same random process used to generate the initial subpopulation, but this time without setting the remaining genes to zero.

After mutation, a new individual replaces a parent individual when the former has a higher fitness score. This is to improve the overall fitness of the subpopulation.

This co-evolutionary process stops after a fixed number of generations (other stopping criteria may be applied). A kernel is constructed by selecting any two individuals, one from each subpopulation.

From all possible kernels, we choose the optimal one based on prediction error as discussed before.

7.2 Learning-To-Rank with Semantic Kernels

As we discuss in Sec. 2.3.4, Learning-to-Rank (LTR) approaches do not assume a generative model for relevance, but uses the training data to find the optimal model as a discriminative model for relevance. In training such a model, the LTR approaches assumes that any data point (i.e. query-document pair) is first transformed to the set of features that is represented as a feature vector $\overrightarrow{x}$ =<f_1,..., f_m> where every feature f_i represents the score of a corresponding feature of the document and query under a predefined criteria. Most of these features are on basis of textual content of documents and queries. For example, some of the most popular conventional features include IR-based ranking model scores (also called as *low-level content features* in the literature [249]) such as:

- Term Frequency (TF) — the number of times a term (belonging to the given query) occurs within a document
- Inverse Document Frequency (IDF) — inverse of the number of documents within the document corpus, which contains the given term
- Document length (DL)— number of terms within a document
- TFIDF — TF*IDF, combination of TF and IDF

In addition to those features, some other ranking model scores such as BM25 or language model are also very widely used. The query-independent features such as PageRank of the document has also been applied.

Using these linear vector of features, LTR approaches, in fact, transform into an optimization problem of deciding on which features are more preferred over the others. Let's assume that we are learning a linear ranking function $h(\overrightarrow{x}) = \overrightarrow{w}.\overrightarrow{x}$ similar to those in *RankingSVM* [107]. Here the weight vector $\overrightarrow{w}$ is adjusted (i.e. optimized) by the learning algorithm and determines the relative importance of the features. For example, if the data points are represented by the abovementioned four features, then a weight vector $\overrightarrow{w} = (2, 0, 3, 8)$ gives high importance to the fourth feature which is the TFIDF score of the query-document pair. In fact, most of the LTR approaches employ a similar scheme to this and act as a combination technique to find out which mixture of features is the best based on the given training data. In this regard, the performance of a LTR approach is upper-bounded by the optimum performance of its features and thus rely on the techniques to estimate these features (e.g. TFIDF).

Despite such importance of the features, only a small number of works have been concerned to elaborate how the LTR features can be improved by employing a feature selection as a pre-processing step to learning [78, 99, 176], while most of the LTR approaches focus on improving the learning algorithms considering the features as fixed. In [78] a method for feature selection among a large feature set is proposed that uses the value of every feature to rank the training data and defines ranking accuracy as the metric of the importance of a feature. This is further extended in [176] which explored different feature selection methods using booted regression trees, or greedy algorithms. [99] proposes a hierarchical feature selection method containing two phases to make the features not biased and has designed a quality measure to decide the proper number of selected features. Although these works are an initial step to discuss the quality of the features, they do not suggest any new features that can replace and perform better than the conventional features used in LTR.

In this work, we relax the abovementioned feature weighting nature of the previous LTR approaches and propose a technique to train a discriminative ranking model by utilizing the so-called *semantic features* emerging from structured RDF data. In particular, our work, called *learning-to-rank with semantic kernels*, relies on the *semantic kernel machines* (SKM) concept that is introduced in the previous section.

Instead of conceiving the training data points as linear feature vectors, we consider that every document is actually embedded as a part of an extended data graph on which we can apply the clause-based learning approach of SKM using multiple kernels. This results in a discriminative model in which the ranking of a document depends on its positioning in the data graph and relationships it holds to other entities. In this regard, the relevance is determined not by the conventional features, but the discriminative semantic informaiton learned by our SKM approach.

In the following, we give the details of our LTR approach. In Sec. 7.2.1 we elaborate the concept of semantic features and discuss how they differ from the conventional features of LTR. In Sec. 7.2.2, we present how the documents can be linked to the entities in the data graph to construct an extended data graph in which documents are also conceived as a part of the RDF data. Sec. 7.2.3 explains the kernel technique, which has played a very important role in our case, ensuring the processing of semantic data. In Sec. 7.2.4, the query-dependent LTR problem will be discussed.

7.2.1 Semantic Features

The core difference between the semantic features and the conventional features of LTR is that semantic features focus on *relational information* while conventional features rely on the existing IR-based metrics. Although the latter is easy-to-obtain from the index of a search engine, their capability to define the relevance in the learned ranking function is only limited to the capabilities of the features. This poses difficulties to capture the actual semantic information imposed in the training data and particularly within a specific query session from which the training data is obtained. To elaborate this a bit further, suppose that the following document is available in the training data:

> "It is an island of Indonesia. With a population of 135 million , it is the world's most populous island, and one of the most densely populated regions in the world. It is home to 60% of Indonesia's population. The Indonesian capital city, Jakarta, is in west of it. Much of Indonesian history took place here: it was the centre of powerful Hindu-Buddhist empires, Islamic sultanates, the core of the colonial Dutch East Indies, and was at the centre of Indonesia's campaign for independence. The island dominates Indonesian social, political and economic life."

It is easy to interpret with some background knowledge that the document is about "Java island" – an island of Indonesia– although it is not explicitly mentioned within the text. Therefore, the standard feature values such as TF, TFIDF, or BM25 will be (close to) zero for this document. If the data point containing this document is labelled as positive, which is apparently so, the learning algorithm will lower the weight of such content-based features (i.e. TFIDF, BM25 etc.) and opt for non-content-based features such as Pagerank score, or document length because it will assume that the latter is more discriminative. In Chapter 5, we already discuss that such vocabulary mismatch is very common in Web documents and can lead to omiting the actual content of documents in LTR.

Instead, such a document can be linked to the entities in RDF graph via an linking function $L : D \rightarrow V_E$ as we present in the following. For the moment, we detect the entities directly associated with the document such as *Indonesia*, *Jakarta*, *Dutch East Indies*. Fig. 7.5 shows part of RDF graph including those entities and it is observed that they are connected to the entity *Java* via some relationships. In particular, *Java* has direct relationships with the entities *Indonesia* and *Jakarta* such as *Java* island is located in *Indonesia* and the largest city on the island is *Jakarta*. Although the entity *Dutch East Indies* has no direct relationship with entity Java, we can see that the capital of *Dutch East Indies* was today's *Jakarta*. In this regard, the clause-based learning of SKM of the previous section can be applied for learning in such a setting. In particular, if the document is linked to the entities via the function $L(d) = \{Indonesia, Jakarta, Dutch\_East\_Indies\}$, the following clause explores the connection

between the $Java$ and $Dutch\_East\_Indies$:

$$about(?d, ?e) \leftarrow capital(?e, ?c), largestCity(?l, ?c)$$

As we recall from SKM that such clauses are not given a priori but dynamically learned during the optimization process. Once the clauses are found, our learning approach introduces a clause kernel for every clause instead of relying on a predefined linear feature vector. Intuitively, every clause kernel corresponds to a particular feature space that characterizes the projection of the semantic information in RDF into the data points in the training data. Note that every data point is mostly related to more than one feature space (i.e. clause kernels). Then there exists m clauses and accordingly m mappings that for each $i = 1, ..., m$ we can assume that there exists a mapping Φ_i from input space $\mathcal{X}$ to a Hilbert space $\mathcal{F}_i$ such that:

$$K_i(x_1, x_2) =< \Phi_i(x_1), \Phi_i(x_2) >$$

This result provides a first useful intuition about kernel K_i that we can construct m kernels each of which can be thought of as dot products in some space $\mathcal{F}_i$. As in the case of SKM, the representation $\Phi_i(x)$ does not need to be computed explicitly that we can apply *kernel trick* to replace each dot product by a clause kernel calculation. In this way, the non-linearity of the semantic features can be handled without the need to generate a linear feature vector.

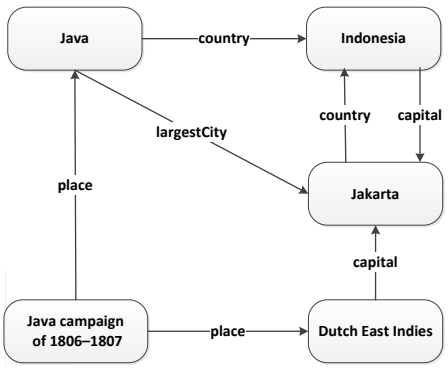


Figure 7.5: Part of RDF graph about Java island.

Another difference of our approach from the previous LTR approaches is the way we handle the query dependency in the training data. Most of the LTR approaches implicitly incorporates the query information into the learning algorithm via the query-dependent features such as TFIDF, BM25, or language model scores, since the value of those features depends on the query. Even in some datasets, the query information itself such as the number of words in a query is also considered as additional feature [249]. However, in our approach a document d is linked to the data graph via a entity linking function $L(d)$ which only depends on the document content and discards the query information. We actually incorporate the query dependency in our multiple kernel learning approach by introducing the query-dependent weights to each clause kernel in order to create *query-dependent localized multiple kernels*. The intuition behind this is that the importance of each clause (and accordingly kernel) varies according to the each query, because in our approach clauses capture the actual semantic meaning of the feature spaces.

7.2.2 Topical Linking of Documents to RDF Data

In order to adopt our method of SKM for learning-to-rank, our initial requirement is the data points of the training data to be part of the RDF data graph. In particular, given all the documents in the training data $D = \{d_1, ..., d_m\}$, we introduce every document $d \in D$ with its unique identifier as an entity. In addition, we specify that those document entities has $type$ of a special class called $Document$ in order to distinguish them from other entities. We also consider that textual content of the document d can be used by an *entity linking function* $L(d)$ to find out the associated entities in the RDF graph. For those entities $e \in L(d)$ we assume a special relationship, called $about$, to exists in order to indicate such a connection. Then based on our definition of data graph in Def. 1, we define an extended data graph including the document entities and their associations to other entities as follows:

Definition 14 *(Extended Data Graph) Given a data graph $\mathcal{G}$ and a set of documents $D = \{d_1, ..., d_m\}$, a* document-entity linking function *L is defined from D to a subset of V_E that for each $d \in D$, $L(d) = \{e_1, ..., e_{k_d}\}$ gives the annotated entities for that document. Based on L we define an* extended data graph *$\mathcal{G}^+ = (\mathcal{V}^+, \mathcal{E}^+, l_{V^+})$ where:*

- $\mathcal{V}^+ = \mathcal{V} \cup D \cup \{Document\}$,
- $\mathcal{E}^+ = \mathcal{E} \cup \{< d, about, e >\}$ *for all $d \in D$ and $e \in L(d)$*

The key assumption in this definition is the existence of the entity linking function $L(d)$. In particular, different techniques can be employed for this purpose [34, 55, 163]. One simple technique is to exploit Wikipedia for such a linking by using the features from the associated Wikipedia text to the entities and train a classifier to make predictions. As we discuss in Chapter 4, such approaches are not generic and highly tailored to the underlying knowledge base in terms of the allocated features. Another problem of these approaches is the strong dependency on the so-called textual entity mention (i.e. keywords) in the document text which is mostly identified by a named entity recognizer and matched to entities containing similar keywords in their identifiers. For example, in our experiments with *Wikipedia Miner*[1] [163], the classifier incorrectly tags the words "about" with the Wikipedia entity for About.com which is a commercial Web site composed of editorial articles about public-related topics.

In our approach we derive an *entity linking function* based on the *topical relational models* (TRM) of Chapter 4. In particular, TRM provides a well-established model to probabilistically determine the connection between the documents and the entities. With a trained TRM model as described in Chapter 4, our aim is to predict the probability of observing a document $d \in D$ given an entity e which is denoted as $p(d \mid e)$. Recall that in TRM topic-word probabilities are learned as a parameter $\boldsymbol{\beta}$ for each topic and each entity is considered as a mixture of topics captured by a topic proportion variable $\boldsymbol{\theta}(e)$. Then the probability of a document given an entity is defined as:

$$\begin{aligned} p(d \mid e) &= \sum_{w \in d} \sum_{t=1}^{K} p(w \mid \beta_t)\, p(t \mid e) \\ &\approx \sum_{w \in d} \sum_{t=1}^{K} p(w \mid \beta_t)\, \mathrm{E}_q[\theta_t(e)] \\ &= \sum_{w \in d} \sum_{t=1}^{K} \beta_{tw}\, \gamma_t(e) \end{aligned} \tag{7.7}$$

where we exchange the probability $p(t \mid e)$ with the posterior distribution $q(\boldsymbol{\theta}(e) \mid \boldsymbol{\gamma}(e))$ from Equation A.2. We use Equation 7.7 to score each entity and consider that top-k entities, denoted $topK(d)$, are

[1] http://www.nzdl.org/wikification/index.html

relevant to the document. In this case entity linking function can be defined as $L(d) = \{e \mid e \in topK(d) \wedge p(d \mid e) > 0\}$.

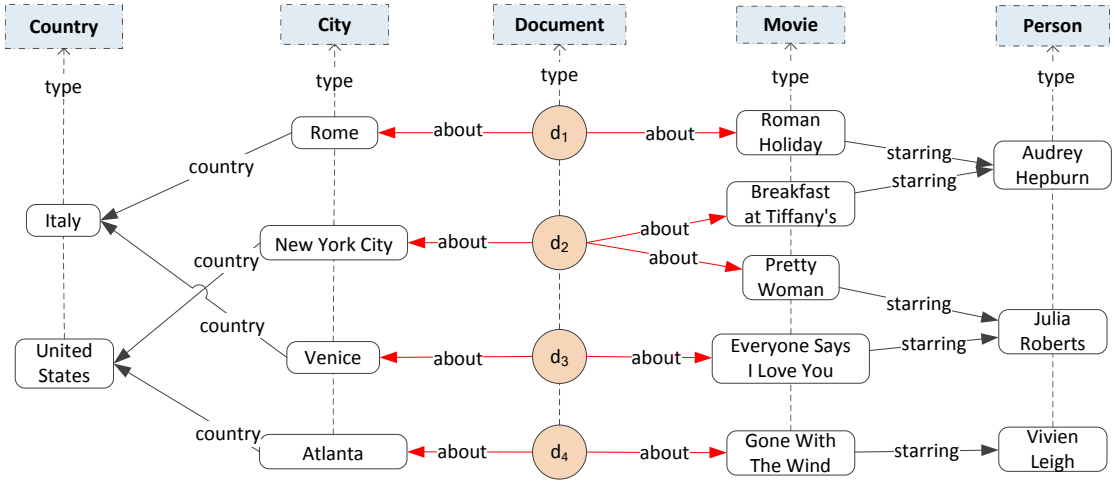


Figure 7.6: An example extended data graph.

An illustration of extended data graph is shown in Figure 7.6, where d_1, d_2, d_3 and d_4 denote documents to be included into the data graph. In this example, document d_1 has relevant entities such as the city "Rome" in "Italy" and the famous movie "Roman Holiday" starred by "Audrey Hepburn". Document d_2 talks about the city "New York City" and the movie named "Breakfast at Tiffany's" as well as the movie "Pretty Woman", while document d_3 is about the city "Venice" and the movie "Everyone Says I Love You". Document d_4 is relevant with the city "Atlanta" and the movie "Gone With The Wind" starred. Here we can see that the documents can be connected to each other via some paths over the associated entites even if they may have no direct relationship among each other.

7.2.3 Pairwise Clause Kernels

With the extended data graph in which the document entities are associated with the other entities, we now turn our focus on the learning problem where we adopt our SKM approach. In our setting, we approach the LTR problem as a pairwise approach in which the input space $\mathcal{X}$ is given as:

$$\mathcal{X} = \{< q_i, d_{ik}, d_{il} > \mid q_i \in Q \; and \; d_{ik}, d_{il} \in D^i\}$$

and the output space corresponds to the pairwise preference (which takes values from $\{+1, -1\}$) between each pair d_{ik}, d_{il} of documents. Here the positively labeled data point indicates that the first document is preferred over the second document in terms of the relevance to the query. In fact, such a training data can also be generated from a pointwise data in which only the relevance of the documents are given: Given the vector R^i of relevance of documents to the query q_i, then the pairwise preference for $< q_i, d_{ik}, d_{il} >$ can be defined as $y^i_{kl} = 2 \cdot \delta(r_k \succ r_l) - 1$ where $\delta()$ is the indicator function.

As in the case of SKM, we also employ the ILP view of using a hypothesis to construct the intensional description of data which consist of a set of clauses $\{c_1, \ldots, c_n\}$, and each clause is an expression of the form: $h(\vec{v}) \leftarrow b_1(\vec{v}_1), \ldots, b_m(\vec{v}_m)$, where both h and b_i are predicates representing the head and body of a clause respectively, and $\vec{v_i}$ can be either variables or constants. Here, we extend such definition of the clause with a specific type of the clause unique to our LTR problem, called *document-specific clause* with the following definition:

Definition 15 *(Document-specific Clauses) Given a set of documents D^i, an extended data graph $\mathcal{G}^+$, a clause c of the form $about(V_d, e) \leftarrow p_1(\overrightarrow{v_1}),\ldots, p_n(\overrightarrow{v_n})$ is defined where:*

- *V_d is the document variable*
- *$e \in L(d)$ is an entity variable, which is connected to d with the predicate:* ***about***
- *p_i is a binary predicate*
- *$\overrightarrow{v_i}$ is the parameter of predicate p_i, representing variables or constants.*

We instantiate the clause c with a substitution $\theta = \{V_d/d\}, d \in D^i$. A document-specific clause $c\theta$ can be defined as:

$$c\theta : about(d, e) \leftarrow p_1(\overrightarrow{v_1}), \ldots, p_n(\overrightarrow{v_n}).$$

The clause kernel in this case is not in pointwise any more, instead we use the pairwise clause kernel, which concerns each data point as a pair of documents.

Given the input space $\mathcal{X}$, an extended graph $\mathcal{G}^+$, a clause c, we now consider two pairwise data points $x_1 =< q_i, d_k, d_l >$ and $c_2 =< q_j, d_p, d_q >$ in $\mathcal{X}$. We instantiate four different document-specific clauses: $c\theta_k$, $c\theta_l$, $c\theta_p$, and $c\theta_q$, and four different result sets S is obtained for each document-specific clause according to the clause kernel in Sec. 7.1.2. Then pairwise clause kernel value $K_c(x_1, x_2)$ for two pairwise data points can be calculated as:

$$K_c(x_1, x_2) = |\{< b_k, b_p > | b_k \in (S_{c\theta_k} - S_{c\theta_l}) \wedge b_p \in (S_{c\theta_p} - S_{c\theta_q}) \wedge b_k = b_p\}|$$

Here the main difference from the kernel calculation in Sec. 7.1.2 is that we take the set difference (e.g. $S_{c\theta_k} - S_{c\theta_l}$) between the result sets of two documents in pairwise data points with the intention to find the bindings of which make the first document preferable over the second. Note that while doing so we do not take the label (i.e. $\{+1, -1\}$) of the pairwise data point into account. Assume that $d_k \succ d_l$ (so label is $+1$), which indicates that document d_k is more preferred over document d_l, then we are interested in counting those bindings which are able to distinguish the preferred document from the unpreferred one. In the other case (i.e. label is -1) we want to penalize according to those bindings which are in the first result set, but not in the second, because they are not able to make the first document more preferred. Thus in both cases the kernel value gives accurate results.

Example. Suppose that we have a simple clause $c_1 : about(V_d, e) \leftarrow starring(e, f)$ and data points $x_1 =< q_1, d_1, d_3 >$, and $x_2 =< q_1, d_2, d_3 >$ concerning the query q_1 (Audrey Hepburn Movie). An extended data graph for this is shown in Figure 7.7 where three documents from the data points, namely d_1, d_2, and d_3, are already part of this extended data graph. According to the definition of the pairwise clause kernel, we can get the result sets as:

- $S_{c_1\theta_1}$={Roman Holiday, Audrey Hepburn}
- $S_{c_1\theta_2}$={Breakfast at Tiffany's, Audrey Hepburn, Pretty Woman, Julia Roberts}
- $S_{c_1\theta_3}$={Everyone Says I Love You, Julia Roberts}

Therefore the value of clause kernel in this example is:

$$K_{c_1}(x_1, x_2) = |\{Audrey\ Hepburn\}| = 1.$$

where the binding of "Julia Roberts" has been removed from the result set during kernel calculation, so that only those useful and discriminant information w.r.t query q_1 such as "Audrey Hepburn" is obtained. In addition consider another clause $c_2 : about(V_d, e) \leftarrow country(g, h)$ whose subgraph is shown in

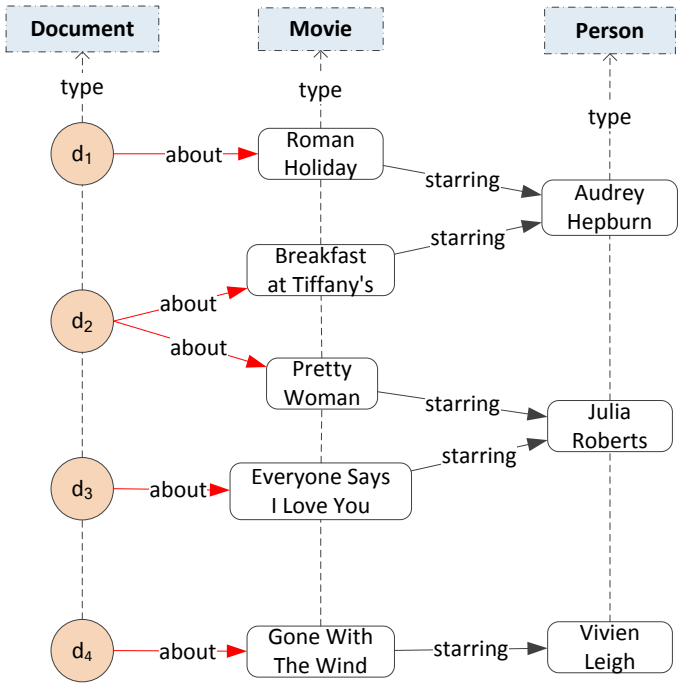


Figure 7.7: Subgraph constructed for clause c_1

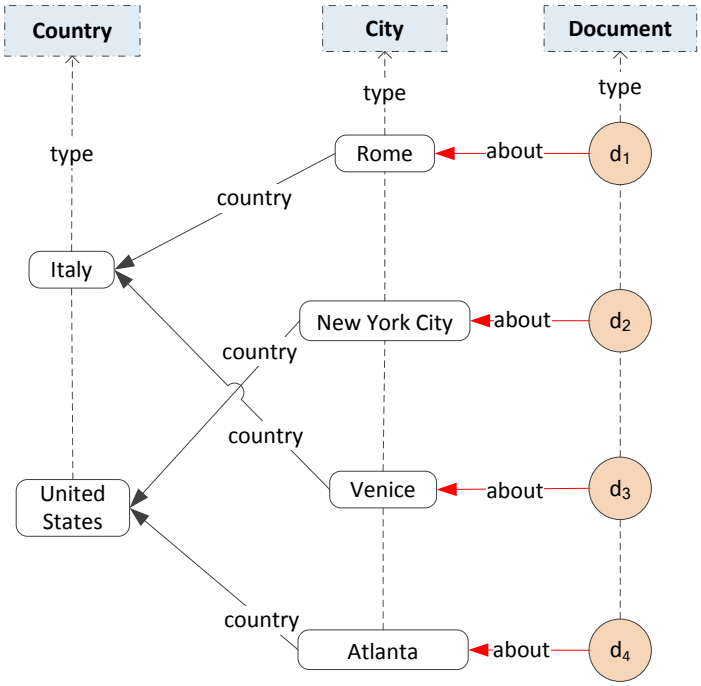


Figure 7.8: Subgraph constructed for clause c_2

Figure 7.8. Similarly we can obtain a kernel value for the same data points where $K_{c_2}(x_1, x_2) = 0$ due to the lack of a matching binding. This simple example reveals one important fact that the clauses allow to consider different parts of the RDF graph and models different semantic information. In fact, for query q_1 (i.e. "Audrey Hepburn Movies") we can assume that *starring* information has more discriminative power than the *location* information. We elaborate such dependency between the query and clauses in the next section.

7.2.4 Query-dependent Multiple Kernel Learning

In Sec. 7.1.3, we utilize each clause kernel as a base kernel and given a hypothesis $H = \{c_1, ..., c_n\}$ as a set of clauses we consider the learning problem as multiple kernel learning where combined kernel is specified as:

$$K(x_i, x_j) = \sum_{c \in H} d_c K_c(x_i, x_j)$$

Here d_c denotes the weight of clause c, which reflects the importance of the clause in the combination of multiple kernels and will be optimized in the learning process. When we adapt this MKL method to our LTR domain, this introduces some difficulties, since under this condition the diversity among the queries has not been taken into consideration, so that the MKL method can not apply to our case. One problem for most of the LTR approaches is that they employ a unified function as the ranking model according to the retrieved documents for all queries without considering the diversity among the queries. Queries can differ from each other according to their association and closeness to various clauses, which can result in a biased ranking model according to the queries. Instead, we aim to capture the semantic information contained in the queries according to the weights they assign to every clause and to reveal the characteristics of different queries. Recalling our example above, one query can be more related to movie actors while the other can give more weight to location information. This is, in fact, in line with the recent proposals of LTR [128, 129] which point out that the key for addressing the LTR problem is to look at it from the viewpoint of query. Thus a query-level stability and generalization is needed unlike most of the LTR approaches that assume and treat every data point in the training to be independent and identically distributed.

For this purpose we localize the weights of every clause kernel by making them query-specific. Actually, this results in a localized multiple kernel learning which has been introduced by Gönen et al. [83]. Instead of specifying the weight vector $\mathbf{d}$ which is identical for all the data points, it proposes a gating function to assign different weights to each base kernel for every particular data point. Formally, we define a query-dependent weighting function, $\eta_c : Q \to \mathbb{R}, q_i \in Q$, which assigns a value to every clause c. That is to say, a query q may prefer one clause rather than the other, where this preference is reflected in the weights $\{\eta_c(q)\}_{c \in H}$. Therefore, the multiple kernel in our LTR problem is defined as:

$$K(x_i, x_j) = \sum_{c \in H} \eta_c(q_i)\eta_c(q_j)K_c(x_i, x_j)$$

$$s.t. \sum_{c \in H} \eta_c(q_i) = 1, \ \forall q_i \in Q$$

where $x_i, \ x_j \in \mathcal{X}$ in the form of $x_i =< q_i, d_k, d_l >$, $x_j =< q_j, d_p, d_q >$, and K_c is a pairwise clause kernel of hypothesis H. Here the input data will be grouped by queries and $\eta_c(q_i)$ is the query-dependent weight function, which assigns the same query weight to all the input data belonging to one query group q_i. In this way, the query-dependent multiple kernel $K(x_i, x_j)$ takes not only the generality of different queries but also the specialty of each query into consideration: The generality refers to the extracted relevant features, i.e. clauses, in learning process, whereas the specialty refers to the learned

query weights that reflect the importance of each query as a combination of multiple kernels. The query-dependent weight function $\eta_c(q_i), q_i \in Q$ can be expressed as:

$$\eta_c(q_i) = \frac{exp(v_{ci})}{\sum_{c'} exp(v_{c'j})} \tag{7.8}$$

where c' denotes all of the clauses inclusive clause c, v_{ci} denotes the weight value for clause c w.r.t query q_i.

To solve the query dependent multiple kernel problem, a two-step alternating optimization algorithm can still be used as specified in Sec. 7.1.3. In the first step, the SVM coefficients $\{\alpha_i\}_{i=1}^n$ are optimized with $\eta_c(q_i)$ fixed. In the second step, the query-dependent function $\eta_c(q_i)$ is updated with $\{\alpha_i\}_{i=1}^n$ fixed. The dual form of this problem for the first step is:

$$\max_{\substack{\alpha \\ \sum_{i=1}^n \alpha_i y_i = 0, C \geq \alpha_i \geq 0}} -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_{i=1}^{n} \alpha_i$$

In the second step we replace the clause kernel weights d_c with $\eta_c(q_i)$ and optimize according to the following:

$$\max_{\substack{\eta_c(q_i) \\ \sum_{c \in H} \eta_c(q_i) = 1, \eta_c(q_i) \geq 0}} -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i^* \alpha_j^* y_i y_j K(x_i, x_j) + \sum_{i=1}^{n} \alpha_i^*$$

where α^* are the coefficient results got from the first optimization step. In the gradient descent optimization process, we take a simple differentiation of the dual function w.r.t. every v_{ci} as:

$$\frac{\partial J}{\partial v_{ci}} = -\frac{1}{2} \sum_{k,l=1}^{n} \sum_{c' \in H} \alpha_k^* \alpha_l^* y_k y_l \eta_{c'}(q_k) \eta_{c'}(q_l) K_{c'}(x_k, x_l)(2\delta(c = c') - \eta_c(x_k) - \eta_c(x_l))$$

and apply an updating scheme such as $v_{ci} \leftarrow v_{ci} + \gamma \frac{\partial J}{\partial v_{ci}}$.

Since we build the model based on the pairwise training data with the label only indicating whether the first document is preferred over the second, our final prediction function is also in the same form. This function can then only predict the preference between two documents of a given query in the test data. In other words, given a test data point $x_t =< q_t, d_{t1}, d_{t2} >$, the following function:

$$f(x_t) = \sum_{i=1}^{n} \alpha_i y_i \sum_{c \in H} \eta_c(q_i) \eta_c(q_t) K_c(x_i, x_t) \tag{7.9}$$

returns positive if d_{t1} is preferred more than d_{t2} and negative otherwise. However, in our LTR task we need to produce a ranked list of documents in descending order, aiming to find the relevant documents as much as possible. In order to achieve this, we develop an algorithm (i.e. Alg 5) that uses the Eq. 7.9 to generate a ranked list of documents. It is a greedy ordering algorithm that generates an approximate ordering in which every document is assigned a relevance value. After every document in the test data is assigned a relevance value, a ranked list is created according to their final score.

7.3 Experiments

We divide the experiments into two parts: In the first part we evaluate the SKM approach to evaluate its classification performance based on two different datasets. In the second part, we evaluate the LTR with semantic kernels by training a discriminative ranking function in an IR evaluation setting.

Algorithm 5: Order-by-Preference algorithm

Input: T // Test data set of documents

for each $d_i, d_j \in T$ and $d_i \neq d_j$ **do**

$\pi(d_i) \leftarrow \sum_{d_i \in V} PREF(d_i, d_j) - \sum_{d_i \in V} PREF(d_j, d_i)$

$Score.add(d_i, \pi(d_i))$

$bubble(Score)$

end for

Output: ordered list of $Score$

7.3.1 Classification Experiments

In the first experiment we evaluate our approach on the movie recommendation scenario based on the Filmtipset[1] dataset. It contains information about the user ratings of movies, the time of the ratings, the people starring in movies, directors, user profile information like home country, age, gender, and movie specific information like genre, comments etc. on movies or lists of similar movies. We randomly select 150 users (out of 1000) from this dataset and the movie ratings, (in our case *likes* or *dislikes*, which indicate the positive and negative examples), the movie information, location, and user profile information. This subset is divided into a training set with around 80 percent and a test set with around 20 percent of the examples. We are interested in predicting the predicate $like(?user, ?movie)$.

Accuracy. First, we focus on the prediction performance under different parameter settings. One way to measure this is using the accuracy defined as $ACC = \frac{(TP+TN)}{(P+N)}$ where *TP* are true positives, *TN* - true negatives, *P* - all positives and *N* - all negatives. Fig. 7.9a plots accuracy values in percent against different settings we used for co-evolution, i.e., for different numbers of generations and population sizes. We observe that accuracy improved both with increased number of generations and population size. However, at a certain point, incrementing these numbers does not yield much improvement. We consider this point to be domain and dataset specific and should be found out via experiments.

Accuracy Compared to Standard SVM. As discussed, a naive baseline would be to generate all possible hypotheses and to use SVM optimization to find the optimal parameters. An alternative is to use SVM optimization only to obtain SVM coefficients, but to use the evolutionary technique on a population of hypotheses to focus on the fittest ones. The former is clearly very inefficient. For brevity, we focus on the comparison of the latter with our co-evolutionary optimization. Against this baseline, we compared prediction quality as well as running time. The configuration we used for the following experiments is: 100 generations and 50 population size. The results shown in Fig. 7.9b indicate that accuracy of both approaches increased with the size of the dataset. We noticed that they generated different coefficients and different numbers of support vectors. Standard SVM aims to find a global optimum of the data points, resulting in a smaller number of support vectors compared to our approach. Despite this, accuracy results of both approaches were quite similar, suggesting that our approach was relatively good at finding optimal coefficients.

Running Time Compared to Standard SVM. However, differences in these approaches take influence on the running time. We distinguished running from prediction time. The training time of our approach and the SVM baseline can be observed in Fig. 7.9c. These results show that the co-evolution has significant improvements over standard SVM training. This effect is important when the dataset is large. However, standard SVM is slightly faster at prediction, as shown in Fig. 7.9d. This is due to the smaller number of support vectors used by the SVM, which means less complex computation is required for prediction.

[1] http://www.filmtipset.se/

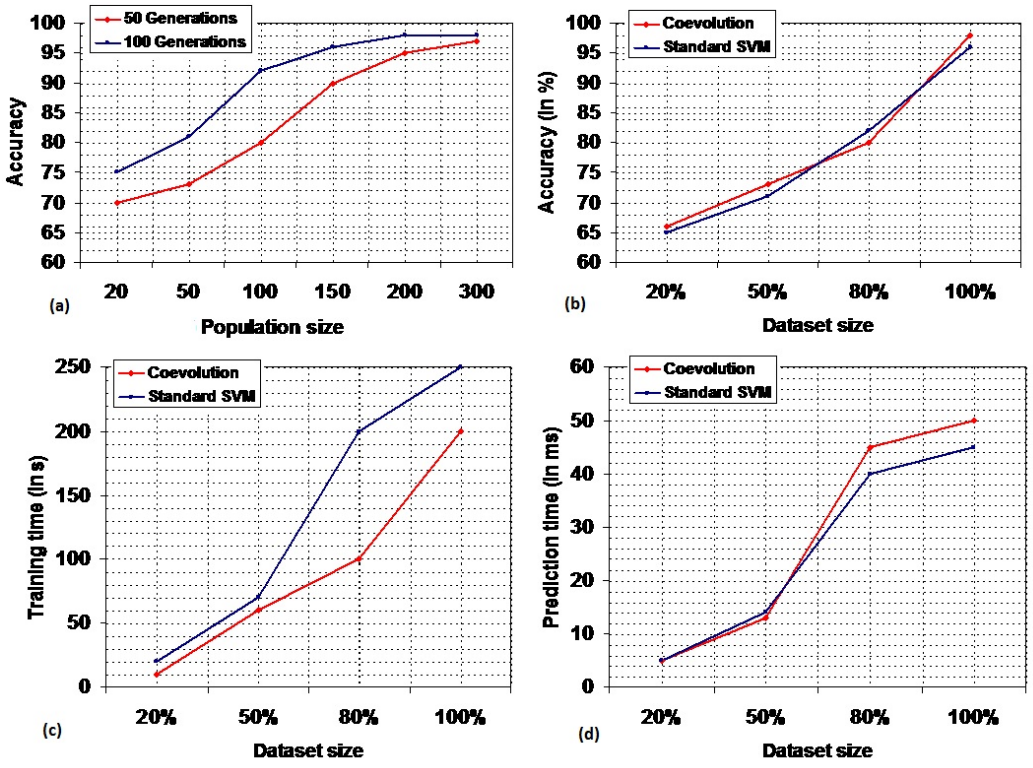


Figure 7.9: a) Accuracy of our approach, and b) accuracy c) training time and d) prediction time compared to SVM baseline

Kernel	Error	Precision	Recall	F
CCOP1	6.88	85.32	46.07	59.83
CCOP2	6.04	90.70	48.78	63.44
CCOPDP	4.49	95.87	57.27	71.71
RDFLearner	0.85	99.26	97.12	98.18

Table 7.1: SWRC experiment results

Comparison with Semantic Kernel. In this second dataset we compare our approach with related work on kernel-based SVM learning proposed for semantic data [29]. We use the same SWRC ontology[1] containing information about researchers, same task of predicting the affiliation of a person, the employed kernels predefined by the authors, the same metrics for comparison, and the same test setting. The employed kernels represent different combinations of edges, and their weights were set to the same values as defined by these authors. For our approach, optimization was performed based on 100 generations and 30 individuals in each subpopulation. We use recall for the loss function. Compared to the best baseline kernel, which was 4.49, 95.87, 57.27 and 71.71 in terms of error, precision, recall and F-measure, respectively, our approach yields large improvement. The result was 0.85, 99.26, 97.12, and 98.18, respectively. Thus, without relying on predefined features and weights, our approach was able to outperform the baseline. We discovered that these improved results are partially due to better feature selection. A larger impact on this positive performance may be attributed to MKL, which results in higher weights for more important features.

7.3.2 Ranking Experiments

Dataset. Standard LTR datasets such as LETOR [249] only contain pre-computed features and relevance judgements for query-document pairs without any information of regarding the document content or query topic. Therefore we built a new testbed on a recent large Web corpus *ClueWeb09-CatB* which consists of 50 million pages collected from the Web between Jan. 2009 and Feb. 2009. This dataset is also used by several tracks of the TREC conference [37]. We also picked 200 queries from the TREC 2009 Million Query track[2] with relevance judgements. The queries are classified into five distinct classes, namely *navigational*, *information*, *resource*, *advice*, and *list*. Then for each query pair, a 8-dimensional feature vector is created using the features shown in Table 7.2. These features are obtained from the index created by Terrier IR framework[3]. For structured data, we used DBPedia maintained in a Virtuoso RDF repository to be used by SKM for kernel calculations.

Evaluation Metric. We adapt the *Normalized Discounted Cumulative Gain (NDCG)* metric which is widely used to assess relevance in the context of IR and LTR [103]. In addition, we used mean average precision (MAP) as another metric over the selected queries. Average precision for each query is defined as the mean of the precision at n values calculated after each relevant documents was retrieved. The final MAP value is defined as the mean of average precisions of all queries in the test set. This metrics is the most commonly used single-value summary of a run over a set of queries. Detailed information about NDCG and MAP can be found in Appendix B.

Baselines. To evaluate the performance of our approach with SKM, we employ the following methods together with our approach (where two of the baselines are also query-dependent):

1. *RankSVM:* As a first baseline, we employ the RankSVM algorithm from [107] as it is the widely

[1] http://ontoware.org/projects/swrc/

[2] http://ir.cis.udel.edu/million/guidelines.html#queries

[3] http://terrier.org/

Feature	Feature Description	Feature	Feature Description
BM25	Okapi BM25 score for body text	Len(Body)	Body text length
TitleLM	Title-text language modeling (Jelinek-Mercer) score	TF	Body text term frequency
TitleMaxTF	Maximum query term in title text	BodyLM	Body text language model (Dirichlet) score
RQT	Ration of covered terms in title text	AvgNTF	Average normalized TF in body text
Len(URL)	Length of URL	TFIDF	TFIDF of whole document

Table 7.2: Conventional features used by the baseline methods.

compared baseline and trains a SVM with a single ranking function over all the training data using pairwise preferences.

2. *KNN-RSVM:* In this baseline, we apply a query-dependent "local-ranking" model based on the work presented in [77]. It classifies the queries into different categories by using the so-called query features in a K-Nearest Neighbor (KNN) clustering. Query features are obtained by averaging the features of the documents that are stated as relevant to each query. Based on such a hard partitioning of queries, separate local ranking model for each category using its own fraction of queries are trained using a RankingSVM method. Then during the testing we rank the documents of the test data point using these trained local model corresponding to its query.

3. *Topic-RSVM:* This method employs "*Topical RankingSVM*" a query dependent ranking model by considering the topics of the queries as described in [15]. Unlike the KNN-RSVM which utilizes a hard clustering such as KNN, it uses the mixture model as the clustering method to obtain rank-sensitive query topics. A probability function of topics w.r.t. the queries are estimated based on the mixture model and incorporated into one global ranking function combining many local models.

4. *RankSKM:* This is our method proposed in this chapter. Similar to KNN-RVM and Topic-RSVM, our method is also query-dependent. However, it utilizes the semantic kernels and trains a SKM model instead of conventional features with RankSVM training.

Results. In this experiment, we compare the relevance of different LTR methods according to the NDCG and MAP metrics. Fig. 7.10 shows the NDCG values of four methods, namely RankSVM, KNN-RSVM, Topic-RSVM and RankSKM. It is clearly observed from the figure that three query-dependent methods of KNN-RSVM, Topic-RSVM and RankSKM simply outperform the query-independent model learned by RankSVM on all the NDCG values. Furthermore, by using query-dependent semantic kernels, RankSKM performs better relevance scores than KNN-RSVM and Topic-RSVM which relies on the conventional features given in 7.2. We conduct the paired t-test with the significance at $p < 0.05$ on the improvements of RankSKM on NDCG@5 and NDCG@10 metrics over other ranking methods and observed that the improvements are statistically significant. In Fig. 7.11, we report the MAP scores of RankSKM compared with RankSVM, KNN-RSVM, and Topic-RSVM. It indicates that RankSKM achieves much better relevance than the other three in terms of the MAP scores. In particular, RankSKM improves the MAP by a gain of about 16% relative to RankSVM, 11% relative to KNN-RSVM, and 10% relative to Topic-RSVM. On the other hand, the biggest improvement among the baseline methods is about 7% (between Topic-RSVM and RankSVM).

Discussion. The results reveals the effectiveness of the query-dependent approaches over the query-independent ones. Table 7.3 shows the top-4 most important features for single model learned by RSVM and for four separate models corresponding to two query clusters by KNN-RSVM and two topics by Topic-RSVM, respectively. The feature importance is measured based on the absolute value of learned feature weights from the SVM model.

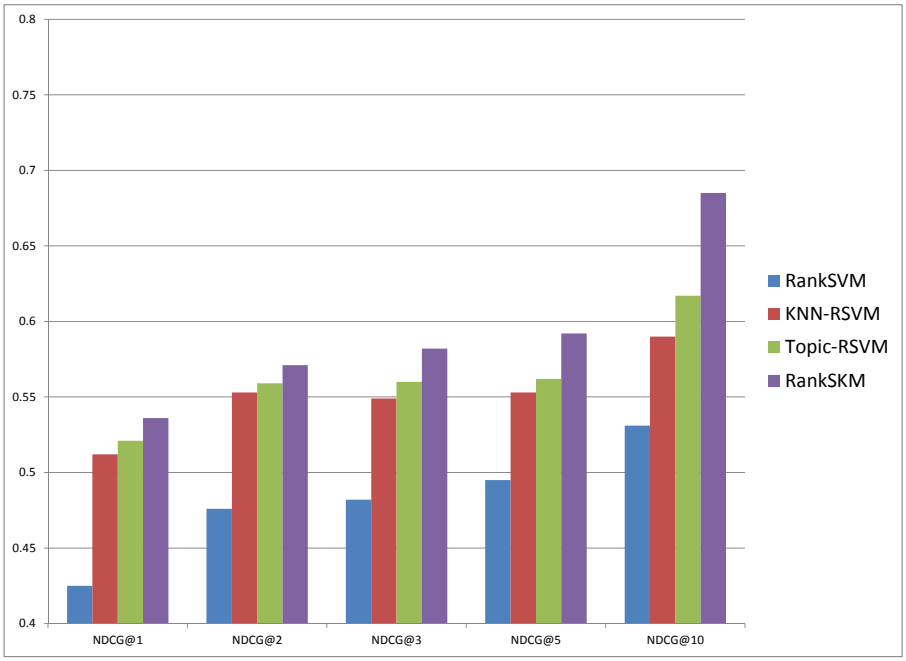


Figure 7.10: Ranking relevance in terms of NDCG@K of RankSKM compared with other methods on ClueWeb-CatB dataset.

Observing the features, we can see that query-dependent approaches (i.e. KNN-RSVM and Topic-RSVM) introduce multiple ranking models for different classes (cluster or topic) by assigning different level of importance to the features. This is in contrast to the single model learned by RSVM which describes a coarse summarization of ranking characteristics over all various queries. In particular, features that hold for a wide range of queries such as TF, TF-IDF are weighted higher in RSVM model. On the other hand, query-dependent models give more importance to the features depending on the nature of the queries. In particular, the title-specific features such as TitleMaxTF, RQT, TitleLM are weighted higher for the classes (e.g. C1) of navigational queries, whereas the features such as BM25, BodyLM are given more importance for the classes (e.g. C2, T2) of information queries. Thus, query-dependent models perform a more fine granular learning depending on the type of the query (e.g. navigational, information, resource etc.) and adapt the weights of features according to the training data coming from each of those classes. In addition, the main distinction between the KNN-RSVM and Topic-RSVM arises due to the hard query classification of KNN-RSVM as it builds multiple ranking models corresponding to each query class/cluster, however, many queries can fit into more than one classes/clusters. In this regard, Topic-RSVM partially address this issue by taking advantage of soft query classification but the improvement is low since both approaches try to generalize the query characteristics into the class the query belongs to. Here, the problem mainly arises due to the fact that the features used in those models to classify the query are not sufficient for an effective classification and queries belonging to the same class can actually refer to different concepts and the nature of the query is difficult to capture.

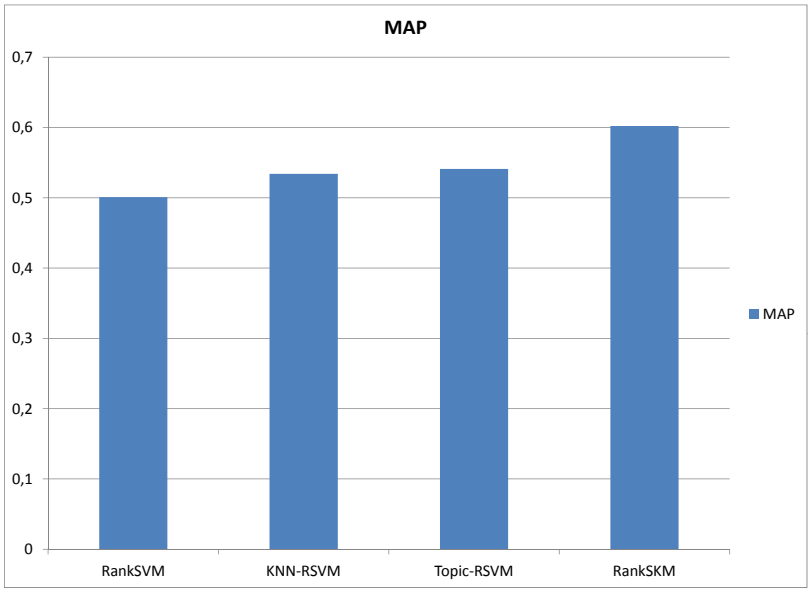


Figure 7.11: Ranking relevance in terms of MAP value of RankSKM compared with other methods on ClueWeb-CatB dataset.

In this regard, our RankSKM approach models each query q with a function $\boldsymbol{\eta}(q)$ which considers the query as mixture of clauses by assigning different weights to different clauses for each query. Since every clause is mapped to a semantic kernel, those kernels with high weights for the query become more discriminant for that query. This way RankSKM becomes more effective to capture the query characteristics. Table 7.4 list some of the clauses generated in our experiments and the associated queries from the query set that give highest weights to those clauses. It is clearly observed that queries with similar information needs are easily captured by the semantics of the clause which gives more accuracy in the ranking function.

In addition, one can draw an analogy between our intuition of mixture of clauses and mixture of topics of Topic-RSVM since both assume a query as a mixture model. However, there exists significant differences in detail: First, in Topic-RSVM topic probabilities w.r.t. the query is calculated with a mixture model as clustering method and pregiven to the actual RankSVM learning algorithm. Therefore those probabilities are not optimized according to the problem at hand (i.e. SVM problem), but assumed to be fixed based on some initial criteria. In other words, the topic probability assignments of queries and the actual learning problem stays highly decoupled in that way. Instead, we apply an embedded learning method using localized multiple kernel learning which optimizes the weight along with the actual learning problem and gives more exact weights between the queries and clauses. In addition, the number of the topics in the Topic-RSVM is fixed and pre-given (usually set to some small numbers from 1 to 10), while clauses are generated dynamically and selected as needed by the learning algorithm itself. Therefore, the proposed RankSKM approach is more powerful to capture query characteristics and to provide query-dependent models than the other three approaches.

RSVM	KNN-RSVM (C1)	KNN-RSVM (C2)	Topic-RSVM (T1)	Topic-RSVM (T2)
TF	RQT	BM25	TitleMaxTF	BodyLM
TitleMaxTF	TitleMaxTF	BodyLM	Len(URL)	TFIDF
TFIDF	TFIDF	TF	BM25	Len(URL)
Len(Body)	TitleLM	AvgNTF	TF	Len(Body)

Table 7.3: Top features for RSVM, KNN-RSVM, and Topic-RSVM.

Clause	$about(?d, ?o) \leftarrow Organization(?o), location(?o, ?l), Place(?l)$
Top queries	microsoft home, aircanada, wizz air, lego, auto parts, usairway, disneyland hotel, volvo, air travel information, usa today
Clause	$about(?d, ?l) \leftarrow Place(?l), country(?l, ?c), Country(?c)$
Top queries	map of the united states, workforce of utah, baltimorecity, kansas usa, starbucks, area codes,local vacation spots

Table 7.4: Some clauses generated for RankSKM and a list of queries for which $\eta_c(q)$ assigns the highest weight for that clause.

7.4 Conclusion

We presented the framework of SKM which considers the combined problem of feature selection and learning of combinations of multiple kernels. We show that existing techniques can be applied to this problem which however, yields high complexity. We thus proposed an co-evolutionary optimization which can jointly optimize these two dependent learning processes. This work facilitates the application of learning to semantic data, as it does not require experts to predefine features and to set weights, as opposed to existing approaches. Additionally, the experiments showed better results compared to previous approaches. Also, the co-evolutionary optimization exhibited good performance when compared to the naive application of SVM optimization.

In addition, we presented how SKM can be applied to the ranking problem to learn a ranking function that is able to use semantic information as features. For this, we adapt the framework 1) to learn from pairwise preferences instead of pointwise training data which is shown to be highly effective in LTR literature, 2) to utilize a query-dependent learning which distinguishes the learning depending on the query characteristics, 3) to show the effectiveness of SKM to capture the query characteristics as a set of clauses better than the existing query-dependent LTR approaches.

8

Conclusion

This thesis focuses on the subject of search relevance based on the semantic Web. To introduce the readers to this new concept, we provided a general discussion on search relevance in the field of IR and also clarified major differences between the semantic relevance and the existing types of relevance in the current state-of-the-art. In particular, we presented a general model for search relevance, which is further discussed in the context of semantic search. There is not one single definition of search relevance but rather semantic relevance shall be conceived as another notion of search relevance that employs a semantic model as its relevance source. Then, we particularly draw two unique hypotheses indicating that relevance can be determined using semantic models both in a generative and discriminative way. These are general hypotheses that can fit to both document and data retrieval. Based on the defined semantic relevance, we first introduced a Semantic Relevance Model for document retrieval that is focused on the retrieval of unstructured data by exploiting the existing semantic Web data. In order to bridge the gap between the structured and unstructured data this model relies on Topical Relational Model that is trained from an existing RDF-based dataset by utilizing the text and structured data under the same probabilistic graphical model.

In addition, the notion of relevance for data retrieval is also important and so far the existing approaches for ranking in this setting has been suboptimal. With the aim of improving this we developed a similar notion of relevance for data retrieval in order to obtain a more robust and precise ranking model for data retrieval. In particular, Edge-specific (Semantic) Relevance Models (ERM) provides an expanded and smoothed query model by exploiting the underlying data and structure offering a better way to represent the relevance. We also discussed how the proposed ranking model can be integrated into top-k query processing of the data retrieval in order to obtain top relevant results effectively. The results show that the proposed semantic relevance for data retrieval offers a principled approach to obtain highly effective results on various datasets.

Finally, we presented how to exploit the semantic Web data in a discriminative ranking model of learning-to-rank with semantic kernels. This model also shows that similar principles of the semantic relevance are still valid in a discriminative setting in which the existing training data is used to learn a ranking function using the exploited semantic kernels from the data. For this purpose, a specific learning method, namely Semantic Kernel Machines(SKM), which extends the well-established Support Vector Machines and multiple kernel learning, is developed. The unique characteristic of SKM to incorporate semantic information into the learning process as a number of semantic kernels is further exploited in our learning-to-rank approach. Using the semantic data to better represent the available training data (e.g. pairwise data points), our approach is able to train a query-dependent discriminative model for ranking.

Although the use of explicit semantic information within IR literature is a long-studied problem, an

effective and widely use of semantics to improve search performance is still an open challenge today. Especially, the emerging sematic Web data in large amounts emphasize the existence of a condensed, well-maintained semantic information and offers a new perspective to enable a widely available semantic model in order to improve the search relevance. However, with respect to the quality and applicability of the semantic information to improve search performance, the previous approaches of semantic search as reviewed in this thesis adopt notably varying points of view: The use of high level domain ontologies with rich expresiveness and structured query processing capabilities is challenged by the computational requirements and trade-off between the deterministic knowledge representation and uncertain nature of the Web. Notwithstanding, certain levels of assumptions are made in most cases such as the automation of semantic resource construction in terms of annotations or populating ontologies as well as some compromise on the scale and generality of the application scenarios (e.g. enterprise search, desktop search, or domain-specific search). In this regard, we take an important step to make use of already available semantic Web data and to extend well-founded IR-based techniques to incorporate this external knowledge and to obtain a comparable approach in terms of feasibility and generality.

Several interesting research directions can be taken to enhance the proposed approaches in this thesis. Among those we can include the following, notable research opportunities:

- *Learning-to-Rank for Data Retrieval.* The application of machine learning techniques to train ranking functions can also be applied to the setting of data retrieval with the availability of the data to train a discriminative model. In fact, such a data can easily be obtained from the logs of the major Web sites that utilize a keyword search capability over their underlying databases. In fact, the capabilities of our proposed learning algorithm, Semantic Kernel Machines, which trains a linear discriminative model from the graph-structured data can be adopted to apply learning-to-rank for data retrieval. In particular, this extension mainly benefits from constructing semantic kernels directly from structured data instead of finding linear feature sets to define and quantify structured search results.

- *Social Semantic Search.* As online communities such as Digg (websites), Twitter (status messages), Cite-U-Like (research papers) or Facebook have recently exploded in popularity, the paradigm of social search becomes a major focus of the current IR research. In fact, social search exploits a new sort of data emerging from these communities in the forms of user tags, or social network structure that provide useful source of information to improve search process with the active participation of the users. An interesting research direction can be to incorporate the additional data from online communities into our Semantic Relevance Model in order to better interpret users' information needs and accordingly to improve search performance. In this regard, semantic Web data provides a better understanding of the social network data (e.g. tags) by clustering the tags or users within certain topics. This extension also enables our proposed semantic search to focus not only to the semantic content but also to external data coming from online sources.

- *Crowdsourcing Semantic Search.* Crowdsourcing is a general term to describe harvesting the involvement of Web users into particular small tasks in order to accomplish bigger tasks traditionally performed by a small group of people. So far, crowdsourcing has been successfully applied to enable mass collaboration for some human-intelligence tasks such as Amazon's Mechanical Turk[1] (general tasks), reCaptcha[2] (digitizing books), or FreeBase[3] (knowledge representation). In fact, such a paradigm can also be used to improve the effectiveness of our proposed Semantic Relational Model approach, as it can be directly used to solve representational ambiguities

[1] https://www.mturk.com
[2] http://www.google.com/recaptcha
[3] http://www.freebase.com/

between the semantic model and search-specific artifacts (e.g. user queries, documents etc.). Currently, enriching the semantic model is already underway as Web sites such as FreeBase or Wikipedia are already using large user communities to improve their content. An interesting research direction of our work will be to also benefit from such a large user base to solve more search-related tasks instead of purely relying on algorithmic solutions as proposed in the early chapters of this thesis.

In conclusion, the use of semantic Web data for improving the search relevance is a fresh approach to avoid the drawbacks of the existing keyword-based search paradigm and also opens a new perspective to overcome the information overload problem on the Web. As the amount of publicly available semantic Web data increases on the Web, it provides an important source of information to be utilized for Web search.

Appendix A

Topical Relational Models

A.1 Joint Probability Distribution of TRM

The joint probability distribution of TRM given the parameters are specified as follows:

$$
\begin{aligned}
p(\mathbf{w}, \mathbf{c}, \mathbf{r}, \mathbf{b}, \boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{Z} \mid \alpha, \rho, \beta, \omega, \lambda) \quad = \quad & \prod_{t} \left[p(\pi_t \mid \alpha) \prod_{e} p(b_t(e) \mid \pi_t) \right] \\
& \prod_{e} \left[p(\boldsymbol{\theta}(e) \mid \mathbf{b}(e), \rho) \prod_{c} p(c(e) \mid \mathbf{b}(e), \lambda_c) \right] \\
& \prod_{e} \left[\prod_{v, <e,v> \in \mathcal{O}(w)} p(z(e,v) \mid \boldsymbol{\theta}(e)) p(w(e,v) \mid z(e,v), \beta_{1:K}) \right] \\
& \prod_{e,e'} \prod_{r} p(r(e,e') \mid \mathbf{b}(e), \mathbf{b}(e'), \omega_r)
\end{aligned} \tag{A.1}
$$

A.2 Variational Inference

Recall from Sec. 4.2.3 that our goal is to find a distribution q which is closest to the true distribution p in terms of $KL(q \parallel p)$ within the class of distributions representable as a product of independent marginals in the form of:

$$
\begin{aligned}
q(\mathbf{b}, \boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{Z} \mid \boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{\tau}) \quad = \quad & \prod_{t=1}^{K} \left[q(\pi_t \mid \tau_{t1}, \tau_{t2}) \prod_{e} q(b_t(e) \mid \nu_t(e)) \right] \\
& \prod_{e} \left[q(\boldsymbol{\theta}(e) \mid \boldsymbol{\gamma}(e)) \prod_{v, <e,v> \in \mathcal{O}(w)} q(z(e,v) \mid \boldsymbol{\phi}(e,v)) \right]
\end{aligned} \tag{A.2}
$$

The parameters $\boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{\tau}$ are variational parameters for Bernoulli, Dirichlet, Multinomial, and Beta distributions, respectively. The optimization problem of minimizing $KL(q \parallel p)$ can be rephrased in terms of maximizing variational lower bound as follows:

$$
\arg\min_{\{\tau,\nu,\gamma,\phi\}} KL(q \parallel p) = \arg\max_{\{\tau,\nu,\gamma,\phi\}} \mathcal{L}(q) \tag{A.3}
$$

where the full expression of $\mathcal{L}(q)$ is given in terms of certain expectations and entropies in Appendix A.3.

Solving this optimization using ordinary techniques can be quite difficult. However, when both distributions q and p are in the exponential family, each step in the coordinate ascent has a closed form according to [11, 234]. For example, if we are updating a variational parameter ε of q that corresponds to variable X, then the optimal ε are the solution to,

$$\ln q_\varepsilon(X_\varepsilon) = \mathrm{E}_{-\varepsilon}[\log p(\mathbf{X})] + c \tag{A.4}$$

where $\mathrm{E}_{-\varepsilon}$ represents the expectation taken over all variables $\mathbf{X}$ except X_ε according to variational distribution q. In other words, it says that the log of the optimal solution for the factor q_ε is obtained by simply considering the log of the joint distribution over all hidden and observed variables and then taking the expectation w.r.t. all of the other factors of q. We will use this approach to directly reach the update equations for variational parameters $\boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{\tau}$ in the following. We let c be a constant independent of the variable of interest that may change accordingly in every equation.

A.3 Variational Lower Bound

Variational lower bound introduced in Eq. 4.4 can be decomposed into:

$$\mathcal{L}(q) = \sum_{t=1}^{K} \mathcal{L}_t(q) + \sum_e \mathcal{L}_e(q) + \sum_r \mathcal{L}_r(q) + H[q] \tag{A.5}$$

where $H[q]$ is the entropy of the distribution q. We now expand each part of the lower bound by using the factorizations of p:

$$\mathcal{L}_t(q) = \mathrm{E}_q[\log p(\pi_t \mid \alpha)] \tag{A.6}$$

$$\begin{aligned}\mathcal{L}_e(q) = & \sum_{t=1}^{K} \mathrm{E}_q[\log p(b_t(e) \mid \pi_t)] + \sum_c \mathrm{E}_q[\log p(c(e) \mid \mathbf{b}(e), \boldsymbol{\lambda}_c)] + \\ & \mathrm{E}_q[\log p(\boldsymbol{\theta}(e) \mid \mathbf{b}(e), \rho)] + \sum_{v, <e,v> \in \mathcal{O}(w)} \mathrm{E}_q[\log p(z(e,v) \mid \boldsymbol{\theta}(e))] + \\ & \sum_{v, <e,v> \in \mathcal{O}(w)} \mathrm{E}_q[\log p(w(e,v) \mid \beta_{1:K}, z(e,v))]\end{aligned} \tag{A.7}$$

$$\mathcal{L}_r(q) = \sum_{e,e'} \mathrm{E}_q[\log p(r(e,e') \mid \mathbf{b}(e), \mathbf{b}(e'), \boldsymbol{\omega}_r)] \tag{A.8}$$

A.4 Derivation of Variational Parameter Updates

A.4.1 Expectations

We derive the expressions for each expectation in Equations A.6, A.7, and A.8.

A.4.1.1 Computing $\mathbf{E}_q[\log p(\pi_t \mid \alpha)]$

Each stick in IBP is independent, and substituting the form of the beta prior we get:

$$\begin{aligned}\mathrm{E}_q[\log p(\pi_t \mid \alpha)] &= \mathrm{E}_q[\log(\frac{\alpha}{K}\pi_t^{\frac{\alpha}{K}-1})] \\ &= \log\frac{\alpha}{K} + (\frac{\alpha}{K}-1)\mathrm{E}_q[\log(\pi_t)] \\ &= \log\frac{\alpha}{K} + (\frac{\alpha}{K}-1)(\psi(\tau_{t1}) - \psi(\tau_{t1}+\tau_{t2}))\end{aligned}$$

where $\psi(.)$ is the digamma function.

A.4.1.2 Computing $\mathbf{E}_q[\log p(b_t(e) \mid \pi_t)]$

For the feature assignments, which are Bernoulli-distributed, we derive the expectation as follows:

$$\begin{aligned}\mathrm{E}_q[\log p(b_t(e) \mid \pi_t)] &= \mathrm{E}_q[\log p(b_t(e)=1 \mid \pi_t)^{b_t(e)} p(b_t(e)=0 \mid \pi_t)^{1-b_t(e)}] \\ &= \mathrm{E}_q[b_t(e)]\mathrm{E}_q\left[\log\pi_t\right] + \mathrm{E}_q[1-b_t(e)]\mathrm{E}_q\left[\log(1-\pi_t)\right] \\ &= \nu_t(e)\psi(\tau_{t1}) - \psi(\tau_{t1}+\tau_{t2}) + (1-\nu_t(e))\psi(\tau_{t2})\end{aligned}$$

where $\mathrm{E}_q[b_t(e)] = \nu_t(e)$.

A.4.1.3 Computing $\mathbf{E}_q[\log p(r(e,e') \mid \mathbf{b}(e), \mathbf{b}(e'), \boldsymbol{\omega}_r)]$

The relationship probability between two entities is modeled as a response variable which is a generalized linear model (GLM) [159] with a linear predictor $\mathbf{b}(e)^T\boldsymbol{\omega}_r\mathbf{b}(e')$. The link function σ is the sigmoid and depends on $\mathbf{b}(e)$ and $\mathbf{b}(e')$ and we can approximate the expectation using the multivariate delta method for moments [32] for a first-order approximation as follows:

$$\begin{aligned}\mathrm{E}_q[\log p(r(e,e') \mid \mathbf{b}(e), \mathbf{b}(e'), \boldsymbol{\omega}_r)] &= \mathrm{E}_q\left[\log\sigma(\mathbf{b}(e)^T\boldsymbol{\omega}_r\mathbf{b}(e'))\right] \\ &\approx \log\sigma(\mathrm{E}_q\left[\mathbf{b}(e)\right]\boldsymbol{\omega}_r\mathrm{E}_q\left[\mathbf{b}(e')\right]) \\ &= \log\sigma(\boldsymbol{\nu}(e)^T\boldsymbol{\omega}_r\boldsymbol{\nu}(e')))\end{aligned}$$

where $\boldsymbol{\omega}_r$ is relationship-specific model parameter and $\mathrm{E}_q[\mathbf{b}(e)] = \nu(e)$.

A.4.1.4 Computing $\mathbf{E}_q[\log p(c(e) \mid \mathbf{b}(e), \boldsymbol{\lambda}_c)]$

Similarly, the concept probability of an entity is modeled as another response variable with a linear predictor $\boldsymbol{\lambda}_c^T\mathbf{b}(e)$ and expectation is approximated as follows:

$$\begin{aligned}\mathrm{E}_q[\log p(c(e) \mid \mathbf{b}(e), \boldsymbol{\lambda}_c)] &= \mathrm{E}_q\left[\log\sigma(\boldsymbol{\lambda}_c^T\mathbf{b}(e))\right] \\ &\approx \log\sigma(\boldsymbol{\lambda}_c^T\mathrm{E}_q\left[\mathbf{b}(e)\right]) = \log\sigma(\boldsymbol{\lambda}_c^T\boldsymbol{\nu}(e)))\end{aligned}$$

where $\boldsymbol{\lambda}_c$ is concept-specific model parameter and $\mathrm{E}_q[\mathbf{b}(e)] = \nu(e)$.

A.4.1.5 Computing $\mathrm{E}_q[\log p(\boldsymbol{\theta}(e) \mid \mathbf{b}(e), \rho)]$

Feature specific topic proportions are dependent on $\mathbf{b_e}$ and ρ. The expectation is derived as follows:

$$\begin{aligned}
\mathrm{E}_q[\log p(\boldsymbol{\theta}(e) \mid \mathbf{b}(e), \rho)] &= \mathrm{E}_q\left[\log \frac{\Gamma(\sum_{t=1}^{K} \rho b_t(e))}{\prod_{t=1}^{K} \Gamma(\rho b_t(e))} \prod_{t=1}^{K} \theta_t(e)^{\rho b_t(e)-1}\right] \\
&= \sum_{t=1}^{K} \rho \mathrm{E}_q[b_t(e)]\mathrm{E}_q[\log \theta_t(e)] + \mathrm{E}_q[\Gamma(\sum_{t=1}^{K} \rho b_t(e))] - \sum_{t=1}^{K} \mathrm{E}_q[\Gamma(\rho b_t(e))] \\
&\propto \sum_{t=1}^{K} \rho \nu_t(e)(\psi(\gamma_t(e)) - \psi(\sum_{t'} \gamma_{t'}(e)))
\end{aligned}$$

A.4.1.6 Computing $\mathrm{E}_q[\log p(z(e, v) \mid \boldsymbol{\theta}(e))]$

Similar to LDA [28], topic-word random variables are multinomial distribution with the parameter $\boldsymbol{\theta}(e)$ and the expectation is computed as follows:

$$\begin{aligned}
\mathrm{E}_q[\log p(z(e, v) \mid \boldsymbol{\theta}(e))] &= \mathrm{E}_q[\log \prod_{t=1}^{K} \theta_t(e)^{z(e,v)}] \\
&= \sum_{t=1}^{K} \phi_t(e, n)\mathrm{E}_q[\log \theta_t(e)] \\
&= \sum_{t=1}^{K} \phi_t(e, n)(\psi(\gamma_t(e)) - \psi(\sum_{t'} \gamma_{t'}(e)))
\end{aligned}$$

where $q(z(e, v) = t \mid \phi(e, v)) = \phi_t(e, n)$.

A.4.1.7 Computing $\mathrm{E}_q[\log p(w(e, v) \mid \beta_{1:K}, z(e, v))]$

The expectation of observed word probabilities are derived as follows:

$$\begin{aligned}
\mathrm{E}_q[\log p(w(e, v) \mid \beta_{1:K}, z(e, v))] &= \mathrm{E}_q[\log \prod_{t=1}^{K} \beta_{tw(e,v)}^{z(e,v)}] \\
&= \sum_{t=1}^{K} \phi_t(e, v)\log \beta_{tw(e,v)}
\end{aligned}$$

A.4.1.8 Computing $H[q]$

Finally, the entropy $H[q]$ of the distribution q is given by:

$$\begin{aligned} H[q] &= -\mathrm{E}_q\left[\log\left(\prod_{t=1}^{K} q_{\tau_t}(\pi_t)\prod_{t=1}^{K}\prod_{e=1}^{E} q_{\nu_t(e)}(b_t(e))\prod_{e=1}^{E} q_{\gamma(e)}(\theta(e))\prod_{e=1}^{E}\prod_{v} q_{\phi(e,v)}(z(e,n))\right)\right] \\ &= \sum_{t=1}^{K}\mathrm{E}_q\left[-\log q_{\tau_t}(\pi_t)\right] + \sum_{t=1}^{K}\sum_{e=1}^{E}\mathrm{E}_q\left[-\log q_{\nu_t(e)}(b_t(e))\right] + \\ &\quad \sum_{e=1}^{E}\mathrm{E}_q\left[-\log q_{\gamma(e)}(\theta(e))\right] + \sum_{e=1}^{E}\sum_{n=1}^{N}\mathrm{E}_q\left[-\log q_{\phi(e,v)}(z(e,n))\right] \end{aligned}$$

where:

$$\begin{aligned} \mathrm{E}_q\left[-\log q_{\tau_t}(\pi_t)\right] &= \log\left(\frac{\Gamma(\tau_{t1})\Gamma(\tau_{t2})}{\Gamma(\tau_{t1}+\tau_{t2})}\right) - (\tau_{t1}-1)\psi(\tau_{t1}) - (\tau_{t2}-1)\psi(\tau_{t2}) + \\ &\quad (\tau_{t1}+\tau_{t2}-2)\psi(\tau_{t1}+\tau_{t2}) \end{aligned}$$

and

$$\mathrm{E}_q\left[-\log q_{\nu_t(e)}(b_t(e))\right] = -\nu_t(e)\log\nu_t(e) - (1-\nu_t(e))\log(1-\nu_t(e))$$

$$\mathrm{E}_q\left[-\log q_{\gamma(e)}(\theta(e))\right] \propto -\sum_{t=1}^{K}(\gamma_t(e)-1)(\psi(\gamma_t(e)) - \psi(\sum_{t'=1}^{K}\gamma_{t'}(e)))$$

$$\mathrm{E}_q\left[-\log q_{\phi(e,v)}(z(e,n))\right] = \sum_{t=1}^{K}\phi_t(e,v)\log\phi_t(e,v)$$

A.4.2 Updates

1. For the updates of the variational parameters τ_{t1} and τ_{t2} of Beta-distributed IBP prior, Eq. A.4 holds for the optimal values of τ_{t1} and τ_{t2} as follows:

$$\begin{aligned} \ln q_{\tau_t}(\pi_t) &= \mathrm{E}_q\left[\log p(\pi_t \mid \alpha) + \sum_{e\in\mathcal{O}(\mathbf{b})}\log p(b_t(e)\mid\pi_t)\right] + c \\ &= (\frac{\alpha}{K}-1)\log\pi_t + \sum_{e\in\mathcal{O}(\mathbf{b})}\nu_t(e)\log\pi_t + (1-\nu_t(e))\log(1-\pi_t) + c \end{aligned}$$

and the updates are:

$$\tau_{t1} = \frac{\alpha}{K} + \sum_{e\in\mathcal{O}(\boldsymbol{b})}\nu_t(e)$$

$$\tau_{t2} = 1 + |\mathcal{O}(\boldsymbol{b})| - \sum_{e\in\mathcal{O}(\boldsymbol{b})}\nu_t(e)$$

2. For the updates of the variational parameters $\nu_t(e)$ of Bernoulli-distributed topic-indicators, we cannot directly use Eq. A.4 due to the logistic regression terms. Instead we collect the associated terms of $\nu_t(e)$ from Eq. A.5 as follows:

$$\begin{aligned}\mathcal{L}_\nu &= \mathrm{E}_q\left[\log p(b_t(e) \mid \pi_t) + \log p(\boldsymbol{\theta(e)} \mid \mathbf{b}(e), \rho) + \sum_{c \in C(e)} \log p(c(e) \mid \mathbf{b}(e), \lambda_c)\right] \\ &+ \mathrm{E}_q\left[\sum_{e', r(e,e')} \log p(r(e,e') \mid \mathbf{b}(e), \mathbf{b}(e'), \boldsymbol{\omega}_r) + \sum_{e', r(e',e)} \log p(r(e',e) \mid \mathbf{b}(e), \mathbf{b}(e'), \boldsymbol{\omega}_r)\right] \\ &- \mathrm{E}_q\left[\log q_{\nu_t(e)}(b_t(e))\right]\end{aligned}$$

and replacing the expectations,

$$\begin{aligned}\mathcal{L}_\nu &= \nu_t(e)(\psi(\tau_{t1}) - \psi(\tau_{t2}) + \nu_t(e)\rho\,(\psi(\gamma_t(e)) - \psi(\sum_{t'} \gamma_{t'}(e))) + \sum_{c \in C(e)} \log \sigma(\boldsymbol{\lambda}_c^T \boldsymbol{\nu}(e))) \\ &+ \sum_{e', r(e,e')} \log \sigma(\boldsymbol{\nu}(e)^T \boldsymbol{\omega}_r \boldsymbol{\nu}(e'))) + \sum_{e', r(e',e)} \log \sigma(\boldsymbol{\nu}(e')^T \boldsymbol{\omega}_r \boldsymbol{\nu}(e))) \\ &- -\nu_t(e)\log\nu_t(e) - (1 - \nu_t(e))\log(1 - \nu_t(e))\end{aligned}$$

Taking the derivatives w.r.t. $\nu_t(e)$ and setting equal to zero leads to the following update:

$$\nu_t(e) = \frac{1}{1 + e^{-\vartheta}}$$

where ϑ is:

$$\vartheta_t(e) = \Psi(\tau_{t1}) - \Psi(\tau_{t2}) + \sum_{c \in C(e)} \vartheta_c + \sum_{r(e,e')} \vartheta_{r1} + \sum_{r(e',e)} \vartheta_{r2} + \vartheta_\gamma$$

The partial updates for ϑ_c, ϑ_{r1}, ϑ_{r2}, and ϑ_γ is given in Eq. 4.12, 4.13, 4.14, and 4.15, respectively.

3. For the optimal of variational parameter $\gamma_t(e)$ of Dirichlet-distributed topic-proportions, the following holds,

$$\begin{aligned}\ln q_{\gamma_t}(\theta_t(e)) &= \mathrm{E}_q\left[\log p(\boldsymbol{\theta}(e) \mid \mathbf{b}(e), \rho) + \sum_{v, <e,v> \in \mathcal{O}(w)} \log p(z(e,v) \mid \boldsymbol{\theta}(e))\right] \\ &= \rho\nu_t(e)\,\log\theta_t(e) + \sum_{v, <e,v> \in \mathcal{O}(w)} \phi_t(e,v)\log\theta_t(e)\end{aligned}$$

and the update for $\gamma_t(e)$ is obtained as:

$$\gamma_t(e) = \rho\nu_t(e) + \sum_{v, <e,v> \in \mathcal{O}(w)} \phi_t(e,v)$$

4. Finally, the optimal of the variational parameter γ_t satisfies,

$$\ln q_{\phi_t}(z_t(e,v)) = \mathrm{E}_q\left[\log p(z(e,v) \mid \boldsymbol{\theta}(e)) + \log p(w(e,v) \mid \beta_{1:K}, z(e,v)\right] \quad \text{(A.9)}$$

$$= z_t(e,v)\log\beta_{tv} + z_t(e,v)\left[\Psi(\gamma_t(e)) - \Psi(\sum_{t'}\gamma_{t'}(e))\right] \quad \text{(A.10)}$$

which leads to,

$$\phi_t(e,v) \propto \exp\{\log\beta_{tv} + \Psi(\gamma_t(e)) - \Psi(\sum_{t'}\gamma_{t'}(e))\}$$

A.5 Regularization of Logistic Regression Parameters

The parameter updates for $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ are obtained by taking the gradient of Eq. A.5 with respect to each parameter. As the expectations of the logarihm of the sigmoid function depends linearly on the topic indicator vector $\mathbf{b}$ for both class and relationship regression in Appendix A.4.1.4 and A.4.1.3 respectively, these updates take a convenient form as follows:

$$\bigtriangledown_{\lambda_{ct}} = \sum_{e}(1 - \sigma(\boldsymbol{\lambda}_c^T\boldsymbol{\nu}(e)))\nu_t(e) \quad \text{(A.11)}$$

and

$$\bigtriangledown_{\omega_{rtt'}} = \sum_{e,e'}(1 - \sigma(\boldsymbol{\nu}(e)^T\boldsymbol{\omega}_r\boldsymbol{\nu}(e')))\omega_{rtt'}\nu_{t'}(e') \quad \text{(A.12)}$$

However, these gradients are always positive since they are defined only when a class or relationship exist for the entity. This leads the learning algoritm to diverge. For class parameters $\boldsymbol{\lambda}$, we address this issue by assuming a fixed number ε of observations (e.g. entities) for which the class membership is observed to be zero (i.e. $c(e) = 0$). We also associate with these observations an expected topic-indicator vector $\overline{\boldsymbol{\pi}}$ which is set to be the mean of Beta-distributed variable π, $\overline{\boldsymbol{\pi}} = \frac{\boldsymbol{\tau}_1}{\boldsymbol{\tau}_1+\boldsymbol{\tau}_2}$. This leads to the following regularization for $\boldsymbol{\lambda}_c$:

$$\mathfrak{R}_{\boldsymbol{\lambda}_c} = -\varepsilon(\sigma(\boldsymbol{\lambda}_c^T\overline{\boldsymbol{\pi}}))\overline{\boldsymbol{\pi}}$$

Similarly, we assume a fixed number ζ of observations where a particular relationship is observed to be zero (i.e. $r(e,e') = 0$) and add the following regularization term for $\boldsymbol{\omega}_r$:

$$\mathfrak{R}_{\boldsymbol{\omega}_r} = -\zeta(\sigma(\overline{\boldsymbol{\pi}}^T\boldsymbol{\omega}_r\overline{\boldsymbol{\pi}})\overline{\boldsymbol{\pi}}$$

Here, ε and ζ is set during the experiments.

Appendix B

Performance Measures of IR

In the literature of IR, many different performance and correctness measures have been proposed with the use of relevance judgements. Most of these measures are defined for each query by taking a ranked list π of results and relevance judgements as parameters of the measure functions. In most cases, the query-specific measures are averaged over all the queries in the evaluation dataset to obtain cumulative results. Throughout the thesis, three of these measures, namely *Mean Reciprocal Rank* (MRR) [233], *Mean Average Precision* (MAP) [259], and *Normalized Discounted Cumulative Gain* (NDCG) [103], are widely used. Here we briefly introduce these three measures.

B.1 Mean Reciprocal Rank

Mean Reciprocal Rank (MRR) is a measure that assumes the existence of only one relevant document in the ranked list π. Then we may want to measure the quality of π by how long it takes to find this particular, relevant document. If, for each query q, the rank position of its first relevant document is denoted as r_q, then MRR can be defined as the average of the reciprocal ranks of results for a selection of queries Q (where $q \in Q$) as follows:

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{r_q}$$

B.2 Mean Average Precision

To define Mean Average Precision (MAP), we need to first define *Precision@k* which is a measure for evaluating top-k positions of a ranked list π using two levels (relevant and irrelevant) of relevance judgements:

$$Precision@k = \frac{1}{k} \sum_{i=1}^{k} r_i$$

where k denotes truncation position and r_i is one if the result at the position i is relevant, and zero otherwise. Then *Average Precision* (AP) is defined on the basis of Precision@k as follows:

$$AP(q) = \frac{1}{|D_q|} \sum_{j} r_j \times Precision@j$$

where D_q is the number of relevant documents w.r.t. the query q. Finally MAP is defined as the mean of AP over all the queries in Q:

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} AP(q)$$

B.3 Normalized Discounted Cumulative Gain

For a ranked list of documents, the NDCG score at position n is calculated as:

$$NDCG(n) = Z_n \sum_{j=1}^{n} \frac{2^{r_j} - 1}{log(j+1)}$$

where j is the position in the ranked list, r_j is the relevance of document at the position j, and Z_n is the normalization factor which is chosen to be the maximum possible value so that the NDCG takes values from 0 to 1.

References

[1] A. Abecker, V. Bicer, W. Kazakos, G. Nagypal, R. Nedkov, and A. Valikov. **User-Friendly Access to Structured Environmental Data**. *Environmental Software Systems. Frameworks of eEnvironment*, pages 357–363, 2011.

[2] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, P. Parag, and S. Sudarshan. **Banks: Browsing and keyword searching in relational databases**. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 1083–1086. VLDB Endowment, 2002.

[3] E. Agichtein, E. Brill, and S. Dumais. **Improving web search ranking by incorporating user behavior information**. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 19–26. ACM, 2006.

[4] S. Agrawal, S. Chaudhuri, and G. Das. **DBXplorer: A system for keyword-based search over relational databases**. In *Proceedings of the 18th International Conference on Data Engineering (ICDE)*, pages 5–16. IEEE, 2002.

[5] B. Amento, L. Terveen, and W. Hill. **Does "authority" mean quality? Predicting expert quality ratings of Web documents**. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 296–303. ACM, 2000.

[6] S. Amer-Yahia and M. Lalmas. **XML Search: Languages, INEX and Scoring**. *ACM SIGMOD Record*, **35**(4):16–23, 2006.

[7] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. **DBpedia: A nucleus for a web of open data**. *Lecture Notes in Computer Science*, **4825**:722, 2007.

[8] F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan. **Multiple kernel learning, conic duality, and the SMO algorithm**. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

[9] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.

[10] C.L. Barry. **User-defined relevance criteria: an exploratory study**. *Journal of American Society for Information Sciences*, pages 149–159, 1994.

[11] M.J. Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

[12] M. Bender, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J.X. Parreira, R. Schenkel, and G. Weikum. **Exploiting social relations for query expansion and result ranking**. In *IEEE 24th*

International Conference on Data Engineering Workshop, 2008 (ICDEW 2008), pages 501–506. IEEE.

[13] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. **Keyword searching and browsing in databases using BANKS**. In *Proceedings of the 18th International Conference on Data Engineering (ICDE)*, pages 431–440. IEEE, 2002.

[14] J. Bhogal, A. MacFarlane, and P. Smith. **A review of ontology based query expansion**. *Information processing & management*, **43**(4):866–886, 2007.

[15] J. Bian, X. Li, F. Li, Z. Zheng, and H. Zha. **Ranking specialization for web search: a divide-and-conquer approach by using topical RankSVM**. In *Proceedings of the 19th international conference on World wide web*, pages 131–140. ACM, 2010.

[16] V. Bicer, A. Abecker, T. Tran, and R. Nedkov. **KOIOS: Utilizing Semantic Search for Easy-Access and Visualization of Structured Environmental Data**. In *Proceedings of the 10th International Semantic Web Conference (ISWC)*. Springer, 2011.

[17] V. Bicer, O. Kilic, A. Dogac, and G.B. Laleci. **Archetype-based semantic interoperability of web service messages in the health care domain**. *International Journal on Semantic Web and Information Systems (IJSWIS)*, **1**(4):1–23, 2005.

[18] V. Bicer, G.B. Laleci, A. Dogac, and Y. Kabak. **Artemis message exchange framework: semantic interoperability of exchanged messages in the healthcare domain**. *ACM SIGMOD Record*, **34**(3):71–76, 2005.

[19] V. Bicer, G.B. Laleci, A. Dogac, and Y. Kabak. **Providing semantic interoperability in the healthcare domain through ontology mapping**. In *Proceedings of eChallenges*, 2005.

[20] V. Bicer, T. Tran, and A. Gossen. **Relational Kernel Machines for Learning from Graph-Structured RDF Data**. In *Proceedings of the 8th Extended Semantic Web Conference (ESWC)*, ESWC'11, pages 47–62, Berlin, Heidelberg, 2011. Springer-Verlag.

[21] V. Bicer, T. Tran, and R. Nedkov. **Ranking Support for Keyword Search on Structured Data using Relevance Models**. *Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM)*, 2011.

[22] K. Bischoff, C.S. Firan, W. Nejdl, and R. Paiu. **Can all tags be used for search?** In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 193–202. ACM, 2008.

[23] C.M. Bishop. *Pattern recognition and machine learning*. Springer New York, 2006.

[24] Christian Bizer, Tom Heath, and Tim Berners-Lee. **Linked Data - The Story So Far**. *Int. J. Semantic Web Inf. Syst.*, **5**(3):1–22, 2009.

[25] D. Blei. **Introduction to Probabilistic Topic Models**. *Technical Report, Princeton University*, 2011.

[26] D.M. Blei and J.D. Lafferty. **Topic models**. *Text mining: classification, clustering, and applications*, page 71, 2009.

[27] D.M. Blei and J. McAuliffe. **Supervised topic models**. *Advances in Neural Information Processing Systems*, **20**:121–128, 2008.

[28] D.M. Blei, A.Y. Ng, and M.I. Jordan. **Latent dirichlet allocation**. *The Journal of Machine Learning Research*, **3**:993–1022, 2003.

[29] Stephan Bloehdorn and York Sure. **Kernel Methods for Mining Instance Data in Ontologies**. In *Proceedings of the 6th International Semantic Web Conference (ISWC) and 2nd Asian Semantic Web Conference (ASWC)*, pages 58–71, 2007.

[30] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. **Freebase: a collaboratively created graph database for structuring human knowledge**. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM, 2008.

[31] P. Borlund. **The concept of relevance in IR**. *Journal of the American Society for information Science and Technology*, **54**(10):913–925, 2003.

[32] M. Braun and J. McAuliffe. **Variational inference for large-scale models of discrete choice**. *Journal of the American Statistical Association*, **105**(489):324–335, 2010.

[33] S. Brin and L. Page. **The anatomy of a large-scale hypertextual Web search engine**. *Computer networks and ISDN systems*, **30**(1-7):107–117, 1998.

[34] R. Bunescu and M. Pasca. **Using encyclopedic knowledge for named entity disambiguation**. In *Proceedings of EACL*, **6**, pages 9–16, 2006.

[35] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. **Learning to rank using gradient descent**. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[36] M.J. Cafarella, A. Halevy, and N. Khoussainova. **Data integration for the relational web**. *Proceedings of the VLDB Endowment*, **2**(1):1090–1101, 2009.

[37] J. Callan, M. Hoy, C. Yoo, and L. Zhao. **Clueweb09 data set**. *boston. lti. cs. cmu. edu, Jan*, 2009.

[38] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. **Selecting good expansion terms for pseudo-relevance feedback**. In *SIGIR*, pages 243–250, 2008.

[39] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. **Learning to rank: from pairwise approach to listwise approach**. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM.

[40] P. Castells, M. Fernandez, and D. Vallet. **An adaptation of the vector-space model for ontology-based information retrieval**. *IEEE Transactions on Knowledge and Data Engineering*, pages 261–272, 2007.

[41] P. Castells, B. Foncillas, R. Lara, M. Rico, and J.L. Alonso. **Semantic web technologies for economic and financial information management**. *The Semantic Web: Research and Applications*, pages 473–487, 2004.

[42] J. Chang, J. Boyd-Graber, and D.M. Blei. **Connections between the lines: augmenting social networks with text**. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–178. ACM, 2009.

[43] K.C.C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. **Structured databases on the web: Observations and implications**. *ACM SIGMOD Record*, **33**(3):61–70, 2004.

[44] C. Chemudugunta, A. Holloway, P. Smyth, and M. Steyvers. **Modeling documents by combining semantic concepts with unsupervised statistical learning**. *The Semantic Web-ISWC 2008*, pages 229–244.

[45] P.A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter. **Using ODP metadata to personalize search**. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 185. ACM, 2005.

[46] M. Claypool, D. Brown, P. Le, and M. Waseda. **Inferring user interest**. *IEEE Internet Computing*, pages 32–39, 2001.

[47] Joel Coffman and Alfred C. Weaver. **A framework for evaluating database keyword search strategies**. In *CIKM*, pages 729–738, 2010.

[48] W.S. Cooper. **A definition of relevance for information retrieval* 1**. *Information storage and retrieval*, **7**(1):19–37, 1971.

[49] W.S. Cooper. **On selecting a measure of retrieval effectiveness**. *Journal of the American Society for Information Science*, **24**(2):87–100, 1973.

[50] E. Cosijn and P. Ingwersen. **Dimensions of relevance**. *Information Processing & Management*, **36**(4):533–550, 2000.

[51] D. Cossock and T. Zhang. **Subset ranking using regression**. *Learning Theory*, pages 605–619, 2006.

[52] K. Crammer and Y. Singer. **Pranking with ranking**. *Advances in neural information processing systems*, **14**:641–647, 2001.

[53] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. **On kernel target alignment**. *Innovations in Machine Learning*, pages 205–256, 2006.

[54] W.B. Croft. **User-specified domain knowledge for document retrieval**. In *Proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 201–206. ACM, 1986.

[55] S. Cucerzan. **Large-scale named entity disambiguation based on Wikipedia data**. In *Proceedings of EMNLP-CoNLL*, **2007**, pages 708–716, 2007.

[56] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. **A framework and graphical development environment for robust NLP tools and applications**. In *ACL 2002*, 2002.

[57] C. da Costa Pereira, M. Dragoni, and G. Pasi. **Multidimensional relevance: Prioritized aggregation in a personalized Information Retrieval setting**. *Information Processing & Management*, 2011.

[58] Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. **Classification and Retrieval through Semantic Kernels**. In *KES (3)*, pages 252–259, 2008.

[59] A. Damian, W. Nejdl, and R. Paiu. **Peer-Sensitive ObjectRank–Valuing Contextual Information in Social Networks**. *Web Information Systems Engineering–WISE 2005*, pages 512–519, 2005.

[60] Luc De Raedt and Sašo Džeroski. **First-order jk-clausal theories are PAC-learnable**. *Artif. Intell.*, **70**(1-2):375–392, 1994.

[61] H. Deng, J. Han, B. Zhao, Y. Yu, and C.X. Lin. **Probabilistic topic models with biased propagation on heterogeneous information networks**. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1271–1279. ACM, 2011.

[62] F. Diaz. **Regularizing ad hoc retrieval scores**. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 672–679. ACM, 2005.

[63] F. Diaz and D. Metzler. **Improving the estimation of relevance models using large external corpora**. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–161. ACM, 2006.

[64] A. Doan, R. Ramakrishnan, and A.Y. Halevy. **Crowdsourcing systems on the world-wide web**. *Communications of the ACM*, **54**(4):86–96, 2011.

[65] A. Dogac, V. Bicer, and A. Okcan. **Collaborative business process support in ihe xds through ebxml business processes**. *Proceedings of 22nd International Conference on Data Engineering (ICDE'06).*, 2006.

[66] J. dos Reis, R. Bonacin, and M. Baranauskas. **Beyond the social search: personalizing the semantic search in social networks**. *Online Communities and Social Computing*, pages 345–354, 2011.

[67] F. Doshi-Velez and Z. Ghahramani. **Accelerated sampling for the Indian buffet process**. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 273–280. ACM, 2009.

[68] Z. Dou, R. Song, and J.R. Wen. **A large-scale evaluation and analysis of personalized search strategies**. In *Proceedings of the 16th international conference on World Wide Web*, pages 581–590. ACM, 2007.

[69] M. Eisenberg and L. Schamber. **Relevance: the search for a definition**. In *Proc. 51st Annual Meeting of the American Society for Information Science*, 1988.

[70] A.P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley, 2007.

[71] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. **Learning with Kernels in Description Logics**. In *ILP*, pages 210–225, 2008.

[72] M. Fernandez, I. Cantador, V. López, D. Vallet, P. Castells, and E. Motta. **Semantically-enhanced Information Retrieval: An Ontology-based Approach**. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2010.

[73] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. **Evaluating implicit measures to improve web search**. *ACM Transactions on Information Systems (TOIS)*, **23**(2):147–168, 2005.

[74] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. **An efficient boosting algorithm for combining preferences**. *The Journal of Machine Learning Research*, **4**:933–969, 2003.

[75] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. **Learning probabilistic relational models**. In *International Joint Conference on Artificial Intelligence*, **16**, pages 1300–1309, 1999.

[76] N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas. **INEX: INitiative for the Evaluation of XML retrieval**. In *Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval*, **2006**, pages 1–9, 2002.

[77] X. Geng, T.Y. Liu, T. Qin, A. Arnold, H. Li, and H.Y. Shum. **Query dependent ranking using k-nearest neighbor**. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–122. ACM, 2008.

[78] Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. **Feature selection for ranking**. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 407–414, New York, NY, USA, 2007. ACM.

[79] L. Getoor, N. Friedman, D. Koller, and B. Taskar. **Learning probabilistic models of relational structure**. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 170–177, 2001.

[80] L. Getoor and B. Taskar. *Introduction to statistical relational learning*. MIT Press, 2007.

[81] Z. Ghahramani and M.J. Beal. **Propagation algorithms for variational Bayesian learning**. *Advances in neural information processing systems*, pages 507–513, 2001.

[82] Z. Ghahramani, T.L. Griffiths, and P. Sollich. **Bayesian nonparametric latent feature models**. *Bayesian Statistics*, **8**:1–25, 2007.

[83] Mehmet Gönen and Ethem Alpaydin. **Localized multiple kernel learning**. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 352–359, New York, NY, USA, 2008. ACM.

[84] J. Gonzalo, F. Verdejo, I. Chugur, and J. Cigarran. **Indexing with WordNet synsets can improve text retrieval**. *Arxiv preprint cmp-lg/9808002*, 1998.

[85] T.L. Griffiths and Z. Ghahramani. **The Indian Buffet Process: An Introduction and Review**. *Journal of Machine Learning Research*, **12**:1185–1224, 2011.

[86] KP Gummadi, A. Mislove, and P. Druschel. **Exploiting social networks for internet search**. In *Proc. 5th Workshop on Hot Topics in Networks*, pages 79–84.

[87] U. Hanani, B. Shapira, and P. Shoval. **Information filtering: Overview of issues, research and systems**. *User Modeling and User-Adapted Interaction*, **11**(3):203–259, 2001.

[88] D. Haussler. **Convolution kernels on discrete structures (Technical Report UCSC-CRL-99-10)**. *University of California, Santa Cruz*, 1999.

[89] T.H. Haveliwala. **Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search**. *Knowledge and Data Engineering, IEEE Transactions on*, **15**(4):784–796, 2003.

[90] Hao He, Haixun Wang, Jun Yang, and Philip S. Yu. **BLINKS: ranked keyword searches on graphs**. In *SIGMOD Conference*, pages 305–316, 2007.

[91] Marti A. Hearst and Jan O. Pedersen. **Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results**. In *SIGIR*, pages 76–84, 1996.

[92] W. Hersh and R.T. Bhupatiraju. **TREC genomics track overview**. In *TREC 2003*, pages 14–23, 2003.

[93] P. Heymann, D. Ramage, and H. Garcia-Molina. **Social tag prediction**. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 531–538. ACM, 2008.

[94] B. Hjørland. **The foundation of the concept of relevance**. *Journal of the American Society for Information Science and Technology*, **61**(2):217–237, 2010.

[95] T. Hofmann. **Probabilistic latent semantic indexing**. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57. ACM, 1999.

[96] V. Hristidis and Y. Papakonstantinou. **DISCOVER: Keyword search in relational databases**. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 670–681. VLDB Endowment, 2002.

[97] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. **Efficient IR-Style Keyword Search over Relational Databases**. In *VLDB*, pages 850–861, 2003.

[98] X. Hu, R. Shonkwiler, and MC Spruill. **Random restarts in global optimization**. 2009.

[99] Guichun Hua, Min Zhang, Yiqun Liu, Shaoping Ma, and Liyun Ru. **Hierarchical feature selection for ranking**. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1113–1114, New York, NY, USA, 2010. ACM.

[100] Y. Huang, M. Nickel, V. Tresp, and H.P. Kriegel. **A Scalable Kernel Approach to Learning in Semantic Graphs with Applications to Linked Data**. *Proc. of the 1st Workshop on Mining the Future Internet*, 2010.

[101] I.F. Ilyas, G. Beskales, and M. A. Soliman. **A Survey of Top-k Query Processing Techniques in Relational Database Systems**. *ACM Computing Surveys*, **40**(4), 2008.

[102] N. Jardine and C.J. van Rijsbergen. **The use of hierarchic clustering in information retrieval**. *Information storage and retrieval*, **7**(5):217–240, 1971.

[103] K. Järvelin and J. Kekäläinen. **Cumulated gain-based evaluation of IR techniques**. *ACM Transactions on Information Systems (TOIS)*, **20**(4):422–446, 2002.

[104] G. Jeh and J. Widom. **Scaling personalized web search**. In *Proceedings of the 12th International Conference on World Wide Web*, pages 271–279. ACM, 2003.

[105] D. Jensen and J. Neville. **Linkage and autocorrelation cause feature selection bias in relational learning**. In *Nineteenth International Conference on Machine Learning (ICML)*, pages 259–266, 2002.

[106] J. Jeon, V. Lavrenko, and R. Manmatha. **Automatic image annotation and retrieval using cross-media relevance models**. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 119–126. ACM, 2003.

[107] T. Joachims. **Optimizing search engines using clickthrough data**. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM, 2002.

[108] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. **Accurately interpreting clickthrough data as implicit feedback**. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–161. ACM, 2005.

[109] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. **Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search**. *ACM Transactions on Information Systems (TOIS)*, **25**(2):7, 2007.

[110] T. Joachims and F. Radlinski. **Search engines that learn from implicit feedback**. *Computer*, **40**(8):34–40, 2007.

[111] K.S. Jones, S. Walker, and S.E. Robertson. **A probabilistic model of information retrieval: development and comparative experiments**. *Information Processing and Management: an International Journal*, **36**(6):779–808, 2000.

[112] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. **An introduction to variational methods for graphical models**. *Machine learning*, **37**(2):183–233, 1999.

[113] Y. Kabak, M. Olduz, G.B. Laleci, T. Namli, V. Bicer, N. Radic, G. Milis, and A. Dogac. **A Semantic Web Service Based Middleware for the Tourism Industry**. *Semantic Enterprise Application Integration for Business Processes: Service-Oriented Frameworks*, page 189, 2009.

[114] Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, S. Sudarshan, Rushi Desai, and Hrishikesh Karambelkar. **Bidirectional Expansion For Keyword Search on Graph Databases**. In *VLDB*, pages 505–516, 2005.

[115] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. **The YAGO-NAGA approach to knowledge discovery**. *SIGMOD Record*, **37**(4):41–47, 2008.

[116] G. Kasneci, F.M. Suchanek, G. Ifrim, S. Elbassuoni, M. Ramanath, and G. Weikum. **NAGA: harvesting, searching and ranking knowledge**. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1285–1288. ACM, 2008.

[117] D. Kelly and J. Teevan. **Implicit feedback for inferring user preference: a bibliography**. In *ACM SIGIR Forum*, **37**, pages 18–28. ACM, 2003.

[118] John Kemp, Tapio Kaukonen, and Timo Skytta. *Mobile Web Services: Architecture and Implementation*. Wiley & Sons, 2006.

[119] Christoph Kiefer, Abraham Bernstein, and André Locher. **Adding Data Mining Support to SPARQL Via Statistical Relational Learning Methods**. In *ESWC*, pages 478–492, 2008.

[120] S.B. Kim, H.C. Seo, and H.C. Rim. **Information retrieval using word senses: root sense tagging approach**. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 258–265. ACM, 2004.

[121] J.M. Kleinberg. **Authoritative sources in a hyperlinked environment**. *Journal of the ACM (JACM)*, **46**(5):604–632, 1999.

[122] D. Koller and N. Friedman. *Probabilistic graphical models*. MIT press, 2009.

[123] S. Kramer, N. Lavrac, and P. Flach. **Propositionalization Approaches to Relational Data Mining**. *Relational Data Mining*, page 262, 2001.

[124] O. Kurland and L. Lee. **Corpus structure, language models, and ad hoc information retrieval**. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 194–201. ACM, 2004.

[125] O. Kurland and L. Lee. **Respect my authority!: HITS without hyperlinks, utilizing cluster-based language models**. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90. ACM, 2006.

[126] J. Lafferty, A. McCallum, and F. Pereira. **Conditional random fields: Probabilistic models for segmenting and labeling sequence data**. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 282–289, 2001.

[127] J. Lafferty and C. Zhai. **Document language models, query models, and risk minimization for information retrieval**. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 111–119. ACM, 2001.

[128] Y. Lan, T.Y. Liu, Z. Ma, and H. Li. **Generalization analysis of listwise learning-to-rank algorithms**. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 577–584. ACM, 2009.

[129] Y. Lan, T.Y. Liu, T. Qin, Z. Ma, and H. Li. **Query-level stability and generalization in learning to rank**. In *Proceedings of the 25th international conference on Machine learning*, pages 512–519. ACM, 2008.

[130] F.W. Lancaster and E.G. Fayen. **Information retrieval on-line**. *Melville Pub. Co. Los Angeles*, 1973.

[131] V. Lavrenko. *A generative theory of relevance*. Springer, 2008.

[132] V. Lavrenko, M. Choquette, and W.B. Croft. **Cross-lingual relevance models**. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 175–182. ACM, 2002.

[133] V. Lavrenko and W.B. Croft. **Relevance based language models**. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127. ACM, 2001.

[134] V. Lavrenko, SL Feng, and R. Manmatha. **Statistical models for automatic video annotation and retrieval**. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, **3**, pages iii–1044. IEEE, 2004.

[135] V. Lavrenko, T.M. Rath, and R. Manmatha. **Holistic word recognition for handwritten historical documents**. In *Document Image Analysis for Libraries, 2004. Proceedings. First International Workshop on*, pages 278–287. IEEE, 2004.

[136] Victor Lavrenko, Xing Yi, and James Allan. **Information Retrieval On Empty Fields**. In *HLT-NAACL*, pages 89–96, 2007.

[137] K.S. Lee, W.B. Croft, and J. Allan. **A cluster-based resampling method for pseudo-relevance feedback**. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 235–242. ACM, 2008.

[138] G. Li, S. Ji, C. Li, and J. Feng. **Efficient type-ahead search on relational data: a tastier approach**. In *Proceedings of the 35th SIGMOD International Conference on Management of Data*, pages 695–706. ACM, 2009.

[139] Guoliang Li, Beng Chin Ooi, Jianhua Feng, Jianyong Wang, and Lizhu Zhou. **EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data**. In *SIGMOD Conference*, pages 903–914, 2008.

[140] P. Li, C. Burges, Q. Wu, JC Platt, D. Koller, Y. Singer, and S. Roweis. **Mcrank: Learning to rank using multiple classification and gradient boosting**. *Advances in neural information processing systems*, **20**:897–904, 2007.

[141] F. Liu, C. Yu, and W. Meng. **Personalized web search by mapping user queries to categories**. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 558–565. ACM, 2002.

[142] F. Liu, C. Yu, and W. Meng. **Personalized web search for improving retrieval effectiveness**. *IEEE Transactions on knowledge and data engineering*, **16**(1):28–40, 2004.

[143] Fang Liu, Clement T. Yu, Weiyi Meng, and Abdur Chowdhury. **Effective keyword search in relational databases**. In *SIGMOD Conference*, pages 563–574, 2006.

[144] J. Liu, P. Dolan, and E.R. Pedersen. **Personalized news recommendation based on click behavior**. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.

[145] S. Liu, F. Liu, C. Yu, and W. Meng. **An effective approach to document retrieval via utilizing WordNet and recognizing phrases**. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 266–272. ACM, 2004.

[146] X. Liu and W.B. Croft. **Cluster-based retrieval using language models**. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 193. ACM, 2004.

[147] Z. Liu, P. Sun, Y. Huang, Y. Cai, and Y. Chen. **Challenges, Techniques and Directions in Building XSeek: an XML Search Engine**. 2009.

[148] S. Lucchetti. *The Principle of Relevance: The Essential Strategy to Navigate Through the Information Age*. Publishing Technology Center, Hong Kong University of Science and Technology, 2011.

[149] S. Luke, L. Spector, and D. Rager. **Ontology-based knowledge discovery on the world-wide web**. In *Working Notes of the Workshop on Internet-Based Information Systems at the 13th National Conference on Artificial Intelligence (AAAI96)*, pages 96–102. AMI, 1996.

[150] Yi Luo, Xuemin Lin, Wei Wang, and Xiaofang Zhou. **Spark: top-k keyword query in relational databases**. In *SIGMOD Conference*, pages 115–126, 2007.

[151] Y. Lv and C.X. Zhai. **Positional relevance model for pseudo-relevance feedback**. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 579–586. ACM, 2010.

[152] D.J.C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge Univ Pr, 2003.

[153] J. Madhavan, L. Afanasiev, L. Antova, and A. Halevy. **Harnessing the Deep Web: present and future**. *Arxiv preprint arXiv:0909.1785*, 2009.

[154] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. **Semantic portal-the seal approach**. *Spinning the Semantic Web. Bringing the World Wide Web to Its Full Potential*, pages 317–359.

[155] C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*, **59**. MIT Press, 1999.

[156] M. Marchiori. **The quest for correct information on the web: Hyper search engines**. *Computer Networks and ISDN Systems*, **29**(8-13):1225–1235, 1997.

[157] D. Martinenghi and M. Tagliasacchi. **Proximity rank join**. *Proceedings of the VLDB Endowment*, **3**(1-2):352–363, 2010.

[158] A.K. McCallum. **MALLET: A machine learning for language toolkit. 2002**. *URL http://www. cs. umass. edu/mccallum/mallet*.

[159] P. McCullagh and J.A. Nelder. *Generalized linear models*. Chapman & Hall/CRC, 1989.

[160] Q. Mei, D. Cai, D. Zhang, and C.X. Zhai. **Topic modeling with network regularization**. In *Proceeding of the 17th international conference on World Wide Web*, pages 101–110. ACM, 2008.

[161] Qiaozhu Mei, Duo Zhang, and ChengXiang Zhai. **A general optimization framework for smoothing language models on graph structures**. In *SIGIR*, pages 611–618, 2008.

[162] E. Meij, D. Trieschnigg, M. de Rijke, and W. Kraaij. **Conceptual language models for domain-specific retrieval**. *Information Processing & Management*, **46**(4):448–469, 2010.

[163] D. Milne and I.H. Witten. **Learning to link with wikipedia**. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM, 2008.

[164] S. Mizzaro. **Relevance: The whole history**. *Journal of the American Society for Information Science*, **48**(9):810–832, 1997.

[165] S. Mizzaro. **How many relevances in information retrieval?** *Interacting with computers*, **10**(3):303–320, 1998.

[166] M. Montaner, B. López, and J.L. De La Rosa. **A taxonomy of recommender agents on the internet**. *Artificial intelligence review*, **19**(4):285–330, 2003.

[167] S. Muggleton and L. De Raedt. **Inductive logic programming: Theory and methods**. *The Journal of Logic Programming*, **19**:629–679, 1994.

[168] Stephen Muggleton. **Inductive logic programming**. *New Gen. Comput.*, **8**(4):295–318, 1991.

[169] M. Najork. **Comparing the Effectiveness of HITS and SALSA**. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM)*, 2007.

[170] R. Nallapati. **Discriminative models for information retrieval**. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 64–71. ACM, 2004.

[171] R.M. Nallapati, A. Ahmed, E.P. Xing, and W.W. Cohen. **Joint latent topic models for text and citations**. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 542–550. ACM, 2008.

[172] Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji-Rong Wen, and Wei-Ying Ma. **Web object retrieval**. In *WWW*, pages 81–90, 2007.

[173] M.G. Noll and C. Meinel. **Web search personalization via social bookmarking and tagging**. In *Proceedings of the 6th International Semantic Web Conference (ISWC) and 2nd Asian Semantic Web Conference (ASWC)*, pages 367–380. Springer-Verlag, 2007.

[174] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johnson. **Terrier information retrieval platform**. *Advances in Information Retrieval*, pages 517–519, 2005.

[175] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. **The PageRank Citation Ranking: Bringing Order to the Web**, 1998.

[176] Feng Pan, Tim Converse, David Ahn, Franco Salvetti, and Gianluca Donato. **Feature selection for ranking using boosted trees**. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 2025–2028, New York, NY, USA, 2009. ACM.

[177] V. Petras, S. Baerisch, and M. Stempfhuber. **The domain-specific track at CLEF 2007**. *Advances in multilingual and multimodal information retrieval*, pages 160–173, 2008.

[178] B. Piwowarski, G. Dupret, and R. Jones. **Mining user web search activity with layered bayesian networks or how to capture a click in its context**. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 162–171. ACM, 2009.

[179] Jay M. Ponte and W. Bruce Croft. **A Language Modeling Approach to Information Retrieval**. In *SIGIR*, pages 275–281, 1998.

[180] J.M. Ponte and W.B. Croft. **A language modeling approach to information retrieval**. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281. ACM, 1998.

[181] H. Poon and P. Domingos. **Joint inference in information extraction**. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, **22**, page 913. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[182] A. Popescul and L.H. Ungar. **Feature Generation and Selection in Multi-Relational Statistical Learning**. *Introduction to statistical relational learning*, page 453, 2007.

[183] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. **Ad-hoc object retrieval in the web of data**. In *WWW*, pages 771–780, 2010.

[184] A. Pretschner and S. Gauch. **Ontology based personalized search**. In *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*, pages 391–398. IEEE, 1999.

[185] L. Qin, J.X. Yu, L. Chang, and Y. Tao. **Querying communities in relational databases**. In *Proceedings of the 25th International Conference on Data Engineering (ICDE)*, pages 724–735. IEEE, 2009.

[186] T. Qin, T.Y. Liu, and H. Li. **A general approximation framework for direct optimization of information retrieval measures**. *Information retrieval*, **13**(4):375–397, 2010.

[187] F. Radlinski and T. Joachims. **Query chains: learning to rank from implicit feedback**. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248. ACM, 2005.

[188] F. Raiber and O. Kurland. **On identifying representative relevant documents**. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 99–108. ACM, 2010.

[189] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. **SimpleMKL**. *Journal of Machine Learning Research*, **9**:2491–2521, 2008.

[190] D. Ramage, D. Hall, R. Nallapati, and C.D. Manning. **Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora**. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.

[191] D. Ravichandran. *Terascale knowledge acquisition*. PhD thesis, UNIVERSITY OF SOUTHERN CALIFORNIA, 2005.

[192] Achim Rettinger, Matthias Nickles, and Volker Tresp. **Statistical Relational Learning with Formal Ontologies**. In *ECML/PKDD (2)*, pages 286–301, 2009.

[193] M. Richardson and P. Domingos. **Markov logic networks**. *Machine Learning*, **62**(1):107–136, 2006.

[194] M. Richardson, A. Prakash, and E. Brill. **Beyond PageRank: machine learning for static ranking**. In *Proceedings of the 15th international conference on World Wide Web*, pages 707–715. ACM, 2006.

[195] S.E. Robertson. **The probability ranking principle in IR**. *Journal of documentation*, **33**(4):294–304, 1977.

[196] S.E. Robertson and K.S. Jones. **Relevance weighting of search terms**. *Journal of the American society for Information Science*, **27**(3):129–146, 1976.

[197] S.E. Robertson and S. Walker. **Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval**. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241. Springer-Verlag New York, Inc., 1994.

[198] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi de Aragão. **A hybrid approach for searching in the semantic web**. In *WWW*, pages 374–383, 2004.

[199] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. **The author-topic model for authors and documents**. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 487–494. AUAI Press, 2004.

[200] G. Salton. *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968.

[201] G. Salton, A. Wong, and C.S. Yang. **A vector space model for automatic indexing**. *Communications of the ACM*, **18**(11):613–620, 1975.

[202] T. Saracevic. **Relevance: A review of and a framework for the thinking on the notion in information science**. *Journal of the American Society for Information Science*, **26**(6):321–343, 1975.

[203] T. Saracevic. **Relevance reconsidered**. In *Proceedings of the Second Conference on Conceptions of Library and Information Science (CoLIS 2)*, pages 201–218, 1996.

[204] S. Sarawagi and W.W. Cohen. **Semi-markov conditional random fields for information extraction**. *Advances in Neural Information Processing Systems*, **17**:1185–1192, 2005.

[205] K. Schnaitter and N. Polyzotis. **Evaluating rank joins with optimal cost**. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 43–52. ACM, 2008.

[206] B. Scholkopf and A.J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. the MIT Press, 2002.

[207] H. Schutze and J.O. Pedersen. **Information retrieval based on word senses**. In *In Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, 1995.

[208] W. Shen, A.H. Doan, J.F. Naughton, and R. Ramakrishnan. **Declarative information extraction using datalog with embedded extraction predicates**. In *Proceedings of the 33rd international Conference on Very Large Data Bases (VLDB)*, pages 1033–1044. VLDB Endowment, 2007.

[209] X. Shen, B. Tan, and C.X. Zhai. **Context-sensitive information retrieval using implicit feedback**. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 50. ACM, 2005.

[210] A. Sieg, B. Mobasher, and R. Burke. **Web search personalization with ontological user profiles**. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 525–534. ACM New York, NY, USA, 2007.

[211] Amit Singhal, Chris Buckley, and Mandar Mitra. **Pivoted Document Length Normalization**. In *SIGIR*, pages 21–29, 1996.

[212] S. Soderland. **Learning information extraction rules for semi-structured and free text**. *Machine learning*, **34**(1):233–272, 1999.

[213] R. Srikant, S. Basu, N. Wang, and D. Pregibon. **User browsing models: Relevance versus examination**. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 223–232. ACM, 2010.

[214] Padmini Srinivasan. **Query Expansion and MEDLINE**. *Inf. Process. Manage.*, **32**(4):431–443, 1996.

[215] M. Steyvers and T. Griffiths. **Probabilistic topic models**. *Handbook of latent semantic analysis*, **427**(7):424–440, 2007.

[216] N. Stojanovic, R. Studer, and L. Stojanovic. **An approach for the ranking of query results in the semantic web**. *The SemanticWeb-ISWC 2003*, pages 500–516, 2003.

[217] K. Sugiyama, K. Hatano, and M. Yoshikawa. **Adaptive web search based on user profile constructed without any effort from users**. In *Proceedings of the 13th international conference on World Wide Web*, pages 675–684. ACM New York, NY, USA, 2004.

[218] T. Tao, X. Wang, Q. Mei, and C.X. Zhai. **Language model information retrieval with document expansion**. In *Proceedings of the Human Language Technology Conference*, pages 407–414. Association for Computational Linguistics, 2006.

[219] M. Taylor, J. Guiver, S. Robertson, and T. Minka. **SoftRank: optimizing non-smooth rank metrics**. In *Proceedings of the International Conference on Web Search and Web Data Mining*, pages 77–86. ACM, 2008.

[220] J. Teevan, S.T. Dumais, and E. Horvitz. **Personalizing search via automated analysis of interests and activities**. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 456. ACM, 2005.

[221] J. Teevan, S.T. Dumais, and E. Horvitz. **Potential for personalization**. *ACM Transactions on Computer-Human Interaction (TOCHI)*, **17**(1):1–31, 2010.

[222] E. Terra and R. Warren. **Poison pills: harmful relevant documents in feedback**. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 319–320. ACM, 2005.

[223] Anastasios Tombros, Robert Villa, and C. J. van Rijsbergen. **The effectiveness of query-specific hierarchic clustering in information retrieval**. *Inf. Process. Manage.*, **38**(4):559–582, 2002.

[224] T. Tran, P. Cimiano, S. Rudolph, and R. Studer. **Ontology-based interpretation of keywords for semantic search**. *The Semantic Web*, pages 523–536, 2007.

[225] Thanh Tran, Haofen Wang, Sebastian Rudolph, and Philipp Cimiano. **Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data**. In *ICDE*, pages 405–416, 2009.

[226] Dolf Trieschnigg, Wessel Kraaij, and Martijn J. Schuemie. **Concept Based Document Retrieval for Genomics Literature**. In *TREC*, 2006.

[227] T. Upstill, N. Craswell, and D. Hawking. **Query-independent evidence in home page finding**. *ACM Transactions on Information Systems (TOIS)*, **21**(3):286–313, 2003.

[228] D. Vallet, M. Fernández, and P. Castells. **An ontology-based information retrieval model**. *The Semantic Web: Research and Applications*, pages 103–110, 2005.

[229] CJ Van Rijsbergen. *Information retrieval*. London, Butterworths, 1979.

[230] M. Varma and B.R. Babu. **More generality in efficient multiple kernel learning**. In *Proceedings of ICML*, pages 1065–1072. ACM, 2009.

[231] E.M. Voorhees. **Implementing agglomerative hierarchic clustering algorithms for use in document retrieval* 1**. *Information Processing & Management*, **22**(6):465–476, 1986.

[232] E.M. Voorhees. **Query expansion using lexical-semantic relations**. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69. Springer-Verlag New York, Inc., 1994.

[233] E.M. VOORHEES. **Overview of the TREC 2001 question answering track**. *NIST special publication*, pages 42–51, 2002.

[234] M.J. Wainwright and M.I. Jordan. **Graphical models, exponential families, and variational inference**. *Foundations and Trends in Machine Learning*, **1**(1-2):1–305, 2008.

[235] R.H. Warren and T. Liu. **A review of relevance feedback experiments at the 2003 reliable information access (RIA) workshop.** In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 571. ACM, 2004.

[236] Martin Wechsler and Peter Schäuble. **The Probability Ranking Principle Revisited**. *Inf. Retr.*, **3**(3):217–227, 2000.

[237] W. Weerkamp, R. Berendsen, B. Kovachev, E. Meij, K. Balog, and M. de Rijke. **People searching for people: Analysis of a people search engine log**. In *Proceedings of the 34th Annual International ACM SIGIR Conference (SIGIR 2011), Beijing*, 2011.

[238] X. Wei and W.B. Croft. **LDA-based document models for ad-hoc retrieval**. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185. ACM, 2006.

[239] R. Wetzker, C. Zimmermann, C. Bauckhage, and S. Albayrak. **I tag, you tag: translating tags for advanced user models**. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 71–80. ACM, 2010.

[240] P. Wilson. **Situational relevance**. *Information storage and retrieval*, **9**(8):457–471, 1973.

[241] Stefan Wrobel and Saso Dzeroski. **The ILP description learning problem: Towards a general model-level definition of data mining in ILP**, 1995.

[242] H. Wu, G. Kazai, and M. Taylor. **Book search experiments: Investigating IR methods for the indexing and retrieval of books**. *Advances in Information Retrieval*, pages 234–245, 2008.

[243] X. Wu, L. Zhang, and Y. Yu. **Exploring social annotations for the semantic web**. In *Proceedings of the 15th international conference on World Wide Web*, pages 417–426. ACM, 2006.

[244] J. Xu and H. Li. **Adarank: a boosting algorithm for information retrieval**. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 391–398. ACM, 2007.

[245] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. **Exploring folksonomy for personalized search**. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 155–162. ACM, 2008.

[246] Y. Xu, G.J.F. Jones, and B. Wang. **Query dependent pseudo-relevance feedback based on wikipedia**. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 59–66. ACM, 2009.

[247] Y.C. Xu and Z. Chen. **Relevance judgment: What do information users consider beyond topicality?** *Journal of the American Society for Information Science and Technology*, **57**(7):961–973, 2006.

[248] Z. Xu, V. Tresp, K. Yu, and H.P. Kriegel. **Infinite hidden relational models**. In *Proceedings of the 22nd International Conference on Uncertainity in Artificial Intelligence (UAI 2006)*, 2006.

[249] Tie yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. **Letor: Benchmark dataset for research on learning to rank for information retrieval**. In *In Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

[250] J.Y. Yeh, J.Y. Lin, H.R. Ke, and W.P. Yang. **Learning to rank for information retrieval using genetic programming**. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007)*, 2007.

[251] J.X. Yu, L. Qin, and L. Chang. **Keyword Search in Databases**. *Synthesis Lectures on Data Management*, **1**(1):1–155, 2009.

[252] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. **A support vector method for optimizing average precision**. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 271–278. ACM, 2007.

[253] Yisong Yue. **New Learning Frameworks For Information Retrieval**. pages 20–37, 2011.

[254] C. Zhai and J. Lafferty. **Model-based feedback in the language modeling approach to information retrieval**. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 403–410. ACM, 2001.

[255] C. Zhai and J. Lafferty. **A study of smoothing methods for language models applied to information retrieval**. *ACM Transactions on Information Systems (TOIS)*, **22**(2):179–214, 2004.

[256] C.X. Zhai and J. Lafferty. **A risk minimization framework for information retrieval**. *Information Processing & Management*, **42**(1):31–55, 2006.

[257] Le Zhao and Jamie Callan. **A generative retrieval model for structured documents**. In *CIKM*, pages 1163–1172, 2008.

[258] D. Zhou, J. Bian, S. Zheng, H. Zha, and C.L. Giles. **Exploring social annotations for information retrieval**. In *Proceeding of the 17th international conference on World Wide Web*, pages 715–724. ACM, 2008.

[259] M. Zhu. **Recall, precision and average precision**. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2004.