Kernel Methods for Knowledge Structures

Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften

(Dr. rer. pol.)

von der Fakultät für
Wirtschaftswissenschaften
der Universität Karlsruhe (TH)

genehmigte

DISSERTATION

von
Dipl.-Inform.-Wirt. Stephan Bloehdorn

Tag der mündlichen Prüfung: 10. Juli 2008

Referent: Prof. Dr. Rudi Studer
Koreferent: Prof. Dr. Dr. Lars Schmidt-Thieme

2008 Karlsruhe

Abstract

Kernel methods constitute a new and popular field of research in the area of machine learning. Kernel-based machine learning algorithms abandon the explicit representation of data items in the vector space in which the sought-after patterns are to be detected. Instead, they implicitly mimic the geometry of the feature space by means of the kernel function, a similarity function which maintains a geometric interpretation as the inner product of two vectors. Knowledge structures and ontologies allow to formally model domain knowledge which can constitute valuable complementary information for pattern discovery. For kernel-based machine learning algorithms, a good way to make such prior knowledge about the problem domain available to a machine learning technique is to incorporate it into the kernel function. This thesis studies the design of such kernel functions.

First, this thesis provides a theoretical analysis of popular similarity functions for entities in taxonomic knowledge structures in terms of their suitability as kernel functions. It shows that, in a general setting, many taxonomic similarity functions can not be guaranteed to yield valid kernel functions and discusses the alternatives.

Secondly, the thesis addresses the design of expressive kernel functions for text mining applications. A first group of kernel functions, Semantic Smoothing Kernels (SSKs) retain the Vector Space Models (VSMs) representation of textual data as vectors of term weights but employ linguistic background knowledge resources to bias the original inner product in such a way that cross-term similarities adequately contribute to the kernel result. A second group of kernel functions, Semantic Syntactic Tree Kernels (SSTKs) replace the VSM representation by a representation based on parse trees. In contrast to other kernel functions based on this representation, they are again capable of a more adequate handling of semantic similarities between different terms. Both types of kernel functions are then extensively evaluated in practical experiments.

Thirdly, the thesis provides a framework for kernel functions on instance data in ontologies. These data instances are formally described by means of an ontological structure. The framework allows to flexibly model kernel functions that probe instances for selected ontological properties. Again, the thesis also shows the application of the proposed kernel functions in practical experiments.

Acknowledgements

Foremost, I would like to thank Prof. Dr. Rudi Studer, my advisor, for supporting me and my research during the last years. I have significantly benefited from his long experience and from his way of leading his team which has resulted in an outstanding research environment. I am also extremely grateful for his commitment to timely review this thesis despite his busy schedule.

I would also like to thank Prof. Dr. Dr. Lars Schmidt-Thieme from University of Hildesheim who, without hesitation, accepted the request to serve on the examination committee as second reviewer and provided detailed and valuable technical feedback. Many thanks to Prof. Dr. Detlef Seese and Prof. Dr. Oliver Stein who served on the examination committee as examiner and chairman, respectively.

I am grateful to my friends and colleagues at LS3/WIM for the excellent and lively working atmosphere within the team. Special thanks to York Sure for mentoring my project and thesis work. Many thanks also to Philipp Cimiano and Peter Haase for reviewing parts of this thesis as well as to Max Völkel and Denny Vrandečić for the fun discussions on dissertation progress during the Sushi lunchtime meetings designed for this purpose.

Similarly, I am grateful to all external collaborators and project partners. In this context, special thanks to Prof. Dr. Alessandro Moschitti from University of Trento. Various parts of this thesis have grown out of collaboration with him and I have often benefited from his experience and opinion.

Last but not least, I would like to thank my family and friends. My parents Brigitte and Jörg as well as my brother Johannes have at all times supported my way in every respect. Mostly, however, I am grateful to Melanie for her constant encouragement and, well, for liking me.

Karlsruhe, August 2008
Stephan Bloehdorn

Contents

Notational Conventions

Where possible, this thesis will adhere to the following notational conventions.

$\mathcal{A}, \mathcal{B}, \ldots$	Simple sets.
$x \in \mathcal{X}$	Input and input space; default is $\mathcal{X} = \mathbb{R}^d$
x_i	The i-th data item of a sequence; if clear from context, also i-th component of vector x
$y \in \mathcal{Y}$	Label and label set; default is $\mathcal{Y} = \{+1, -1\}$
$\mathcal{S} = (x_1, y_1), \ldots, (x_n, y_n) \subseteq \mathcal{X} \times \mathcal{Y}$	Example (training or test) set for classification
$\mathcal{S} = (x_1), \ldots, (x_n) \subseteq \mathcal{X}$	Example (training or test) set for clustering
$\mathcal{F}$	The space of (hypothesis) functions
$\mathcal{H}$	Hilbert space
$\mathbb{N}$, $\mathbb{R}$	The natural and real numbers
$\lVert \cdot \rVert_p$ $(\lVert \cdot \rVert)$	L_p-norm (default is L_2-norm)
$\mathbf{P}, \mathbf{Q} \ldots$	Simple matrices
$\mathbf{K}$	Kernel (Gram) matrix
$\mathbf{I}$	Identity matrix
$\mathcal{K}$	Knowledge Structure
$\mathcal{O}$	Ontology
$\mathcal{E}$	Set of entity names in a knowledge structure
$\mathcal{P}$	Set of property names in a knowledge structure
$\mathcal{D}$	Set of data values in a knowledge structure
$\mathcal{S} \subseteq \mathcal{E} \times \mathcal{P} \times (\mathcal{E} \cup \mathcal{D})$	Set of statements in a knowledge structure
$\mathcal{C} = \{\mathsf{class}_1, \mathsf{class}_2, \ldots\}$	Class entities in ontology
$\mathcal{P} = \{\mathsf{prop}_1, \mathsf{prop}_2, \ldots\}$	Property entities in ontology
$\mathcal{I} = \{\mathit{indiv}_1, \mathit{indiv}_2, \ldots\}$	Individual entities in ontology
$\mathsf{class}_1(\mathit{indiv}_1)$	Class instantiation
$\mathsf{prop}_1(\mathit{indiv}_1, \mathit{indiv}_2)$, $\mathsf{prop}_2(\mathit{indiv}_1, \mathtt{value})$	(Object/Datatype) Property instantiation

Chapter 1

Introduction

The ability to learn, i.e. to infer general rules after repeatedly observing recurring phenomena, lies at the core of intelligent behaviour. In analogy with these roots, the field of machine learning studies techniques that deal with the automatic detection of patterns in data by modern computing systems. This thesis is occupied with a specific class of machine learning techniques called kernel methods. Extending classical machine learning techniques, kernel methods distinguish themselves by an elegant way to interface with the data by means of the so-called kernel function. A good way to make prior knowledge about the problem domain available to a machine learning technique is to incorporate this knowledge into the kernel function. Similarly, kernel functions can be designed to work directly on data items that do not have a natural vectorial representation.

As its overall topic, this thesis investigates the design of kernel functions that take advantage of knowledge expressed in formal knowledge structures and associated metadata. In this introductory chapter, we shortly introduce the context of this thesis and give an overview of its contributions.

1.1 Background

Machine learning techniques mimic the process of learning by means of statistical models. Presented with a limited amount of data, machine learning techniques aim at learning about hidden patterns in the data by estimating the parameters of the statistical model based on the observed data characteristics. Depending on the application context, the model may be designed to either give a compact characterisation of the dataset or to predict selected characteristics based on limited information in the future. If the estimation is sufficiently accurate the resulting model can be applied in its respective application context. During the last decade, the use of machine learning techniques has contributed significantly to solving a large set of practical applications in business and scientific settings.

The success of a machine learning approach in solving a given task depends on various factors. Some of these relate to the employed estimation procedures. However, a more important set of factors relates to the capability of accurately representing the relevant data characteristics that can help to describe the sought-after patterns. Kernel methods constitute a fairly recent direction of machine learning research related to this latter aspect. The major paradigm behind kernel methods is the decoupling of the employed learning algorithms from the representations of the data instances under investigation. Different popular learning algorithms can be "kernelized" such as, for example, Support Vector Machines (SVMs) (Vapnik et al., 1997) for supervised learning tasks or Kernel-kMeans and Kernel-PCA (Schölkopf et al., 1996) for clustering and dimensionality reduction, i.e. unsupervised learning tasks. In their original formulation, these algorithms operate on vectors of real numbers. The hypotheses generated by these algorithms are then tied to a geometric interpretation within the corresponding vector space. An example of such a geometric interpretation is the notion of a separating hyperplane in the case of linear classifiers.

In contrast, the "kernelized" variants of these algorithms are designed in such a way that the input vectors need not be accessible directly by the algorithms of interest. Instead, it is sufficient that they can access the evaluations of the inner products of any two vectors in the feature space. In this situation, as we will see, any function which is positive semi-definite can replace the original inner product, thereby defining a different feature space only implicitly. Conceptually, the kernel function can be regarded as a function that encodes a particular notion of similarity of data items of the input domain while complying with the interpretation as an inner product, regardless of the reference space. The kernel function as such thus becomes an interesting subject of research.

At the same time, ontologies and knowledge structures constitute powerful paradigms for modelling a domain of interest in terms of declarative knowledge. While these formalisms have been studied for some time, principled approaches for the exploitation of knowledge structures within machine learning settings are still a major subject of research. Structurally, the approaches that combine machine learning techniques and knowledge structures can be organized according to the type of the objects of interest to be analyzed by the learning techniques (Bloehdorn and Hotho, 2008).

"Ordinary" Data: In this setting, the objects of interest are arbitrary data items which have already been the subject of investigation in conventional machine learning settings. However, their content and conventional representation can be mapped to entities found in a knowledge structure.

Ontology Entities: In this setting, the objects of interest are part of the knowledge structure themselves. This setting covers all cases where ontological entities become the focus of the mining activities.

Knowledge Structures: Finally, this group of approaches covers all cases, where whole knowledge structures or fragments thereof, i.e. the respective sets of entities and statements are the objects of interest.

Along this classification, the kernel functions studied in this thesis are specific instantiations of the first two types of tasks. We will now introduce the respective research questions in closer detail.

1.2 Research Questions and Contributions

In this section, the contributions of this thesis are outlined. In summary, this work covers three sub-problems, namely (i) the analysis of popular similarity functions in taxonomic knowledge structures in terms of their suitability as kernel functions, (ii) the design of expressive and adequate kernel functions for textual data, that employ linguistic background knowledge resources and (iii) the design of kernel functions on data items that are formally described as instances of an ontological structure.

Kernel Functions for Entities in Taxonomic Structures

As we will see, it is not straightforward to see whether a function that assess the similarity of two entities in a taxonomic knowledge structure constitutes a valid kernel function. This property is required if the function should be used as a kernel on entities in the taxonomy. Furthermore, this is also required if the similarity function should be combined with other kernel functions (a matter of interest for the subsequent research question). This issue leads us to the first research question.

Research Question 1 (Validity of Taxonomic Similarity Measures)**.** *Which of the established similarity functions on entities in taxonomic structures constitute valid kernel functions?*

With respect to this question, this thesis conducts a systematic analysis of popular taxonomic similarity functions with respect to the question whether they have an interpretation as kernel functions or under which condition they do so. The problem also lies at the core of the kernel functions to be discussed in the later parts, thus constituting a basic result for these subsequent parts.

Kernel Functions for Semantic Smoothing in Text Mining

Traditionally, text mining systems rely on the Vector Space Model (VSM) that has been introduced early in information retrieval (IR) systems (Salton and McGill, 1983). According to the Vector Space Model, textual inputs are encoded as vectors whose dimensions correspond to the terms in the set of documents under consideration. While this straightforward feature representation has an appealing simplicity, a common caveat is the observation that the terms that constitute the feature space can not be regarded as mutually orthogonal dimensions but rather as dimensions with varying degrees of semantic similarity. In this view, inner products on the VSM appear only as rough approximations of the actual text similarities. On the other hand, lexical knowledge structures like WORDNET provide a rich source of knowledge about the semantic dependencies between different terms which could be exploited to address the shortcomings of the basic VSM.

Research Question 2 (Semantic Smoothing Kernels for the VSM Representation)**.** *How can we design kernel functions which complement the basic VSM representation with information about the semantic similarity of terms from lexical knowledge structures and what are the properties of these kernels?*

Conceptually, this corresponds to the first setting considered above, i.e. mining ordinary data with the help of knowledge structures. The mutual exploitation of the benefits of lexical knowledge structures and statistical text analysis is one of the most prominent problems in text mining and information retrieval (Bloehdorn et al., 2005). Consequently, the model we will study, the class of *Semantic Smoothing Kernels (SSKs)*, bears connections to a variety of existing work in these fields. The main contribution of this thesis w.r.t. SSKs is the systematic analysis of the behavior and the properties of these kernel functions.

While SSKs approach a long-standing problem of the VSM representation in a principled and intuitive way, the perception of natural language does not only draw from the semantic content of isolated terms but also from the syntactic structure they are used in. In linguistics, syntactic structure is commonly represented by parse trees of the textual input. In recent years, specific kernel functions have been designed to work directly on such parse tree representations. However, existing kernel functions again neglect the issue of variability of natural language. Finding principled techniques for exploiting the syntactic structure and the semantic variability of natural language texts at the same time within a unified framework constitutes another promising research line. This leads to the next research question.

Research Question 3 (Combining Semantic Smoothing and Syntactic Structure)**.** *How can we design kernel functions on textual documents that combine concepts of semantic smoothing based on lexical knowledge structures as well as knowledge about their syntactic structure and what are the properties of these kernels?*

With respect to this question, this thesis proposes a generalization of tree kernel functions which directly incorporates semantic background information, a family of kernels which we call *Semantic Syntactic Tree Kernels (SSTKs)*. These new kernel functions constitute the first principled framework for kernel functions that build upon linguistic structure and background knowledge about the semantic dependencies of terms at the same time. Again, we also systematically analyse the behavior and the properties of these kernel functions.

Both types of kernel functions, SSKs and SSTKs, require the choice of adequate smoothing parameters in such a way that they encode a useful notion of semantic similarity while ensuring the validity of the overall kernels. The proposed design choices are motivated by the results of the previous sub-problem. Both types of kernel functions, SSKs and SSTKs, are then applied in a set of practical experiments.

Kernel Functions for Mining Instance Data in Ontologies

With the emergence of the so-called Semantic Web, more and more datasets become available which are formally described according to a specific ontology in a formal ontology language. The increased availability of such data sources on the Semantic Web leads to an increased interest in exploiting these data sources within intelligent applications. While we have so far considered knowledge structures as a source of complementary knowledge in an established machine learning scenario, we now turn our attention to the question of learning from the knowledge structures themselves. Conceptually, this corresponds to the second setting considered above, i.e. mining entities within knowledge structures.

While the mapping of Semantic Web languages to formal logic allows for intelligent applications based on deductive reasoning, the use of inductive approaches is not straightforward. In particular, instances in formal knowledge structures do not lend themselves naturally to a vector-based representation. The aim of this sub-problem is thus to design and work with kernel functions that directly work on individuals described within formal ontologies. This leads to the next research question.

Research Question 4 (Kernels for Instance Date in Ontologies)**.** *How can we design kernel functions on instances described formally by means of an ontology?*

With respect to this question, this thesis introduces a framework for kernel design on ontological instances. Conceptually, such kernels exploit assertions about the argument individuals while they are at the same time based on the formal semantics of the underlying ontology language. Besides the application of machine learning techniques directly to Semantic Web data, this approach enables to naturally incorporate intensional background knowledge. Again, the application of the resulting kernels is shown in a set of practical experiments.

1.3 Outline

The current Chapter 1 has given a general introduction to this thesis, has discussed its research questions and presented the contributions form a bird's eye perspective.

The outline over the subsequent chapters is as follows:

Part I introduces the background for the work of this thesis.

Chapter 2 introduces basic concepts from the field of machine learning and, specifically, kernel methods, including the relevant mathematical background.

Chapter 3 introduce basic concepts from the field of knowledge representation. The exposition is based around a simple formal framework for knowledge structures. Even though the framework is kept generic, it is capable of characterizing a large set of views on the topic of knowledge representation.

Part II reports on the first research question.

Chapter 4 introduces and investigates popular taxonomic similarity functions, in particular in terms of a theoretical analysis whether or under which conditions they constitute valid kernel functions.

Part III reports on the second and third research question.

Chapter 5 investigates how knowledge about term similarities can be exploited to design adequate kernel functions in text mining applications. The chapter introduces two new families of kernel functions, referred to as Semantic Smoothing Kernels and Semantic Syntactic Tree Kernels which take the varying degrees of semantic similarity between terms into account. The section introduces their formal model, gives examples and provides an analysis of the the issues for application.

Chapter 6 investigates the application of both types of kernels presented in Chapter 5 by means of an extensive set of experiments on different datasets.

Part IV reports on the fourth research question.

Chapter 7 investigates how instances, represented within ontologies can become the subject of kernel-based learning. The chapter introduces a framework for such kernel functions, which can be combined and adapted to be tuned for a particular use case.

Chapter 8 again investigates the application of the kernel functions proposed in the preceding chapter in practical experiments.

Part V concludes this thesis.

Chapter 9 summarizes the main contributions and findings of this thesis and discusses future research directions.

1.4 Bibliographic Notes

Parts of the work reported in this thesis are also the subject of the following publications:

- Bloehdorn, S. and Hotho, A. (2008). Machine learning and ontologies. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, chapter 31. Springer, Berlin–Heidelberg, Germany, second edition. To appear

- Bloehdorn, S. and Sure, Y. (2007). Kernel methods for mining instance data in ontologies. In Aberer, K., Choi, K.-S., Noy, N. F., Allemang, D., Lee, K.-I., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web — Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC 2007 + ASWC 2007), November 11-15, 2007, Busan, Korea*, volume 4825 of *Lecture Notes in Computer Science*, pages 58–71. Springer, Berlin–Heidelberg, Germany

- Bloehdorn, S. and Moschitti, A. (2007a). Combined syntactic and semantic kernels for text classification. In Amati, G., Carpineto, C., and Romano, G., editors, *Advances in Information Retrieval — Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007), 2-5 April 2007, Rome, Italy*, volume 4425 of *Lecture Notes in Computer Science*, pages 307–318, Berlin–Heidelberg, Germany. Springer, Berlin–Heidelberg, Germany

- Bloehdorn, S. and Moschitti, A. (2007b). Structure and semantics for expressive text kernels. In Silva, M. J., Laender, A. H. F., Baeza-Yates, R. A., McGuinness, D. L., Olstad, B., Olsen, O. H., and Falcao, A. O., editors, *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007), November 6-9, 2007, Lisbon, Portugal*, pages 861–864. ACM Press, New York, NY, USA

- Bloehdorn, S., Basili, R., Cammisa, M., and Moschitti, A. (2006b). Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*. IEEE Computer Society, Washington, DC, USA

- Bloehdorn, S., Basili, R., Cammisa, M., and Moschitti, A. (2006a). Designing semantic kernels as implicit superconcept expansions. In Schaaf, M. and Althoff, K.-D., editors, *LWA 2006: Lernen — Wissensentdeckung — Adaptivität, KDML 2006 :12. Workshop der Fachgruppe Knowledge Discovery, Data Mining und Maschinelles Lernen und des Arbeitskreises Knowledge Discovery, October 9-11, 2006, Hildesheim, Germany*, volume 1/2006 of *Hildesheimer Informatik-Berichte*, pages 255–261. Universität Hildesheim, Institut für Informatik, Germany

- Bloehdorn, S., Cimiano, P., and Hotho, A. (2006c). Learning ontologies to improve text clustering and classification. In Spiliopoulou, M., Kruse, R., Nürnberger, A., Borgelt, C., and Gaul, W., editors, *From Data and Information Analysis to Knowledge Engineering: Proceedings of the 29th Annual Conference of the German Classification Society (GfKl 2005), March 9-11, 2005, Magdeburg, Germany*, volume 30 of *Studies in Classification, Data Analysis, and Knowledge Organization*, pages 334–341. Springer, Berlin–Heidelberg, Germany

- Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., and Oberle, D. (2005). The SWRC ontology - Semantic Web for research communities. In Bento, C., Cardoso, A., and Dias, G., editors, *Progress in Artificial Intelligence — Proceedings of the 12th Portuguese Conference on Artificial Intelligence (EPIA 2005), December 5-8, 2005, Covilhã, Portugal*, volume 3803 of *Lecture Notes in Computer Science*, pages 218–231. Springer, Berlin–Heidelberg, Germany

- Bloehdorn, S., Cimiano, P., Hotho, A., and Staab, S. (2005). An ontology-based framework for text mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 20(1):87–112

Part I

Preliminaries

Chapter 2

Machine Learning and Kernel Methods

In this chapter, we introduce the foundations of machine learning and kernel methods that are relevant for this thesis. The mathematical theory underlying kernel-based learning techniques was developed by Aronszajn (1950) and Mercer (1909). While the first application of kernel theory in a pattern analysis setting is due to Aizerman et al. (1964), it has received widespread attention in the machine learning theory only with the advent of Support Vector Machines (SVMs) (Boser et al., 1992; Cortes and Vapnik, 1995; Vapnik, 1995). Subsequently, SVMs and other kernel-based learning techniques have been successfully applied to learning problems such as recognition of handwritten digits (Cortes and Vapnik, 1995) or text classification (Joachims, 1998, 2002). Since then, various tutorials and books have been written on kernel methods. The following exposition partly relies on the comprehensive presentations by Shawe-Taylor and Cristianini (2004); Burges (1998), as well as Hofmann et al. (2007).

In the following, we provide a general introduction to kernel functions and kernel-based learning techniques. First, Section 2.1 revisits basic mathematical concepts needed for the subsequent presentation. Section 2.2 then presents a generic introduction to elementary machine learning concepts. One of the various merits of kernel methods is the decoupling of the design of learning algorithms and the design of the kernel functions used. As a result, Section 2.3 first presents a selection of state-of-the art kernel-based learning algorithms. It starts out with illustrating the main concepts behind kernel-based learning in the context of a simple example, the kernel perceptron. Based on this informal discussion, it then gives a systematic overview of kernel-based learning algorithms. The main focus are SVMs, both in their original hard-margin formulation and in their extended soft-margin version. Other kernel-based algorithms are also shortly presented. Then, Section 2.4 formally describes kernel functions and discusses their properties. The section also introduces a number of standard kernels and well-known kernel modifiers. Finally, Section 2.5 summarizes the procedures for performance assessment of a learning output. The chapter concludes with a short summary in Section 2.6.

2.1 Mathematical Foundations

In this section, we review the mathematical foundations relevant for the subsequent exposition. Throughout this presentation, we touch a somewhat subjective choice of topics from analysis and linear algebra. Obviously, the range of relevant topics is too broad to be covered in full detail but the presented content should suffice to make the exposition self-contained. In the same spirit, we will sometimes use simplified definitions at the expense of generality. We include proofs for some of the theorems if they are suitable to help in the understanding of the overall setting while pointing to the appropriate literature for the other cases.

2.1.1 Vector Spaces, Inner Products and Hilbert Spaces

In this section, we mainly review the notion and properties of inner products that play a central role in the context of kernel theory. We will begin with the definition of a vector space and gradually extend this concept up to the concept of the Hilbert space.

Sets and Metric Spaces

Mathematics often deals with *sets* made up of *elements* of whatever type, e.g. the sets of real or integer numbers, sets of data items or sets of symbols of a certain alphabet. The set plays a key concept in modern mathematics and sets form the core of most more advanced mathematical structures that are built upon them. The empty set is denoted $\emptyset$. As usual, we will use $a \in \mathcal{A}$ to denote the membership of a in the set $\mathcal{A}$. Given two sets $\mathcal{A}, \mathcal{B}$, well-known operations upon these sets are their *union* $\mathcal{A} \cup \mathcal{B}$, their *intersection* $\mathcal{A} \cap \mathcal{B}$ and their *difference* $\mathcal{A} \setminus \mathcal{B}$ (i.e. the set of all elements of $\mathcal{A}$, which do not belong to $\mathcal{B}$). Sets can stand in a subset-relation to each other, i.e. $\mathcal{A} \subseteq \mathcal{B}$ ($\mathcal{A} \subset \mathcal{B}$) denotes that $\mathcal{A}$ is (proper) subset of $\mathcal{B}$. The power set of a set $\mathcal{S}$ is the set of all its subsets, denoted $2^{\mathcal{A}}$. The set of all pairs of two sets $\mathcal{A}$ and $\mathcal{B}$ is called their *Cartesian product*, denoted $\mathcal{A} \times \mathcal{B}$.

One of the simplest structures that can be imposed upon a set is a distance function as a concept of "closeness" items.

Definition 2.1 (Metric Space)**.** A *metric space* is a set $\mathcal{A}$, along with a function d: $\mathcal{A} \times \mathcal{A} \mapsto \mathbb{R}$, called the *distance* between two elements, such that for any $x, y \in \mathcal{A}$, the following properties hold:

- $d(x, y) \geq 0$ (positivity),
- $d(x, y) = 0 \Leftrightarrow x = y$,
- $d(x, y) = d(y, x)$ (symmetry),

- $d(x,z) + d(z,y) \geq d(x,y)$ (triangle inequality).

Vector Spaces

We now extend the concept of the set towards vector spaces. The concept of the vector space provides means for combining and transforming items by means of a set of fixed operations. The vector space is the basic structure in which the learning techniques introduced in the remainder will operate.

Definition 2.2 (Vector Space)**.** A set $\mathcal{V}$ is a *vector space over a field* $\mathbb{K}$, if the two operations addition ($+$) and multiplication by a scalar ($\times$) are defined on $\mathcal{V}$ such that for any $x, y, z \in \mathcal{V}$ and $\alpha, \beta \in \mathbb{K}$ we have $x + y \in \mathcal{V}$ and $\alpha \times x \in \mathcal{V}$ whereby $\mathcal{V}$ forms a *commutative group* under addition, i.e.:

- $x + (y + z) = (x + y) + z$ (associativity),
- there is an element $\mathbf{0} \in \mathcal{V}$, such that $x + \mathbf{0} = x$ (identity),
- for every $x \in \mathcal{V}$ there is a unique element $-x \in \mathcal{V}$, such that $x + (-x) = \mathbf{0}$ (inverse),
- $x + y = y + x$ (commutativity) ,

and where scalar multiplication obeys the following axioms:

- $\alpha \times (\beta \times x) = (\alpha \times \beta) \times x$ (associativity),
- $\alpha \times (x + y) = \alpha \times x + \alpha \times y$ (distributivity of vector sums),
- $(\alpha + \beta) \times x = \alpha \times x + \beta \times x$ (distributivity of scalar sums),
- there is an element $1 \in \mathbb{K}$ such that $1 \times x = x$ (identity).

Elements $x \in \mathcal{V}$ are called *vectors*, elements $\alpha \in \mathbb{K}$ are called *scalars*. A non-empty subset $\mathcal{M} \subseteq \mathcal{V}$ is called a *subspace* of $\mathcal{V}$ if the restriction of the above axioms to $\mathcal{M}$ make it a vector space.

As a convention, we will abbreviate $\alpha \times x$ as αx. In the following, we will restrict the attention to vector spaces over the field $\mathbb{R}$ of real numbers which is sufficient for the required results and simplifies the further discussion. Note however that most of the presented results carry over to other fields with minor modifications.

Example 2.1 (d-dimensional Euclidean Space)**.** The standard example of a vector space is the set $\mathbb{R}^d$ of real column vectors of fixed dimensionality d. Using x' to denote the transpose of vector x, such a vector can be written as $x = (x_1, \ldots, x_d)$ where $x_i \in \mathbb{R}, i = 1, \ldots, d$.

While the term *vector* is often used synonymously with the concept of vectors in Euclidean spaces, functional analysis generalizes this concept to arbitrary abstractions of Euclidean vectors whose precise nature (as, for example, sequences or functions of some kind) is unimportant for the basic concept of vector spaces.

Definition 2.3 (Linear and Convex Combinations)**.** Given a vector space $\mathcal{V}$, some elements $x_1, \ldots, x_n \in \mathcal{V}$ and scalars $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$, a sum of the form $\alpha_1 x_1 + \ldots \alpha_n x_n$ is called a *linear combination* of $x_1, \ldots, x_n$. If $\alpha_1, \ldots, \alpha_n \geq 0$ and $\sum_{i=1}^{n} \alpha_i = 1$, the sum is also called a *convex combination*. The set of all linear combinations of $x_1, \ldots, x_n \in \mathcal{V}$ is called the *linear hull* of $x_1, \ldots, x_n$, denoted $\operatorname{span}(x_1, \ldots, x_n)$. The set of all convex combinations of $x_1, \ldots, x_n \in \mathcal{V}$ is called the *convex hull* of $x_1, \ldots, x_n$. A set $\mathcal{M} \subseteq \mathcal{V}$ is called a *convex set* if for any two $x, x' \in \mathcal{M}$, all convex combinations of x and x' are also in $\mathcal{M}$.

Definition 2.4 (Linear Independence, Basis)**.** A finite set of of vectors $x_1, \ldots, x_n \in \mathcal{V}$ is called *linearly independent* if and only if there is no trivial linear combination that yields the zero element, i.e. if $\alpha_1 x_1 + \ldots + \alpha_n x_n = \mathbf{0} \Leftrightarrow \alpha_1 = \ldots = \alpha_n = 0$. An arbitrary (possibly infinite) set $\mathcal{M}$ is linearly independent if and only if every finite subset of $\mathcal{M}$ is linearly independent. A set of vectors $\mathcal{M}$ forms a *basis* for a vector space $\mathcal{V}$ if and only if $\mathcal{M}$ is linearly independent and $\operatorname{span}(\mathcal{M}) = \mathcal{V}$. The cardinality of every basis of a given vector space is constant and is called the *dimension* of the vector space.

Normed Vector Spaces

Building on the notion of the simple vector space, we now move on to introducing means for measuring the "length" or "size" of vectors.

Definition 2.5 (Normed Vector Space)**.** A *normed vector space* is a vector space $\mathcal{V}$ along with a function $\|\cdot\|$: $\mathcal{V} \mapsto \mathbb{R}$, called the *norm* of $\mathcal{V}$, such that for any $x, y \in \mathcal{V}$ and $\alpha \in \mathbb{R}$, the following properties hold:

- $\|x\| \geq 0, \forall x \in \mathcal{V}$, whereby $\|x\| = 0 \Leftrightarrow x = \mathbf{0}$ (positivity),
- $\|x + y\| \leq \|x\| + \|y\|, \forall x, y \in \mathcal{V}$ (triangle inequality),
- $\|\alpha x\| = |\alpha| \|x\|, \forall \alpha \in \mathbb{K}, \forall x \in \mathcal{V}$ (homogeneity).

Remark 2.1. A valid norm also induces a *metric* $d(\cdot, \cdot)$ by defining the *distance* d as $d(x, y) = \|x - y\|, x, y \in \mathcal{V}$.

Example 2.2 (ℓ^p norm on $\mathbb{R}^d$)**.** In a real vector space $\mathbb{R}^d$, for any $p \geq 1$ the ℓ^p norm of a vector $x \in \mathbb{R}^d$ with components $x_1, \ldots, x_d$ is given by:

$$\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{\frac{1}{p}} .$$

For the case of $p = 1$ we get the so called *Manhattan norm*, while $p = 2$ yields the standard *Euclidean norm*.

Definition 2.6 (Banach Space)**.** In a normed vector space $\mathcal{V}$, an infinite sequence of vectors $x_1, x_2, x_3, \ldots$ is said to *converge* to a vector x, if the sequence of distances $d(x, x_n)$, induced by the norm of $\mathcal{V}$, converges to zero. Furthermore, the sequence is said to be a *Cauchy sequence* if $d(x_n, x_m) \rightarrow 0$ as $n, m \rightarrow \infty$. More precisely, given $\epsilon > 0$, there is an integer k such that $d(x_n, x_m) < \epsilon$ for all $n, m > k$. A space is said to be *complete* when every Cauchy sequence converges to an element of the space. A complete normed vector space is also called a *Banach space*.

Remark 2.2. Note that every real Cauchy sequence is convergent. This follows from the fact that (i) every Cauchy sequence of real numbers is bounded and (ii) by virtue of the well-known Bolzano-Weierstrass Theorem, every bounded sequence has a convergent subsequence with some limit point x. Now assuming that such a limit x is also a limit of the Cauchy sequence, we choose some x_n of the subsequence that is within a fixed distance ϵ of x. Provided we are far enough down the Cauchy sequence, any x_m will be within ϵ of x_n and thus also within 2ϵ of x.

Inner Products and Hilbert Spaces

Having introduced the notion of a vector space and the norm of vectors, we now extend these concepts with the so-called inner product as a notion of the mutual orientation of two vectors in the vector space.

Definition 2.7 (Linear Function)**.** Let $\mathcal{V}$ and $\mathcal{M}$ be vector spaces over $\mathbb{R}$. A function $f : \mathcal{V} \mapsto \mathcal{M}$ is called a *linear function* if for any $x, v \in \mathcal{V}$ and $\alpha \in \mathbb{R}$ the following properties hold: $f(x + y) = f(x) + f(y)$ and $f(\alpha x) = \alpha f(x)$.

Definition 2.8 (Inner Product Space)**.** An *inner product space (pre-Hilbert space)* is a vector space $\mathcal{V}$ over $\mathbb{R}$ along with a function $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \mapsto \mathbb{R}$ such that for any $x, y, z \in \mathcal{V}$ and $\alpha \in \mathbb{K}$ the following properties hold:

- $\langle x, y \rangle = \langle y, x \rangle$ (symmetry),
- $\langle x, x \rangle \geq 0$, and $\langle x, x \rangle = 0 \Leftrightarrow x = \mathbf{0}$ (positivity),
- $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle$ and $\langle x + z, y \rangle = \langle x, y \rangle + \langle x, z \rangle$ ((bi-)linearity).

The map $\langle \cdot, \cdot \rangle$ is called *inner product* or, synonymously, *scalar product* or *dot product*.

Remark 2.3. Inner product spaces have a naturally defined norm via the relation $\|x\| = \sqrt{\langle x, x \rangle}$. By means of its corresponding norm, an inner product thus also induces a metric.

Example 2.3 (Euclidean Inner Product)**.** In a real vector space $\mathbb{R}^d$, the Euclidean inner product of two vectors $x, z \in \mathbb{R}^d$ with components $x_1, \ldots, x_d$ and $z_1, \ldots, z_d$ is given by:

$$\langle x, z \rangle = \sum_{i=1}^{d} x_i z_i.$$

The corresponding norm coincides with the ℓ^2 norm. The related *Frobenius inner product* defines an inner product on matrices by treating them as two-dimensional vectors and summing up the products of corresponding components.

Proposition 2.1 (Cauchy-Schwarz Inequality)**.** *In an inner product space we have that*

$$|\langle x, y \rangle|^2 \leq \langle x, x \rangle \, \langle y, y \rangle$$

and the equality holds if and only if x and y are dependent.

Definition 2.9. The *angle* between two vectors $x, y \in \mathcal{V}$ is defined by:

$$\cos\theta = \frac{\langle x, y \rangle}{\|x\| \, \|y\|}.$$

If $|\langle x, y \rangle| = \|x\| \, \|y\|$, the cosine is equal to plus or minus one, the angle θ is zero, and x and y are said to be *parallel*. If $\langle x, y \rangle = 0$, the cosine is zero, the angle $\theta = \frac{\pi}{2}$ and the vectors are said to be *orthogonal*.

Definition 2.10. A finite set of of vectors $x_1, \ldots, x_n \in \mathcal{V}$ is called *orthogonal* if all pairs of vectors are distinct and orthogonal. The set is called *orthonormal*, if it is orthogonal and all vectors have unit norm. An orthogonal set is always linearly independent. A basis which forms an orthogonal or orthonormal set is called an orthogonal or orthonormal basis, respectively.

Definition 2.11 (Gram Matrix)**.** Given a set $\mathcal{M} = \{x_1, \ldots, x_n\}$ of vectors from an inner product space $\mathcal{V}$, the $n \times n$ matrix $\mathbf{G}$ with entries $\mathbf{G}_{ij} = \langle x_i, x_j \rangle$ is called the *Gram matrix* of $\mathcal{M}$.

There are different definitions of the term Hilbert space, we here use the more relaxed definition, which is sufficient for our purpose.[1]

Definition 2.12 (Hilbert Space)**.** A vector space $\mathcal{H}$ is called a *Hilbert space* if it is an inner product space and $\mathcal{H}$ is complete with respect to the metric induced by the inner product $\langle \cdot, \cdot \rangle$ (i.e. if it is also a Banach space).

[1]For example, Kolmogorov and Fomin (1970) define a Hilbert space as above but restrict the definition to infinite-dimensional and separable (in which case, informally speaking, the space must have a countable subset whose closure is the space itself) spaces.

Example 2.4 (Euclidean Space). A real vector space $\mathbb{R}^d$, together with an Euclidean inner product as introduced in example 2.3 is a Hilbert space.

Example 2.5 (L^2 Space). The space L^2 of all functions $f : \mathbb{R} \mapsto \mathbb{R}$ such that the integral of f^2 over $[-\infty, \infty]$ is finite, together with the inner product

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x) g(x) dx$$

is an example of an (infinite-dimensional) Hilbert space.

2.1.2 Fundamental Concepts from Matrix Algebra

In this section, we will constrain the exposition to real matrices. Recall that a real $n \times m$ *matrix* $\mathbf{M}$ is a rectangular arrangement of real numbers along n rows and m columns. By $\mathbf{M}_{ij}$ we denote the real number in the i-th row and j-th column. Every row and column of a real matrix forms a vector in the Euclidean spaces $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively. We denote the transpose of a matrix $\mathbf{M}$ as $\mathbf{M}'$. All linear functions $f : \mathbb{R}^m \mapsto \mathbb{R}^n$ can be encoded as matrix multiplications $f(x) = \mathbf{M}x, x \in \mathbb{R}^m, f(x) \in \mathbb{R}^n, \mathbf{M} \in \mathbb{R}^{n \times m}$. A square matrix is called *symmetric* if $\mathbf{M} = \mathbf{M}'$. A square matrix is called *diagonal* if its off-diagonal cells are all zero. It is called *upper (lower) triangular* if all elements above (below) the diagonal are all zero.

A diagonal matrix $\mathbf{I}$ (of dimensionality d) whose diagonal entries are all equal to '1' is called the *identity matrix* (of dimensionality d). The *Kronecker symbol*, defined as:

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

conveniently expresses the entries of the identity matrix.

Further, recall that the *rank* of a matrix is the number of linearly independent rows or columns of the matrix. An $n \times n$ matrix $\mathbf{M}$ (*square matrix*) is called *regular* if its rank is equal to n, otherwise some of its vectors must be linearly dependent in which case $\mathbf{M}$ is called *singular*. For a regular matrix $\mathbf{M}$ there exists a unique matrix denoted $\mathbf{M}^{-1}$ such that $\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}$ which is called the *inverse* of $\mathbf{M}$. A singular matrix does not have an inverse. A matrix $\mathbf{M}$ for which $\mathbf{M}^{-1} = \mathbf{M}'$ is called an *unitary matrix*.

Also recall that the *determinant* of a square matrix $\mathbf{M}$, denoted $\det(\mathbf{M})$ is a function that associates a scalar value to $\mathbf{M}$ that reveals certain properties of $\mathbf{M}$.

Definition 2.13 (Determinant). Given a $n \times n$ matrix $\mathbf{M}$, the determinant is computed as

$$\det \mathbf{M} = \sum_{\sigma \in P_n} \left(\operatorname{sign}(\sigma) \prod_{i=1}^{n} \mathbf{M}_{i,\sigma(i)} \right)$$

whereby P_n denotes all permutations of σ of $\{1,\ldots,n\}$ and $\mathrm{sign}(\sigma)$ denotes the sign of the permutation σ, namely $\mathrm{sign}(\sigma) = 1$ in case of an even permutation and $\mathrm{sign}(\sigma) = -1$ in case of an uneven permutation.

Obviously, in the case of larger matrices, the computation of the determinant is a bit more involved. For 2×2 matrices, the determinant can be conveniently expressed as $\det(\mathbf{M}) = \mathbf{M}_{11}\mathbf{M}_{22} - \mathbf{M}_{12}\mathbf{M}_{21}$. Without going into too much detail, we sketch some important properties of the determinant. If the determinant of a matrix is zero, the matrix is singular, otherwise it is regular. Specifically, if the determinant is one, the matrix is said to be unimodular.

Definition 2.14 (Eigenvalues and Eigenvectors)**.** Given an $n \times n$ matrix $\mathbf{M}$, the real number λ and the vector $x \in \mathbb{R}^n$ are called *eigenvalue* and *eigenvector*, respectively if $\mathbf{M}x = \lambda x$.

Obviously, a matrix can only have zero as an eigenvalue if it is singular with the respective eigenvector(s) corresponding to a non-trivial combination of the column vectors that yields the zero vector. Note that the product of the eigenvalues of a square matrix is equal to the determinant of that matrix.

Proposition 2.2. *Eigenvectors corresponding to distinct eigenvalues are orthogonal.*

Proof. Let x, λ and z, μ be eigenvector, eigenvalue pairs with $\lambda \neq \mu$ and $\lambda, \mu > 0$ for some symmetric matrix $\mathbf{M}$. We then have that $\lambda\langle x, z\rangle = \langle \mathbf{M}x, z\rangle = \langle x, \mathbf{M}z\rangle = \mu\langle x, z\rangle$. But this implies that $\langle x, z\rangle = 0$. $\square$

As a result, an $n \times n$ matrix can have at most n distinct eigenvalues. The Eigenvectors can be chosen to form an orthonormal basis of $\mathbb{R}^n$.

If we form a diagonal matrix $\mathbf{\Lambda}$ with eigenvalues λ_i of a square matrix $\mathbf{M}$ arranged in decreasing order along the main diagonal and a matrix $\mathbf{V}$ with the corresponding orthonormal eigenvectors arranged columnwise, we have that $\mathbf{V}\mathbf{V}' = \mathbf{V}'\mathbf{V} = \mathbf{I}$ forming the identity matrix and $\mathbf{M}\mathbf{V} = \mathbf{V}\mathbf{\Lambda}$. The latter decomposition is referred to as the *eigen-decomposition* of $\mathbf{M}$. In the case that all eigenvalues are non-negative, they can have valid square roots. Setting $\mathbf{A} = \mathbf{V}\sqrt{\mathbf{\Lambda}}$, the eigen-decomposition can thus also be written as $\mathbf{M} = \mathbf{A}\mathbf{A}'$.

Definition 2.15 (Positive Semi-Definiteness)**.** A symmetric matrix $\mathbf{M}$ is *positive semi-definite (p.s.d.)*, if all eigenvalues of $\mathbf{M}$ are non-negative.

Proposition 2.3. *Let* $\mathbf{M}$ *be a symmetric matrix. Then* $\mathbf{M}$ *is positive semi-definite if and only if for any vector* $x \neq \mathbf{0}$ *we have that* $x'\mathbf{M}x \geq 0$.

Proof. Let $\mathbf{M}$ be a symmetric matrix $n \times n$ and $x \neq \mathbf{0}$ any vector in $\mathbb{R}^n$. Consider the rearrangement of the above condition in terms of the eigen-decomposition of $\mathbf{M}$:

$x'\mathbf{M}x = x'\mathbf{V}'\mathbf{\Lambda}\mathbf{V}x = \langle \mathbf{\Lambda}\mathbf{V}x, \mathbf{V}x\rangle \geq 0$. It clearly follows that if all eigenvalues $\lambda_i \geq 0$, the condition is met as the resulting dot product will be non-negative. Let x be an eigenvector of $\mathbf{M}$ with eigenvalue λ. But then we have $x'\mathbf{M}x = x'\,\lambda x = \lambda\langle x, x\rangle \geq 0 \Rightarrow \lambda \geq 0$. □

The interested reader is pointed to the treatment of positive (semi-) definite matrices by Bhatia (2007).

Definition 2.16 (Singular Values and Singular Vectors)**.** Given a $m \times n$ matrix $\mathbf{M}$, the *singular value decomposition (SVD)* of $\mathbf{M}$ is given by the decomposition $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}'$, whereby $\mathbf{U}$ is a $m \times m$ unitary matrix, the matrix $\mathbf{\Sigma}$ is a $m \times n$ matrix with non-negative numbers on the diagonal and zeros off the diagonal, and $\mathbf{V}$ is a $n \times n$ unitary matrix. Further, the values $\sigma_1, \ldots, \sigma_k$ with $k = \min(n, m)$ lie the diagonal of $\mathbf{\Sigma}$ in decreasing arrangement and are called the *singular values* of $\mathbf{M}$.

Remark 2.4. The SVD of $\mathbf{M}$ is related to the eigendecomposition of $\mathbf{M}\mathbf{M}'$ as follows

$$\mathbf{M}\mathbf{M}' = \mathbf{U}\mathbf{\Sigma}\mathbf{V}'\mathbf{V}\mathbf{\Sigma}\mathbf{U}' = \mathbf{U}\mathbf{\Sigma}\mathbf{I}\mathbf{\Sigma}\mathbf{U}' = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$$

where $\mathbf{\Lambda} = \mathbf{\Sigma}^2$ now is a diagonal matrix containing the (non-negative) eigenvalues of $\mathbf{M}\mathbf{M}'$. The columns of $\mathbf{U}$ are thus also called the *left singular vectors* of $\mathbf{M}$. The same reasoning can be applied to the matrix $\mathbf{M}'\mathbf{M}$ and, correspondingly, the columns of $\mathbf{V}$ are called the *right singular vectors* of $\mathbf{M}$.

To this point, we have introduced the main building blocks of analysis and linear algebra required to express machine learning as function estimation in vector spaces and to investigate the concept of kernel functions in closer detail.

2.2 Machine Learning as Pattern Analysis

Machine learning is a wide research field concerned with methods that deal with the automatic detection of *patterns* in data. See e.g. Mitchell (1997) for a classical textbook. Machine learning is a subdiscipline of the much broader field of *artificial intelligence* which is occupied with devising computational techniques that mimic human intelligent behaviour. The term *machine learning* points to the roots of this perspective as it emphasizes the analogy with the kind of learning we know from humans and animals. After observing or experiencing a certain recurring situation a number of times we learn to recognize the same kind of situation again, even if the amount of presented information is limited. Machine learning aims at achieving a similar effect. Presented with a limited amount of data, machine learning techniques try to learn about hidden patterns in the data and to exploit the results in a corresponding application. Along this line, the term *pattern analysis* is used somewhat synonymously with machine learning. The patterns of interest can be any kind of structure, relations and regularities that can be observed in the data. Typically, the pattern analysis process relies on statistical concepts and, consequently, the term *statistical learning* is used in literature as yet another synonym that stresses the statistical nature of the employed methods.[2]

2.2.1 Classes and Properties of Machine Learning Problems

In the classical machine learning scenario, we are presented a set $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ of data items (with $\mathcal{S} \subset \mathcal{X}$) and try to detect patterns in this data. The x_i are called *(training) instances* and the space $\mathcal{X}$ they originate from is called the *input space*. The type of pattern that we are looking for is constrained by various aspects, most prominently by the type of the learning task.

Supervised Learning

In *supervised learning*, we are interested in a particular property of the data items. This property, typically called the *target*, is a single discrete or real value linked to each instance. The patterns we are looking for have a *predictive* nature: given a limited amount of input-output pairs, we are interested in finding those patterns that are capable of accurately predicting the target variable from the input instances alone. Depending on the space $\mathcal{Y}$ of the target variables, the problem is referred to as *classification* (discrete target variables) or *regression* (real-values variables). We will give a more formal account of supervised learning in the next section. These kind of tasks are considered *supervised* because a (human) authority outside the system decides upon the correct target variables.

Example 2.6 (e-mail Classification)**.** As an example, consider the case of teaching a spam filter application by presenting it a number of e-mails, marked as *"spam"* or *"non-spam"*. The learning task is to detect those patterns in the e-mails that are capable of determining the nature of a new, unclassified, e-mail.

Unsupervised Learning

In *unsupervised learning*, the patterns we are looking for have a *descriptive* nature. In these cases, we can only observe the input instances and try to describe in a more

[2]In fact, literature uses yet another set of names for this field. With a tendency on stressing the application context of pattern analysis methods, *Data Mining* is another term frequently used, typically in conjunction with the term *Knowledge Discovery* for denoting the larger context in which Data Mining techniques operate (Fayyad et al., 1996). In particular, the Knowledge Discovery Process starts from the definition and analysis of the (business) problem, followed by phases for understanding and preparation of data, setup of the learning model, the actual learning, application and evaluation of the results, and, finally, deployment of the solution.

compact way, how the data is organized. A typical unsupervised learning task is *clustering*, where we are concerned with grouping the instances in such a way that the items within each group are homogeneous in some way while heterogeneous items should be placed in distinct clusters. Another unsupervised learning problem is *association rule learning* that aims at identifying co-occurrence patterns of properties of the input instances. Finally, *dimensionality reduction* techniques try to discover projections of arbitrary data items into low dimensional spaces, typically for visualization purposes or as input to other learning algorithms. Here, the projection should retain as much of the pairwise similarities and distributional characteristics of the original data.

Common Principles

Regardless of the type of the learning task and the specific algorithm employed, all pattern analysis endeavours exhibit a common structure:

(1) They require the choice of a suitable data representation.

(2) They require the choice of a suitable class of hypothesis functions $\mathcal{F}$ (model).

(3) They require the specification of an algorithm that, given a set of training examples, identifies a specific function $f \in \mathcal{F}$ by determining specific values for the free parameters of the function class $\mathcal{F}$ (learning).

Each of these aspects critically determines the success of the learning problem. This thesis is mainly concerned with the first two aspects, as kernel functions can be seen as the interface between the data representation and the model. As a consequence, the use of kernel functions restricts the types of models and the learning processes that can be used. However, as we will see, these restrictions are rather mild and the use of kernels enables a wide range of patterns to be accurately modelled and detected.

Along another dimension, Shawe-Taylor and Cristianini (2004) name three desirable properties of modern machine learning techniques. On the one hand, the techniques should be capable of dealing with the fact that real-life data is often corrupted by *noise*, e.g. because of measurement inaccuracies. Algorithms should thus show *robustness* in the sense of tolerating a certain amount of noise while still producing approximate patterns. On the other hand, the algorithms should exhibit *statistical stability* in the sense that regardless of the precise sample supplied, the algorithms should reproduce largely similar results. In particular, the algorithms should not be easily affected by variations in the distribution of instances or by outliers, i.e. rare instances with extreme deviations. Later, we will investigate a set of concepts that directly ensure these properties when investigating large-margin classifiers. Finally, the algorithms should be *computationally efficient*. This requirement is critical when algorithms are to be used in practical applications with large amounts of data.

2.2.2 Supervised Learning and Statistical Learning Theory

Statistical learning theory is a field occupied with investigating general principles and properties of (mostly supervised) learning techniques. In this section, we will review the most relevant findings of this theory. The content of this section is biased towards the case of binary classification problems. Multi-classification can be achieved by various schemes for combining binary classifiers, some of which will be reviewed later. While we will not investigate the case of regression problems, note that many of the results for binary classification can be adapted to regression settings — often in a straightforward way.

The Learning Problem of Binary Classification

In the following, we denote the instance space as $\mathcal{X}$ and the set of possible targets as $\mathcal{Y}$. For the moment, we will entirely focus on the case of binary classification in which case the targets are given by $\mathcal{Y} = \{+1, -1\}$. The classical learning model of statistical learning theory as introduced by Vapnik (1995, 1998) formalizes the classification task as follows. Given input-output training data pairs generated independently and identically distributed (i.i.d.) according to an unknown probability distribution $P(x, y)$:

$$(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}, \mathcal{Y} = \{+1, -1\}\,, \tag{2.1}$$

and a class of functions $\mathcal{F}$, we want to estimate a *discriminant function*:

$$f : \mathcal{X} \mapsto \{-1, +1\},\ f \in \mathcal{F}\,, \tag{2.2}$$

such that f will correctly classify unseen examples (x, y). Given a hypothesis space $\mathcal{F}$, the optimal function f is the one that minimizes the *expected risk* associated with the classifier.

Definition 2.17 (Expected Risk)**.** The *expected risk* of a hypothesis f with respect to a distribution $P(\cdot, \cdot)$ over $(\mathcal{X} \times \mathcal{Y})$ is the quantity:

$$R[f] = \int L(f(x), y)\, dP(x, y),$$

where L denotes a suitably chosen *loss function*.

Formally, the loss function states, how costly a misclassification is.

Definition 2.18 (Loss Function)**.** Given the prediction $f(x)$ of a classifier f on a particular data item x and the true value y of that data item, the loss *loss function* $L(f(x), y)$ measures the loss incurred by the classification decision. Formally, L can be any map $L : \mathcal{Y} \times \mathcal{Y} \mapsto [0, \infty]$ such that $L(y, y) = 0$ for all $y \in \mathcal{Y}$.

Example 2.7 (Zero/One Loss Function)**.** In binary classification, a common and intuitive loss function that weights any kind of misclassification equally, we typically consider the *zero/one-loss* function:

$$L_{0/1}(f(x), y) = \begin{cases} 1 & \text{if } f(x) \neq y \\ 0 & \text{otherwise} \, . \end{cases}$$

Thus, the zero/one loss function assigns a fixed error score if the true labels and the labels predicted by the classifier differ, while assigning zero loss if they coincide. Other loss functions can be defined to account for the fact that some kinds of classification mistakes may be worse, and thus more costly, than others. We will encounter examples of other loss functions, such as the *soft margin loss* in the context of the subsequent treatment of SVMs. As another example, the *squared loss* is also commonly used, especially in regression settings.

The expected risk tells us about the *generalization capabilities*, i.e. the predictive performance of a learned discriminant function. These definitions in mind, we can view the task of estimating an optimal prediction function as a search for a specific function within a given function class that minimizes the expected risk. However, the expected risk can not be calculated directly as the distribution $P(x, y)$ is unknown. As a consequence, we try to estimate a function that is close to the optimal one based on the information available in the training sample and on the function class $\mathcal{F}$.

Empirical Risk Minimization

A straightforward approach would be to estimate the expected risk by means of the *empirical risk* and minimize this quantity instead.

Definition 2.19 (Empirical Risk)**.** The *empirical risk* of a hypothesis f on a data sample $\mathcal{S} = \{(x_1, y_i), \ldots, (x_n, y_n)\}$ is given by:

$$R_{emp}[f, \mathcal{S}] = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i)$$

where L again denotes the chosen loss function.

A learning technique that is based on the minimization of this quantity is said to follow the *empirical risk minimization (ERM) principle*. Unfortunately, the minimization of the empirical error on the training sample *as such* does, in general, not necessarily lead to a classifier with optimal generalization behavior. Instead, we may run into a situation of deriving a (false) model that incurs a substantial expected risk but perfectly fits the training data. This situation, called *overfitting*, is likely to happen if the model has too much complexity, i.e. too many free parameters, in relation to the amount of data available. In cases of overfitting, the learning algorithm may adjust the various model parameters to spurious and random features of the training data, even if they have no causal relation to the target function.

As a means to avoid overfitting, we may try to increase the amount of available training data. It is indeed possible to give conditions on the learning technique that ensure that the empirical risk converges to the expected risk in the size of the training sample. However, the amount of training data is generally limited and other means for controlling the expected risk are required.

Complexity, VC Dimension and Structural Risk Minimization

The second quantity that determines the generalization power of the system is the complexity of the employed function class. One way to avoid overfitting is thus to restrict the complexity of the function class $\mathcal{F}$ under investigation. The intuition, which we will investigate more formally in the following is to favour simple over complex functions. On the other hand, given a function too simple for the learning task at hand, we will again fail to mimic the unknown target concept, or, as Burges (1998) put it:

> "A machine with too much capacity is like a botanist with a photographic memory, who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before; a machine with too little capacity is like the botanist's lazy brother, who declares that if it is green, it is a tree."

In essence, the optimal result for a given learning task will be achieved if the right tradeoff is found between minimizing the empirical error and controlling the complexity of the function class under investigation. For example, a typical approach taken in statistics is to add a regularization term to the empirical risk that penalizes complexity. The problem of generalization has been treated in many different ways, be it as overfitting, as the *bias-variance tradeoff* (Geman et al., 1992), regularization, or from the viewpoint of philosophy of science.[3] In essence, all theoretical background work requires us to find a compromise between optimizing the empirical error and minimizing the model complexity. The subsequent exposition will be based on a particular notion of complexity, the *VC dimension*.

[3] In fact, overfitting is typically considered to be a violation of the philosophical principle of *Occam's razor*, roughly stating that we should make as few assumptions about as possible when trying to explain a given phenomenon.

A common measure of complexity of a function class $\mathcal{F}$ that had substantial impact on modern statistical learning theory is the *Vapnik-Chervonenkis dimension* due to Vapnik and Chervonenkis (1971, 1974).

Definition 2.20 (Vapnik-Chervonenkis (VC) dimension)**.** A set of m data points is *shattered* by functions of a class $\mathcal{F}$, if for all of the 2^m possible labellings a function $f \in \mathcal{F}$ is capable of separating both classes. The *Vapnik-Chervonenkis (VC) dimension* of a class of functions $\mathcal{F}$ corresponds to the maximum number of points that can be shattered by functions of the class.

Vapnik (1995) derived a bound on the expected risk that makes use of the notion of the VC dimension.

Proposition 2.4 (Bound on the Expected Risk (Vapnik, 1995))**.** *Let h denote the VC dimension of the function class $\mathcal{F}$ and let R_{emp} be defined as in Definition 2.19 using the zero/one loss function obtained on the training set $\mathcal{S}$. For all $\eta \in [0,1]$ and $f \in \mathcal{F}$ the inequality*

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{h(\ln \frac{2|\mathcal{S}|}{h} + 1) - \ln(\eta/4)}{|\mathcal{S}|}}$$

holds with probability of at least $1 - \eta$ for $|\mathcal{S}| > h$.

We thus get a means for bounding the expected risk in terms of all relevant parameters, namely the empirical risk R_{emp}, the number of training examples $|\mathcal{S}|$ and the complexity of the employed function class in terms of the VC dimension. Fixing the probability η and the training set, and given two learning functions that yield the same empirical risk, we will thus rationally choose the one whose function class comes with a smaller VC dimension. The bound on the expected risk of Proposition 2.4 makes it possible to work on the basis of the *structural risk minimization (SRM) principle*. The principle advises us to first arrange the hypothesis functions in a nested family of function classes $\mathcal{F}_1 \subset \mathcal{F}_2 \ldots \subset \mathcal{F}_k$ with decreasing VC dimension. Within each class of functions, we then estimate a prediction function solely on the basis of minimizing the empirical risk. We finally choose the function, for which the right hand side of the bound of Proposition 2.4 is minimized.

Despite its appealing properties, the bound of Proposition 2.4 *per se* has limited practical implications. On the one hand, it is often hard to compute as in cases of infinite VC dimension. On the other hand, it may produce trivial results (such as the bound on the expected risk being larger than one). However, the bounds offer interesting theoretical insights that can be exploited to devise appropriate learning schemes. A particularly interesting relation to another quantity, the *margin* of a linear classification function that motivates much of the theory behind SVMs will be discussed in Section 2.3.2.

2.2.3 Unsupervised Learning

With regard to unsupervised learning problems, we only shortly cover the case of clustering as a specific case of unsupervised learning. However, also other techniques like association rule learning or dimensionality reduction techniques could be classified as unsupervised learning. For the case of dimensionality reduction, another important class of unsupervised learning techniques, we will shortly cover a prominent approach called Latent Semantic Indexing (LSI) in the context of the related work in Section 5.5.

Clustering can be seen as the process of segmenting or partitioning a data set into subsets. The site, spatial location, mutual distance and homogeneity of the found clusters can be used for a compact description of the properties of the data. The general objective thereby is that the objects within a cluster are closer or more similar to each other (intra-cluster homogeneity) than to objects outside of the cluster and that the objects across different clusters are as distant or dissimilar as possible (inter-cluster heterogeneity).

The critical notion for all clustering techniques is the notion of distance metric (or, sometimes, arbitrary similarity function) of data items upon which the clustering is based. As we will see, kernel functions provide a good basis for this kind of notion. While several clustering procedures operate heuristically, most techniques implicitly or explicitly optimize a clustering criterion function. The optimization can be based on notions like intra-cluster variance, the sum of squared errors, or the size (in terms of the diameter) of the clusters. The actual clustering can be done in a number of ways. Data clustering algorithms can be *partitional* or *hierarchical*. Beyond the basic objective of partitional clustering, hierarchical clustering also produces a nested hierarchy of groups of objects. Hierarchical algorithms can be agglomerative ("bottom-up") or divisive ("top-down"). While agglomerative clustering algorithms begin with one-element clusters and iteratively merge them into larger clusters, divisive techniques first put the whole dataset into one cluster and successively divide it into smaller clusters. Along another dimension, the produced clustering can be *hard*, or *fuzzy* where each item is assigned to the whole set of clusters with a variable degree of membership. Additionally, clustering algorithms sometimes make explicit assumptions about the structure of the clusters, e.g. many algorithms assume an unknown Gaussian mixture model for the generation of the data.

Various variants of these clustering techniques have been proposed and lead to different clustering techniques. Jain et al. (1999) provides an overview of relevant approaches.

2.3 Kernel-Based Learning Algorithms

In this section we introduce the concept of kernel-based learning algorithms. Starting with an informal discussion of the concept of kernel functions in the context of the *kernel perceptron*, we will then move on to other popular kernel-based learning algorithms, in particular Support Vector Machines which form the basis the practical experiments in this thesis.

2.3.1 Introducing Kernel-Based Learning: The Kernel Perceptron

The perceptron (Rosenblatt, 1958) is one of the earliest machine learning techniques. We use the perceptron paradigm to informally introduce the concept of kernel functions.

Linear Classification and Perceptron Learning

The perceptron learning technique belongs to the family of *linear classifiers*, a large and powerful class of learning methods that will also form the basis for the SVM techniques discussed later on.

Definition 2.21 (Linear Classifiers)**.** Linear classifiers are defined by discriminant functions of the form

$$f(x) = \operatorname{sign}\left(\sum_{j=1}^{d} x^j w^j + b\right) = \operatorname{sign}(\langle x, w\rangle + b);\ w, x \in \mathbb{R}^d, b \in \mathbb{R}.$$

Given a set of input examples in the form of vector-label pairs $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^d \times Y, Y = \{+1, -1\}$, linear classifiers try to estimate the parameters w and b according to a given optimality criterion.

Since the discriminant function is a threshold linear function, we can interpret the learning process as searching for a (d-1)-dimensional *separating hyperplane* defined by the equation $\langle x, w\rangle + b = 0$, each side of which corresponds to a particular classification decision.[4] The *weight vector* w acts as normal vector for this hyperplane while the *bias* parameter b determines the intercept. If $b = 0$, the hyperplane passes through the origin. Figure 2.1 illustrates the situation. Note that there is a whole family of parameters (w, b) that define the same hyperplane as multiplication of both parameters by a positive scalar does not change the classification result. Similarly, multiplication of the parameters by a negative scalar leads to the same classification of the dataset

[4]Formally, the class membership of the points *on* the hyperplane remains undefined. Typically, algorithms use a simple heuristic for this rare situation, e.g. by regarding points on the hyperplane as part of the positive target class.

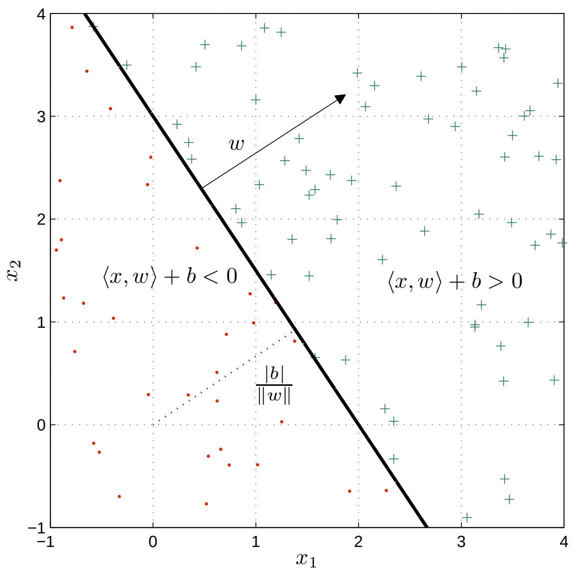


Figure 2.1 — Linear classification boundary in 2D. The decision boundary is given by a hyperplane, i.e. a straight line in 2D. Each side of the hyperplane corresponds to a particular classification decision.

but leads to a flipping of the target classes. As we will see later, this freedom can be restricted by requesting that $\|w\| = 1$. We will refer to this solution as the *canonical representation* of a hyperplane. The magnitude of a weight vector component can be seen as measuring the importance of the respective dimension (feature).

While initial insights and applications of linear classification have been developed by Fisher (1936) as early as in the 1930s, the perceptron algorithm was developed by Rosenblatt (1958) in the late 1950s. Perceptrons set the foundations for the field of neural networks emerging in the 1980s.[5] Procedure 2.1 sketches the perceptron learning algorithm in pseudocode. In Definition 2.21, we have deliberately abstained from defining a specific optimality criterion, as this choice differs among the different learning methods. In particular, the perceptron algorithm does not use any advanced objective for discovering the parameters w and b but simply produces a *feasible* solution, i.e. a solution which is capable of classifying the training vectors without errors. Intuitively, such a solution can only exist if the dataset under consideration can actually be separated by a single hyperplane. In this case, we say that the dataset must be *linearly separable*.

[5]In neural network research terminology, individual perceptrons are sometimes called *artificial neurons*, the components of the argument vectors are called *inputs*, the sign function is an example of a simple *activation function* whereby the bias b is typically rewritten as $b = -\theta$ and is called *activation threshold*.

Procedure 2.1
Perceptron Training — primal version (Rosenblatt, 1958)

```
Input: data $\mathcal{S} = ((x_1, y_1), \ldots, (x_n, y_n))$
Initialize: $w \leftarrow 0$, $b \leftarrow 0$
 1: repeat
 2:   $err \leftarrow 0$
 3:   for $i = 1, \ldots, n$ do
 4:     compute $\gamma_i = y_i(\langle w, x_i \rangle + b)$
 5:     if $\gamma_i < 0$ then
 6:       $w \leftarrow w + y_i x_i$
 7:       $b \leftarrow b + y_i$
 8:       $err \leftarrow err + 1$
 9:     end if
10:   end for
11: until $err = 0$
12: return $w, b$
```

Definition 2.22 (Functional Margin and Linear Separability)**.** The *functional margin* of an example (x_i, y_i) with respect to a hyperplane (w, b) is the quantity:

$$\gamma_i = y_i \left(\langle w, x_i \rangle + b \right).$$

Here, $\gamma_i > 0$ implies correct classification of (x_i, y_i). The functional margin γ of a whole example set $\mathcal{S}$ with respect to a hyperplane (w, b) is the minimum functional margin of all items in the set. The example set $\mathcal{S}$ is said to be *linearly separable* if there exists a hyperplane (w, b) such that $\mathcal{S}$ has a positive functional margin.

As a consequence, the perceptron training algorithm will fail to terminate if the two classes of the training data can not be separated by a hyperplane while it is guaranteed to find a feasible solution after a finite number of steps if the dataset is linearly separable. The situation is summarized in the following proposition.

Proposition 2.5 (Perceptron Convergence (Novikoff, 1962))**.** *Given a non-trivial data set $\mathcal{S} = (x_1, y_1), \ldots, (x_n, y_n)$ and let $R = \max_{1 \le i \le n} \|x_i\|$. Without loss of generality, suppose that there exists a hyperplane (w^*, b^*) such that $\|w^*\| = 1$ and assume that the functional margin γ of $\mathcal{S}$ wrt. (w^*, b^*) is positive, i.e. $y_i \left(\langle w^*, x_i \rangle + b^* \right) \ge \gamma > 0$ for $1 \le i \le n$. Then the number of update steps of the perceptron algorithm on $\mathcal{S}$ is at most $(2R/\gamma)^2$.*

Dual Representation

It is now interesting to observe that the perceptron training algorithm adapts the weight vector w in each iteration by adding or subtracting misclassified training examples, thereby stretching and rotating the weight vector towards the examples under consideration. Assuming that the weight vector is initially set to the zero vector, it can at all times be written as a linear combination of training examples:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i, \tag{2.3}$$

whereby α_i indicates how often a particular training example x_i has been misclassified. This representation of the weight parameter of a hyperplane is typically referred to as the *dual representation*. Based on these observations and the bilinearity of the inner product, we can rewrite the formula in Definition 2.21 (and, equivalently, in Definition 2.22) as follows:

$$\begin{aligned} f(x) &= \operatorname{sign}(\langle x, w \rangle + b) \\ &= \operatorname{sign}\left(\left\langle x, \sum_{i=1}^{n} \alpha_i y_i x_i \right\rangle + b\right) \\ &= \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle + b\right) \end{aligned} \tag{2.4}$$

As a result, the perceptron training algorithm can be entirely rewritten in dual form as done in Procedure 2.2. The dual representation, as an alternative formulation for the decision function and the learning algorithm, has a number of interesting properties. For example, the components α_i of the learned model directly indicate how informative certain training examples are for the overall learning problem. However, as we will see, the most important property of this representation is that all references to the data during training and classification happen within inner products. Under this view, even though we use a geometric concept in a vector space, we do not need to know the precise positions of the vectors, just the inner products between all pairs.

Introducing Kernels

Despite a number of convenient properties, the power of the perceptron technique is limited. In practice, the search heuristic of the perceptron training algorithm may produce suboptimal solutions which do not generalize well while failing altogether when the underlying learning problem is not linearly separable. The notorious example of such a problem is the *XOR problem* introduced by Minsky and Papert (1969). Their main result, namely the fact that the perceptron is not capable of simulating the logical XOR function, led to a decline in the interest in neural networks for a number of years.

A common technique to produce better generalization results for a given dataset is to transform the data and embed it into a different vector space to yield a more

Procedure 2.2
Perceptron Training — dual version

```
Input: data $\mathcal{S} = ((x_1, y_1), \ldots, (x_n, y_n))$
Initialize: $\alpha_1, \ldots, \alpha_\ell \leftarrow 0$, $b \leftarrow 0$
 1: repeat
 2:   $err \leftarrow 0$
 3:   for $i = 1, \ldots, \ell$ do
 4:     compute $\gamma_i = y_i(\sum_{j=1}^{n} \alpha_j y_j \langle x_j, x_i \rangle + b)$
 5:     if $\gamma_i < 0$ then
 6:       $\alpha_i \leftarrow \alpha_i + 1$
 7:       $b \leftarrow b + y_i$
 8:       $err \leftarrow err + 1$
 9:     end if
10:   end for
11: until $err = 0$
12: return $\alpha_1, \ldots, \alpha_n, b$
```

adequate data representation. In particular, this is also a good technique to make a linear learning technique capable of learning non-linear patterns. Formally, this corresponds to defining a feature map $\Phi : \mathcal{X} \mapsto \mathcal{X}^*$ which maps input vectors $x \in \mathcal{X}$ to some (higher dimensional) feature representation. When speaking of this kind of mappings, we will usually refer to $\mathcal{X}$ as the *input space* and to $\mathcal{X}^* = \Phi(\mathcal{X})$ as the *feature space*.

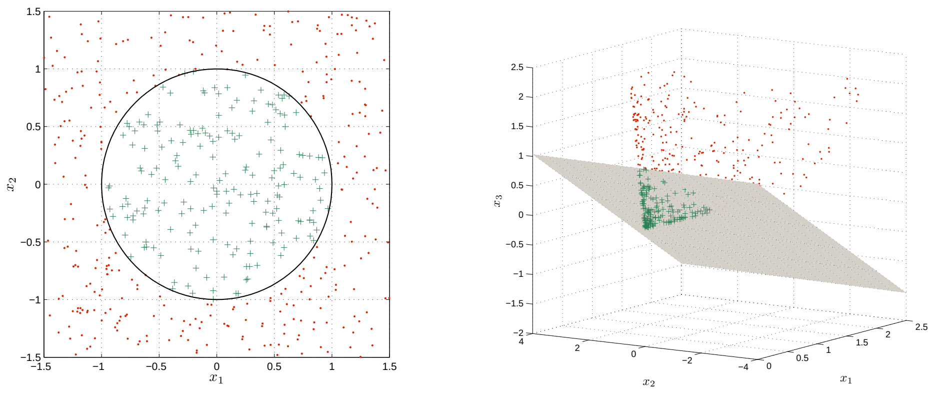
We have seen that the weight vector produced by the perceptron learning algorithm is a linear combination of training data points. Consequently, the evaluation of the decision function as formulated in its dual version in Equation (2.4), relies only on evaluations of the inner product of training data points that contribute to the weight vector and the argument. This makes it possible to replace the techniques of explicit feature mappings by the use of *kernel functions*.

Definition 2.23 (Kernel). A *kernel* is a function $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, such that for all $x, z \in \mathcal{X}$

$$\kappa(x, z) = \langle \phi(x), \phi(z) \rangle.$$

where ϕ is a mapping from $\mathcal{X}$ to an (inner product) feature space $\mathcal{X}^*$.

In general, the kernel function corresponds to any kind of function that is capable of simulating the computation of a dot product in some feature space. Note that, due to the dual formulation in the training process and in the final classification function, this feature space is, as such, not of interest.



(a) Original 2D data. (b) Data mapped into 3D.

Figure 2.2 — 2D sphere learning problem. The data instances within and outside of the sphere are not separable by a linear classification boundary (line) in the original 2-dimensional input space (left), but become separable in the 3-dimensional feature space after the mapping $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$ (right).

Example 2.8 (2D Sphere Learning Problem)**.** A problem similar to the XOR task is illustrated in the left part of Figure 2.2. Here, the two classes are clearly not separable by a linear classification boundary in the original 2-dimensional input space. The right side of Figure 2.2 illustrates the situation after a mapping to a 3-dimensional feature space by means of the transformation $\phi(x) = (x_1^2, \sqrt{2}\,x_1x_2, x_2^2) \in \mathbb{R}^3$. In the new representation, the two groups of data instances can be separated by a linear classification function. The evaluation of dot products can now be rewritten as a closed expression:

$$\begin{aligned}\langle\phi(x),\phi(z)\rangle &= \left\langle(x_1^2,\sqrt{2}x_1x_2,x_2^2),(z_1^2,\sqrt{2}z_1z_2,z_2^2)\right\rangle\\ &= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 = (x_1z_1+x_2z_2)^2 = \langle x,z\rangle^2 = \kappa(x,z).\end{aligned}$$

The resulting kernel is a specific instance of the class of *polynomial kernels* which take the general form $\kappa(x,z) = (\langle x,z\rangle + c)^d$.

Remark 2.5. Usually, the kernel function does not uniquely determine the feature space. For example, the transformation $\phi'(x) = (x_1^2, x_2^2, x_1x_2, x_2x_1) \in \mathbb{R}^4$ also fits to the kernel $\kappa(x,z) = \langle x,z\rangle^2$ of the previous example.

The name "kernel" as used in this context originally stems from integral operator theory, which forms much of the theoretical basis for this field. As the feature vectors need not be represented explicitly, the complexity in terms of computation time and space requirements to compute the inner products need not be proportional to the number of dimensions involved. The use of kernels makes it possible to implicitly map the data into a feature space, train a linear model in that space and (again, implicitly) refer the results back to the original input space. This kind of implicit learning and representation of the resulting model by means of kernel functions is typically referred to as the so-called *kernel trick*. Note that kernels can even be constructed without knowing the feature mapping ϕ associated with them as long as it is possible to prove that such a mapping exists. We will investigate the conditions that ensure kernel validity together with some other theoretical details about kernels in Section 2.4. The use of kernel functions is not restricted to the perceptron algorithm. Instead, a large class of learning algorithms are capable of formulating their solutions as linear combinations of training instances, thereby admitting a reformulation in terms of kernel functions.

2.3.2 Support Vector Machines

After the informal introduction to linear classification and kernels, we are now ready to investigate the best known kernel-based learning algorithm, the *Support Vector Machines (SVMs)* due to Boser et al. (1992). SVMs can be seen as the first practical application of the principle of structural risk minimization. We will first review the notion of the margin of a linear classifier and its relation to the bound on the expected error. We will then introduce the simplest version of the SVM, the *hard-margin SVM* (Boser et al., 1992; Vapnik, 1995), a training algorithm for linear classifiers in cases of separable training data. We then continue with the *soft-margin SVM* (Cortes and Vapnik, 1995), a variant which is also capable of dealing with data that is not linearly separable.

Margins and the VC Dimension

Recall from Definition 2.21 that a linear classification algorithm forms its decisions according to a decision function of the form $f(x) = \operatorname{sign}(\langle x, w \rangle + b)$. Furthermore, Definition 2.22 has introduced the *functional margin* of a training example with respect to a hyperplane as the quantity $\gamma_i = y_i \left(\langle w, x_i \rangle + b \right)$. As the weight vector w and the bias b can be rescaled by a positive factor without changing the actual hyperplane, we can introduce a normalized variant, the *geometric margin*.

Definition 2.24 (Geometric Margin)**.** The *(geometric) margin* of an example (x_i, y_i) with respect to any hyperplane (w, b) is the quantity:

$$\gamma_i = y_i \left(\left\langle \frac{1}{\|w\|} w, x_i \right\rangle + \frac{1}{\|w\|} b \right) .$$

Again, $\gamma_i > 0$ implies correct classification of (x_i, y_i). In analogy with Definition 2.22, the geometric margin γ of a whole example set $\mathcal{S}$ with respect to a hyperplane (w, b) is the minimum geometric margin of all items in the set.

The geometric margin is thus based on the canonical representation of a family of hyperplanes, where the weight vector is required to conform to $\|w\| = 1$. It thus measures the Euclidean distance of a training sample to the decision surface.

We will now investigate the relation between the margin and the VC dimension introduced in Definition 2.20. First note the following result on the VC dimension of general linear classification functions.

Proposition 2.6 (VC Dimension of Hyperplanes)**.** *Consider the class of classification functions in $\mathbb{R}^d$ whose decision boundaries take the form of hyperplanes. This set of functions has VC-dimension $h = d + 1$.*

Together with the result of Proposition 2.4, i.e. the bound on the expected error, a reduction of the VC dimension thus seems to require a reduction of the number of available features. However, this strategy would require complicated feature selection techniques that might also introduce another source of bias. Also, this approach seems contradictory to the use of kernels which implicitly introduce feature spaces of typically higher and, as we will see, possibly even infinite dimensionality.

Let us for a moment assume that we are dealing with a linearly separable dataset, i.e. a dataset for which we can achieve a positive margin. As a crucial observation, Vapnik and Chervonenkis (1974) have shown that, for the case of linear classifiers, the VC dimension can additionally be bounded in terms of the (geometric) margin of the dataset. Although the precise reasoning requires a number of additional results, we can get an intuition on how structural risk minimization can be implemented.

Proposition 2.7 (γ-Margin Separating Hyperplanes (Vapnik and Chervonenkis, 1974))**.** *Let a set $\mathcal{S} \subset \mathbb{R}^d$ of data items belong to a sphere of radius R. Then the set of linear discriminant functions that achieve a margin of at least γ (the set of γ-*margin separating hyperplanes*) has VC dimension h bounded by the inequality*

$$h \leq \min\left(\frac{R^2}{\gamma^2}, d\right) + 1 .$$

The interested reader is pointed to the original analysis by Vapnik (1995) and to Chapter 4 in the book by Cristianini and Shawe-Taylor (2000) for an elaborated discussion of these bounds and for extensions. This result together with the bound on the expected error suggest that the expected error of a separating hyperplane can be close to the training error, even in high-dimensional spaces, if it achieves a large margin. Thus, by maximizing the margin on the training data, we can hope to control the VC dimension and, consequently, the second error term. This is exactly the principle implemented by SVMs.

Hard-Margin Support Vector Machines

Let us further stick to the assumption that we are dealing with a linearly separable dataset. According to our earlier reasoning, the main principle behind SVMs is now to find the optimal separating hyperplane by maximizing the geometric margin. This situation seems intuitive: given no further information, we can assume that a hyperplane that achieves a large margin on a particular dataset is preferable to another one that separates the data only by a small margin. One the one hand, this provides a *unique* solution to the problem of choosing one out of many feasible decision surfaces. On the other hand, as we have seen, it leads to a better generalization behavior as the VC dimension can be bounded by larger margins. Formally, the situation can be expressed in the optimization problem:

$$\begin{aligned} &\max_{w,b,\|w\|=1} \gamma \\ &\quad\text{s.t. } y_i(\langle x_i, w\rangle + b) \geq \gamma,\ i = 1,\dots,n. \end{aligned} \tag{2.5}$$

The conditions ensure that all points are at least a signed distance γ away from the decision boundary and we seek the largest such γ with associated parameters. For the sake of motivation, we have used the notion of the geometric margin by restricting the weight vectors to conform to $\|w\| = 1$. Equivalently, we can replace this constraint by replacing the conditions with:

$$\begin{aligned} \frac{1}{\|w\|} y_i\left(\langle x_i, w\rangle + b\right) &\geq \gamma \\ y_i(\langle x_i, w\rangle + b) &\geq \gamma \|w\| . \end{aligned} \tag{2.6}$$

For any (w, b) that satisfy these constraints, their positively scaled multiples satisfy them, too, so we can arbitrarily set $\|w\| = 1/\gamma$, yielding:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 \\ &\text{s.t. } y_i(\langle w, x_i\rangle + b) \geq 1,\ i = 1,\dots,n. \end{aligned} \tag{2.7}$$

Keeping a large margin is thus equivalent to minimizing the norm of the weight vector while keeping outputs above a fixed value. The resulting margin will be $1/\|w\|$. The overall situation is illustrated on the left side of Figure 2.3.

Equation (2.7) spells out a quadratic criterion with linear inequality constraints and thus constitutes a convex optimization problem. Introducing a vector α of Lagrange multipliers $\alpha_i \geq 0,\ i = 1,\dots,n$, one for each of the constraints in Equation (2.7), we get the following (primal) Lagrange function:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(\langle x_i, w\rangle + b) - 1). \tag{2.8}$$

To minimize this expression, we need to set the derivatives to zero, yielding the following relations:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } \sum_{i=1}^{n} \alpha_i y_i x_i = w. \tag{2.9}$$

By substituting expressions (2.9) back into (2.8) we get the dual quadratic optimization problem.

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle, \\ &\text{s.t. } \alpha_i \geq 0,\ i = 1, \ldots, n, \\ &\sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{2.10}$$

On the one hand, this formulation is simpler to solve for standard convex optimization software. On the other hand, any references to the data again appear only from within inner products. On the one hand, we can reformulate the weight vector in Expression (2.9) in terms of training data items and can thus evaluate the final classification function in its dual version in the same way as done in the case of the perceptron training results. On the other hand, the optimization problem itself is expressed in terms of the coefficients α_i. This again allows us to replace the inner products $\langle x_i, x_j \rangle$ by any valid kernel function $\kappa(x_i, x_j)$ as in the case of the perceptron training.

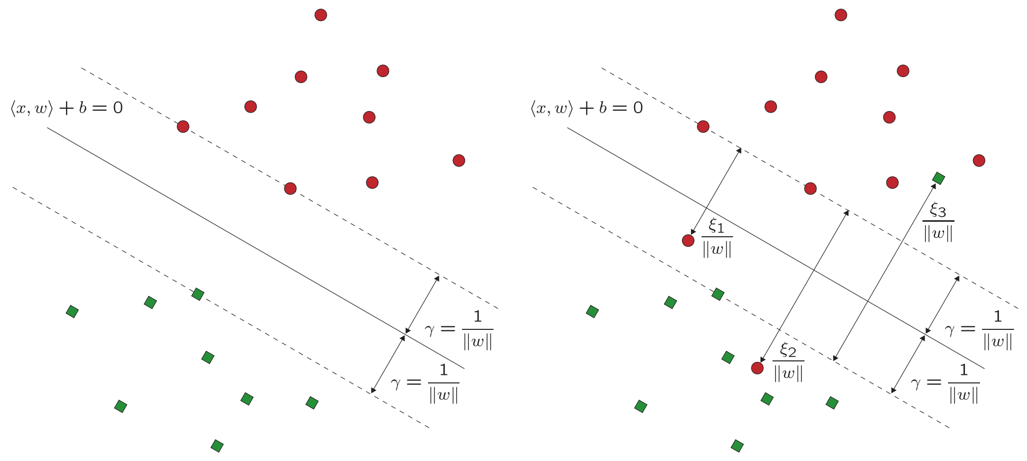
Before we move on to the soft-margin variant of the SVM note another property of the solution. The solution needs to conform to the Karush-Kuhn-Tucker conditions (Kuhn and Tucker, 1950; Karush, 1939), which, in addition to Equations (2.10) and (2.9) require that:

$$\alpha_i(y_i(\langle x_i, w \rangle + b) - 1) = 0, i = 1, \ldots, n. \tag{2.11}$$

From this last condition, the so called Karush-Kuhn-Tucker complementarity constraint, we can see that if $\alpha_i > 0$, then $y_i(\langle x_i, w \rangle + b) = 1$ and, conversely, if $y_i(\langle x_i, w \rangle + b) > 1$, then $\alpha_i = 0$. From this reasoning we see that the solution is not only unique and optimal but also *sparse* as the training points x_i that contribute to the solution vector w via $\alpha_i > 0$ are all placed on the margin, i.e. are the points closest to the hyperplane. These active points are called the *support vectors* of the solution.[6]

As another consequence of the complementarity constraint note that, while w can be determined by the optimization, the threshold b is not directly determined. However, it is given implicitly and can be found by choosing some i with $\alpha_i \neq 0$, inserting w into the corresponding Equation (2.11), and solving the resulting expression. In

[6]Note that any non support vector can be removed without altering the solution. The structure of the solution thus limits the number of possible leave–one–out errors which are upper bounded by the number of support vectors (Vapnik, 1995).



$\langle x, w\rangle + b = 0$
$\gamma = \frac{1}{\|w\|}$
$\gamma = \frac{1}{\|w\|}$
$\langle x, w\rangle + b = 0$
$\frac{\xi_1}{\|w\|}$
$\frac{\xi_3}{\|w\|}$
$\frac{\xi_2}{\|w\|}$
$\gamma = \frac{1}{\|w\|}$
$\gamma = \frac{1}{\|w\|}$

Figure 2.3 — Separating Hyperplanes as produced by hard-margin and soft-margin SVMs. The left illustration shows the separable case and the solution a hard-margin SVM might produce. The separating hyperplane is a solid line, the equidistant broken lines indicate the margins at distance $1/\|w\|$. The right illustration shows the non-separable case and the solution produced by a soft-margin SVM. Some points are on the wrong side of their corresponding margin by the amount $\xi_i/\|w\|$.

practice, the mean over all such solutions is usually preferred to avoid numerical instabilities.

Soft-Margin Support Vector Machines

Up to now, we have only considered the case of separable training data. In reality, given noisy data, we may not be able to find a linear decision boundary at all or, by enforcing a linear separation, may still face overfitting effects. So in reality, a good tradeoff between the empirical risk and the capacity control via the maximum margin approach needs to be set.

One way to deal with this situation is to still maximize the margin γ, but allow a certain fraction of training data points to violate the margin constraint. To achieve this situation, we rephrase the optimization problem of Equation (2.7) by introducing a vector ξ of *slack variables* that relax the hard margin constraints.

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t. } y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i,\ i = 1, \dots, n, \\ &\qquad \xi_i \geq 0. \end{aligned} \tag{2.12}$$

The first constraint allows a data point x_i to violate the hard margin constraint by a fraction of ξ_i. The right side of Figure 2.3 illustrates the situation. Naturally, we want to control the amount of this deviation, which is achieved by including the sum $\sum_{i=1}^{n} \xi_i$ as a term in the criterion function.[7] The *regularization constant* C controls the trade-off between the empirical error and the complexity term. Again, the optimization problem can be recast into the dual problem. We omit the full derivation using Lagrange multipliers and just state the final (dual) optimization problem:

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle, \\ &\text{s.t. } C \geq \alpha_i \geq 0, \; i = 1, \ldots, n, \\ &\sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{2.13}$$

An intriguing observation is that the reference to the slack variables can be dropped during the reformulation and do not occur in the dual optimization problem. The change with respect to the hard-margin problem only affects the range of admissible values of the Lagrange multipliers α_i which are now bounded from above by the tradeoff parameter C, a situation typically referred to as the *box constraint*. Again, note that the references to the data appear within the dot product only and can thus be replaced by any valid kernel function.

Again, the Karush-Kuhn-Tucker yield additional constraints that characterize the support vectors. Differing from the hard margin case, the support vectors now fall into two classes. On the one hand, support vectors with corresponding $\alpha_i < C$ yield a margin of exactly one and thus again lie directly on the margin and thus naturally do not cause any misclassification, such that $\xi_i = 0$. On the other hand, some support vectors correspond to Lagrange multipliers $\alpha = C$ and yield margins $y_i(\langle x_i, w \rangle + b) \leq 1$, i.e. they may lie on the wrong side of the margin (though, not necessarily on the wrong side of the hyperplane). Correspondingly, their deviation from the desired margin is quantified by slack variables $\xi_i \geq 0$.

Various alternative formulations of the basic SVM formulation exist, such as the prominent ν-SVM formulation by Schölkopf et al. (2000). These extensions are too numerous to be presented here in detail. Generally, there is a tight correspondence between these techniques and statistical theory.[8] The interested reader is referred to

[7]The sum corresponds to the 1-norm of the slack variable vector, i.e. $||\xi||_1$, which is the most commonly used version. An alternative formulation is the 2-norm soft margin SVM, where the error term is changed to $||\xi||_2$, i.e. using quadratic slack variables in the criterion function. The 2-norm formulation naturally leads to a slightly different optimization problem.

[8]In statistics, *regularization* is a general method for solving ill-conditioned parameter-estimation problems and for preventing overfitting of data by a model by minimizing a general risk functional of

the comprehensive books by Cristianini and Shawe-Taylor (2000); Shawe-Taylor and Cristianini (2004) as well as Schölkopf and Smola (2002) and the references therein.

2.3.3 Kernel-Based Nearest Neighbour

Nearest Neighbour-type algorithms are a set of simple but powerful supervised learning techniques (Mitchell, 1997, Chapter 8). These algorithms, in their basic form first reported by Cover and Hart (1967), use the training instances closest to the input item x under investigation to predict the corresponding target. To determine the "nearest" data points, this family of algorithms requires the definition of a (pseudo-)metric on the input space. As every inner product space is also a metric space, these algorithms can use kernels in a natural way.

Definition 2.25 (Kernel-induced Distance)**.** Given a valid kernel function $\kappa(\cdot,\cdot)$, the distance $d(\cdot,\cdot)$ between two points is defined as:

$$d(x,z) = \sqrt{\kappa(x,x) - 2\kappa(x,z) + \kappa(z,z)}\,.$$

The k-Nearest Neigbhour (kNN) technique can be adapted to deal with regression and classification problems. In both cases, upon presentation of a new data instance x, the k nearest data points are inspected. In the case of a regression setting, the target value is averaged over these k data items. For classification, the majority class of the k nearest training instances determines the class label prediction.

The critical parameter for nearest neighbour algorithms is the smoothing parameter k. High values of k lead to better smoothing and make the algorithm more stable with respect to random noise in the data. Smaller values of k lead to good approximations of complex patterns but are likely to lead to overfitting effects. A common variant, the smoothing technique, is to downweight the contributions of the k neighbours in proportion to their distance from the test example.[9] In contrast to the cases

the form: $R_{reg}[f] = R_{emp}[f] + C\,\Omega[f]$ where the *regularizer* $\Omega[f]$ penalizes the form of the hypothesis function in terms of the number of parameters and the parameter C determines the trade-off with respect to the empirical risk. The *Representer Theorem* (Kimeldorf and Wahba, 1971) gives an important characterization of the solutions to risk functionals of the type $R_{reg}[f] = R_{emp}[f] + C\Omega[\|f\|^2_{\mathcal{H}}]$ namely that the minimizer in a Reproducing Kernel Hilbert Space (RKHS) always admits a dual representation. In the case of linear classifiers this amounts to solutions of the form of Equation (2.3). The significance of this theorem lies in the fact that it applies to other types of learning and estimation procedures than SVMs, i.e. that the solutions of a large class of learning techniques can be expressed in dual form and thus admit the use of kernels (Hastie et al., 2001).

[9] Interestingly, in statistics, the term *kernel* is often used to refer to such weighting function used in non-parametric estimation techniques such as the nearest neighbour technique. While the term is not used exactly in the sense as we use it throughout this thesis (i.e. to refer to a specific choice of an inner product) there are strong connections between both types of kernels. Refer to the exposition by Hastie et al. (2001) for details on the statistical perspective on the matter.

of linear classification we have seen before, nearest neighbour algorithms do not rely on any assumptions about the pattern to be found. In the terminology of the bias-variance analysis, kNN techniques tend to show high variance but low bias (Hastie et al., 2001).

2.3.4 Kernel K-Means

The k-means algorithm (MacQueen, 1967) is a popular partitional clustering algorithm. The algorithm is related to the expectation maximization (EM) algorithm for estimating Gaussian mixture models as both algorithms try to detect the centers of natural clusters in the data. Specifically, given a set $\mathcal{S}$ of objects to be clustered and a number k of desired clusters, the k-means algorithm seeks to discover a partition $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of $\mathcal{S}$ that minimizes the total intra-cluster variance:

$$\begin{aligned} W(\mathcal{C}) &= \frac{1}{2} \sum_{m=1}^{k} \sum_{x_i, x_j \in \mathcal{C}_m} d(x_i, x_j) \\ &= \sum_{m=1}^{k} |\mathcal{C}_m| \sum_{x_i \in \mathcal{C}_m} d(x_i, \mu_m) \end{aligned} \tag{2.14}$$

whereby μ_m is the centroid or mean point of all the points $x_i \in \mathcal{C}_m$.

The k-means algorithm uses the following iterative procedure to identify this clustering. It starts by partitioning the data into k initial sets according to a random clustering. It then repeats the following two steps. In the first step, it calculates the mean, or centroid, of each cluster. In the second step, it constructs a new partition by assigning each instance to the cluster with the closest centroid. The algorithm repeats by alternating between these two steps. The algorithm has converged when none of the instances switches clusters anymore (or, alternatively, if centroids are no longer changed). Note that each of the two steps reduces the value of the optimization criterion in Equation (2.14) so that the algorithm converges. Note, however, that the result is not guaranteed to constitute a global optimum. The quality of the final solution depends largely on the initial set of clusters, and may, in practice, be significantly worse than the global optimum. As the algorithm is extremely fast, a common method is to run the algorithm several times and return the best clustering found.

Various alternatives of the basic algorithm exist, varying the basic scheme in terms of the initial clustering or the assignment of the clusters in the second step. Jain et al. (1999) give a good overview over these variants as well as other clustering algorithms.

If the data is not naturally clustered into spherical clusters in the input space, the clustering may, however, produce strange results. As with other learning problems, an implicit transformation into a more appropriate feature space by means of a kernel function may thus increase the cluster quality. Similar to the nearest neighbor technique investigated in the last section, the k-means clustering technique can make use of kernel-induced distance functions. However, the algorithm also requires a representation of the cluster centroids. Similar to the linear classifiers investigated above, these can be represented in dual form as the centroids can implicitly be represented by the set of instances in a cluster. The only computation that involves the centroid is the computation of its distance to a data item of interest during step two of the algorithm:

$$\begin{aligned} d(x, \mu_m) &= d\left(x, \frac{1}{|\mathcal{C}_m|} \sum_{x_i \in \mathcal{C}_m} x_i\right) \\ &= \sqrt{\langle x, x\rangle - 2\left\langle x, \frac{1}{|\mathcal{C}_m|} \sum_{x_i \in \mathcal{C}_m} x_i \right\rangle + \left\langle \frac{1}{|\mathcal{C}_m|} \sum_{x_i \in \mathcal{C}_m} x_i, \frac{1}{|\mathcal{C}_m|} \sum_{x_i \in \mathcal{C}_m} x_i \right\rangle} \\ &= \sqrt{\langle x, x\rangle - 2\frac{1}{|\mathcal{C}_m|} \sum_{x_i \in \mathcal{C}_m} \langle x, x_i\rangle + \frac{1}{|\mathcal{C}_m|^2} \sum_{x_i, x_j \in \mathcal{C}_m} \langle x_i, x_j\rangle} \end{aligned} \tag{2.15}$$

Again, the references to the data in Equation (2.15) happen solely from within inner products which can thus be replaced by any valid kernel function. An explicit computation of the cluster centroids in the feature space is not necessary.

2.4 Characterization of Kernels

In Section 2.3.1, we have motivated the use of kernel functions as an inner product in a so-called feature space. Recall from Definition 2.23 that a kernel function is a symmetric function that implicitly computes the dot product of the two arguments after a mapping ϕ to some inner product space, i.e. $\kappa(x, z) = \langle \phi(x), \phi(z)\rangle$. We have then shown how several learning algorithms can make use of the *kernel trick* by reformulating their computations in a dual version, where the data instances appear only within inner products. In particular, this procedure can be applied to design kernel functions on arbitrary types of data which are not transformed into a vector representation in the first hand, as long as the validity of the kernel function can be ensured.

In this section, we will undertake a more formal analysis of the properties of kernel functions. So far, there is only one way to define a valid kernel function: we need to explicitly think about a useful feature space, work out the corresponding mapping and then try to rewrite the resulting inner product in a way such that its computation becomes more attractive. While it often makes sense to illustrate the inner workings by means of such an analysis, it is equally often cumbersome and does not provide more information than the kernel function itself.

In this section, we review a number of properties of valid kernels, in particular their correspondence to positive semi-definite functions which constitutes a criterion of verifying the suitability of an arbitrary similarity function as a kernel. We also discuss some useful closure properties that can aid the construction of complex kernels out of existing kernels.

2.4.1 Positive Semi-Definite Kernels

In Definition 2.15, we have introduced positive semi-definite matrices as symmetric matrices with non-negative eigenvalues. Equivalently, Proposition 2.3 has shown that the condition $x'\mathbf{M}x \geq 0, \forall x \in \mathcal{X}, x \neq \mathbf{0}$ can be used to show that a matrix $\mathbf{M}$ is positive semi-definite. We will now generalize this concept by characterising functions as positive semi-definite.

Definition 2.26 (Positive Semi-definite Function)**.** Given a set $\mathcal{X}$ and a function $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, then κ is a *positive semi-definite function*, if it is symmetric and if for all $n \in \mathbb{N}$ and $x_1, \ldots, x_n \in \mathcal{X}$ the matrix

$$\mathbf{K} := (\kappa(x_i, x_j))_{ij},\ i, j = 1 \ldots n$$

is positive semi-definite.

Remark 2.6. If we are dealing with kernel functions κ, we will also call the matrix $\mathbf{K}$ obtained on some set $\mathcal{S} = \{x_1, \ldots, x_n\}$ the *kernel matrix* of $\mathcal{S}$ or, in analogy with Definition 2.11, its *Gram matrix*.

Proposition 2.8 (Cauchy-Schwarz Inequality for Positive Semi-definite Functions)**.** *If κ is a positive semi-definite function, and $x_1, x_2 \in \mathcal{X}$, then $\kappa(x_1, x_2)^2 \leq \kappa(x_1, x_1)\, \kappa(x_2, x_2)$.*

Proof. The 2×2 matrix with entries $\mathbf{K} := (\kappa(x_i, x_j))_{ij},\ i, j \in \{1, 2\}$ is positive semi-definite. Thus, both its eigenvalues must be non-negative. But then, $\mathbf{K}$'s determinant must also be non-negative:

$$0 \leq \mathbf{K}_{11}\mathbf{K}_{22} - \mathbf{K}_{12}\mathbf{K}_{21} = \mathbf{K}_{11}\mathbf{K}_{22} - \mathbf{K}_{12}^2. \qquad (2.16)$$

This shows the desired inequality. □

The important property of positive semi-definite functions is that they exactly describe the set of valid kernel functions.

Proposition 2.9. *Given a set $\mathcal{X}$ and a function $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ which is either continuous or has a finite domain. Then κ can be decomposed*

$$\kappa(x,z) = \langle \phi(x), \phi(z) \rangle$$

into a feature map $\phi : \mathcal{X} \mapsto \mathcal{H}$ into a Hilbert space $\mathcal{H}$ applied to both its arguments followed by the evaluation of the inner product in $\mathcal{H}$ if and only if it is a positive semi-definite function.

Proof. We first show that the inner product is a positive semi-definite function. Consider the general case of a Gram matrix

$$\mathbf{K} := (\kappa(x_i, x_j))_{ij} = (\langle \phi(x_i), \phi(x_j) \rangle)_{ij}, \ x_i, x_j \in \mathcal{X}. \tag{2.17}$$

For any compatible vector v we have

$$\begin{aligned} v'\mathbf{K}v &= \sum_{i,j=1}^{n} v_i v_j \mathbf{K}_{ij} = \sum_{i,j=1}^{n} v_i v_j \langle \phi(x_i), \phi(x_j) \rangle \\ &= \left\langle \sum_{i=1}^{n} v_i \phi(x_i), \sum_{j=1}^{n} v_j \phi(x_j) \right\rangle \\ &= \left\| \sum_{i=1}^{n} v_i \phi(x_i) \right\|^2 \geq 0, \end{aligned} \tag{2.18}$$

i.e. any such $\mathbf{K}$ is positive semi-definite as required.

We now show that the converse implication is also valid. Assuming that κ is indeed a positive semi-definite function, we define a mapping $\phi : \mathcal{X} \mapsto \mathcal{H}_\kappa$ into a Hilbert space $\mathcal{H}_\kappa$ of *functions* which map $\mathcal{X}$ into $\mathbb{R}$ as follows:

$$\phi(x) = \kappa(\cdot, x). \tag{2.19}$$

Note that the elements of $\mathcal{H}_\kappa$ can equally well be seen as functions or as elements of $\mathcal{X}$. To make $\mathcal{H}_\kappa$ a Hilbert space we first turn it into a vector space by the obvious definition of scalar multiplication and addition defined by

$$(f+g)(x) = f(x) + g(x), \ f, g \in \mathcal{H}_\kappa \tag{2.20}$$

Let $f(\cdot)$ and $g(\cdot)$ be arbitrary functions of the form

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i \kappa(\cdot, x_i) \ \text{ and } \ g(\cdot) = \sum_{j=1}^{m} \beta_j \kappa(\cdot, z_j), \tag{2.21}$$

with arbitrary $n, m \in \mathbb{N}, \alpha_i, \beta_j \in \mathbb{R}$ and $x_i, z_j \in \mathcal{X}$. We then define an inner product on $\mathcal{H}_\kappa$ as

$$\begin{aligned} \langle f, g \rangle &= \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_i \kappa(x_i, z_j) && (2.22) \\ &= \sum_{i=1}^{n} \alpha_i g(x_i) = \sum_{j=1}^{m} \beta_j f(z_j). && (2.23) \end{aligned}$$

From this definition, it is obvious, that $\langle \cdot, \cdot \rangle$ is real-valued, symmetric and bilinear. Furthermore,

$$\langle f, f \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \kappa(x_i, x_j) \geq 0,$$

which follows from the assumption that κ is positive semi-definite. It remains to be shown that $\langle f, f \rangle = 0 \Rightarrow f = \mathbf{0}$. First note that by Equation (2.22) we have for all functions f:

$$\langle \kappa(\cdot, x), f \rangle = f(x), \text{ in particular } \langle \kappa(\cdot, x), \kappa(\cdot, z) \rangle = \kappa(x, z). \quad (2.24)$$

Then, due to the Cauchy-Schwartz inequality in the variant of Proposition 2.8 we have:

$$|f(x)|^2 = |\langle \kappa(\cdot, x), f \rangle|^2 \leq \kappa(x, x) \langle f, f \rangle. \quad (2.25)$$

By this inequality, $\langle f, f \rangle = 0$ implies $f = 0$, such that $\langle \cdot, \cdot \rangle$ is indeed a valid inner product on $\mathcal{H}_\kappa$. Furthermore, Equation (2.24) shows that its value is indeed equivalent to the initial kernel function.

It remains to be shown that $\mathcal{H}_\kappa$ is complete to make it a Hilbert space. Consider a fixed input x and a Cauchy sequence of elements in $\mathcal{H}_\kappa$ $(f_n)_{n=1}^{\infty}$. We now have, by virtue of the Cauchy-Schwarz inequality, that

$$(f_n(x) - f_m(x))^2 = \langle f_n - f_m, \kappa(x, \cdot) \rangle^2 \leq \|f_n - f_m\|^2 \kappa(x, x). \quad (2.26)$$

The sequence $f_n(x)$ is thus a bounded Cauchy sequence of real numbers and hence has a limit. We can then complete the space $\mathcal{H}_\kappa$ by including in it all the functions of the form

$$g(x) = \lim_{n \to \infty} f_n(x). \quad (2.27)$$

The space $\mathcal{H}_\kappa$ is thus a complete inner product space, i.e. it is indeed a Hilbert space. □

The property of Equation (2.24) is known as the *reproducing property* of the kernel κ (Aronszajn, 1950). Correspondingly, $\mathcal{H}_\kappa$ is also called its *Reproducing Kernel Hilbert*

Space (RKHS). In literature, especially in the early publications on SVMs, we often encounter an alternative treatment of the correspondence of positive semi-definite functions and kernel functions from the perspective of integral operator theory mainly due to Mercer (1909). In this context, the kernel functions are also often explicitly referred to as *Mercer kernels*. We here abstain from a detailed exposition on this view and point to the good treatment of this topic by Cristianini and Shawe-Taylor (2000) in the context of kernels. Bhatia (2007) provides additional insights into the theory of positive (semi-) definite functions.

2.4.2 Kernel Construction

So far, we have investigated the theoretical properties of kernel functions. We have seen that it is possible to either construct kernel functions explicitly by means of a dot product formulation or to check a given similarity function whether it is positive semi-definite.

Kernel Closure Properties

The given characterization of kernels also allows us to specify a set of rules that allow the construction of valid kernels by combining existing kernel functions for which the kernel property has already been shown. These closure properties can be used conveniently to create advanced kernels out of simpler kernels.

Proposition 2.10 (Closure under Sum and Rescaling). *Given two kernels $\kappa_1(\cdot,\cdot)$ and $\kappa_2(\cdot,\cdot)$ defined on $\mathcal{X}$, and coefficients $\alpha_1, \alpha_2 \in \mathbb{R}+$, then the function $\kappa_3(x,z) = \alpha_1\kappa_1(x,z) + \alpha_2\kappa_2(x,z)$ also constitutes a valid kernel.*

Proof. Let $\mathbf{K}^1, \mathbf{K}^2$ and $\mathbf{K}^3$ denote arbitrary kernel matrices of $\kappa_1(\cdot,\cdot), \kappa_2(\cdot,\cdot)$ and $\kappa_3(\cdot,\cdot)$, and let $x \in \mathbb{R}^d$ be any compatible vector. We then have that:

$$x'\mathbf{K}^3x = x'(\alpha_1\mathbf{K}^1 + \alpha_2\mathbf{K}^2)x = \alpha_1(x'\mathbf{K}^1x) + \alpha_2(x'\mathbf{K}^2x) \geq 0\,,$$

i.e. $\kappa_3(\cdot,\cdot)$ is positive semi-definite. □

Proposition 2.11 (Closure under Direct Product). *Given two kernels $\kappa_1(\cdot,\cdot)$ and $\kappa_2(\cdot,\cdot)$ defined on $\mathcal{X}$, then the function $\kappa_3(x,z) = \kappa_1(x,z)\,\kappa_2(x,z)$ also constitutes a valid kernel. This type of operation is also often referred to as the Schur product.*

We shortly sketch the proof by Shawe-Taylor and Cristianini (2004). Let $\mathbf{K} = \mathbf{K}^1 \otimes \mathbf{K}^2$ be the *tensor product* of the kernel matrices. The tensor product is obtained by replacing each entry of $\mathbf{K}^1$ by $\mathbf{K}^2$ multiplied by that entry. The tensor product of two positive semi-definite matrices yields a positive semi-definite matrix. The matrix $\mathbf{K}^3$ obtained by the pointwise product of Definition 2.11 is a principal submatrix of $\mathbf{K}$.

A principle submatrix of a positive semi-definite matrix is, however, itself positive semi-definite (Shawe-Taylor and Cristianini, 2004).

Proposition 2.12 (Completeness)**.** *Given a sequence of kernels* $\kappa_1, \kappa_2, \ldots$*, such that the sequence converges, then the limit* $\lim_{n\to\infty} \kappa_n(x, z)$ *also constitutes a valid kernel.*

Proof. Let $\mathbf{K}_n$ denote arbitrary kernel matrices of $\kappa_n(\cdot, \cdot)$, and let $x \in \mathbb{R}^d$ be any compatible vector. We then have that:

$$x' \left(\lim_{n\to\infty} \mathbf{K} \right) x = \lim_{n\to\infty} x'\mathbf{K}x.$$

□

Note that the last proposition, together with Proposition 2.10 characterizes the set of positive semi-definite kernels as a closed convex cone. Of course, all the properties presented in this section remain valid if the kernels are defined on different domains.

Kernel Modifiers

In this section, we consider a number of functions that can be applied to the result of a kernel calculation while retaining the kernel property. Well-known kernel modifiers for a valid kernel $k(x, y)$ on some input set $x, y \in \mathcal{X}$ are, among others, the normalisation kernel, the polynomial kernel, and the Gaussian kernel. Note that most of these *kernel modifiers* were initially introduced as kernels on vector arguments where the embedded kernel was simply the standard dot product.

Proposition 2.13 (Cosine Normalization Kernel Modifier)**.** *Given a kernel function* $\kappa(\cdot, \cdot)$*, the function*

$$\kappa_{norm}(x, z) = \frac{\kappa(x, z)}{\sqrt{\kappa(x, x)\, \kappa(z, z)}}$$

constitutes a valid kernel.

Note that, as $\sqrt{\kappa x, x}$ corresponds to the Euclidean norm $\|x\|$ in the feature space, the kernel modifier is analogous with the cosine normalization for vectors which scales the kernel results to $[0, 1]$ (compare Definition 2.9).

Another, commonly used kernel modifier is the *polynomial kernel.*

Proposition 2.14 (Polynomial Kernel Modifier)**.** *Given a kernel function* $\kappa(\cdot, \cdot)$ *and parameters* $p \in \mathbb{N}$ *and* $c \geq 0$ *the function* $\kappa_{poly}(x, z) = (\kappa(x, z) + c)^p$ *constitutes a valid kernel.*

The validity of the polynomial kernel modifier directly follows from Propositions 2.10 and 2.11. Note that the kernel introduced in Example 2.8 is an example of this kernel modifier in conjunction with the plain dot product.

Proposition 2.15 (Gaussian Kernel Modifier). *Given a kernel function $\kappa(\cdot,\cdot)$ and a parameter $\sigma \in \mathbb{R}^+$ the function*

$$\kappa_{gaussian}(x,z) = \frac{\exp(-\kappa(x,x) - 2\kappa(x,y) + \kappa(y,y))}{\sigma^2} = \frac{\exp(-\|x-z\|_{\mathcal{H}})}{\sigma^2}$$

constitutes a valid kernel.

In this formulation, we have replaced the complex enumerator of the fraction with the shorter formulation as distance, whereby $\|\cdot\|_{\mathcal{H}}$ denotes the norm in the kernel-induced feature space.

Proof. We can decompose the Gaussian kernel modifier as follows:

$$\begin{aligned}\kappa_{gaussian}(x,z) &= \frac{\exp(-\|x-z\|_{\mathcal{H}})}{\sigma^2} \\ &= \exp(-\|x\|^2_{\mathcal{H}}/\sigma^2)\exp(-\|z\|^2_{\mathcal{H}}/\sigma^2)\exp(2\kappa(x,z)/\sigma^2)\,. \qquad (2.28)\end{aligned}$$

The first two factors can be seen as the product of two real-valued mappings of the inputs which thus constitutes a kernel. For the last factor note that an exponential function can be arbitrary closely approximated by polynomials with positive coefficients and thus is a limit of kernels. Together with Proposition 2.12, the result follows. □

The Gaussian kernel forms a specific case of the larger class of Radial Basis Function (RBF) kernels which are general functions over the (potentially kernel-induced) distance of two data items. It is of particular interest due to its peculiar properties and also helps to illustrate the abstract notion of Reproducing Kernel Hilbert Spaces which we have encountered earlier. Recall that under the RKHS interpretation, we can always construct a mapping of data items into a space of functions where each data item is represented by a kernel-shaped function referring to the respective pattern. In this sense, a data item is represented by its similarity to all other items. For the case of Gaussian kernels this mapping is given by:

$$\phi : \mathcal{X} \mapsto \mathcal{H} \text{ with } \phi(x) = \kappa(x,\cdot) = \exp\left(\frac{\|x-\cdot\|^2}{\sigma^2}\right) \qquad (2.29)$$

The function in the RKHS space thus takes the form of a Gaussian bell function. Note that, in the feature space, all data items are normalized to unit length, i.e. they lie on the surface of a hyperball. The feature space itself thus has dimensionality that corresponds to the number of overall instances, thus potentially infinite. The bandwidth parameter σ controls much of the behaviour of the Gaussian kernel. Small values of σ yield a situation where all data items are almost orthogonal, leading to precise kernels which may, however, overfit more easily. On the contrary, larger values of σ lead to a situation where the data items are almost parallel and the kernel has little discriminative power.

2.4.3 Kernel Design

While the validity of a kernel is important for its correct behaviour, it does not say much about the power of the kernel in solving actual learning problems, i.e. its capability to detect *concept classes*. In binary classification settings, the concept class $c(\cdot)$ corresponds to a binary classification label, i.e. $c(x) \in \mathcal{Y} = \{+1, -1\}$.

Based on this reasoning, Gärtner (2003) has introduced formal notions for characterizing *kernel quality* which we shortly review in this section. He distinguishes the concepts of *kernel completeness*, *kernel correctness* and *kernel appropriateness* with respect to a dataset and learning task.

Kernel Completeness This notion characterizes the capability of the kernel to represent the learning problem. Formally, a kernel is called *complete* if $\kappa(x, \cdot) = \kappa(z, \cdot)$ implies $x = z$. With respect to the concept class it is, however, sufficient for a kernel to be complete with respect to a concept class, i.e. $\kappa(x, \cdot) = \kappa(z, \cdot)$ implies $c(x) = c(z) \, \forall \, c, z \in \mathcal{C}$.

Kernel Correctness This notion characterizes how much of the underlying hypothesis language is properly reflected in the kernel. Given the case of linear combinations of kernels on training items as the hypothesis language (e.g. as in the case of SVMs and the perceptron), we can call a kernel correct if for all concepts $c \in \mathcal{C}$ there exist $\alpha_i \in \mathbb{R}$, $x_i \in \mathcal{X}$, $b \in \mathbb{R}$ such that $\forall x \in \mathcal{X} : \sum_i \alpha_i \kappa(x_i, x) + b \geq 0 \Leftrightarrow c(x)$, i.e. that the learning problem can be solved at all by this kernel. Note that a kernel which is not complete can not be correct.

Kernel Appropriateness This notion characterizes the quality of the kernel with respect to the question whether examples that are "close" in class membership are also "close" in the feature space such that a good generalization is possible. This property can only be (experimentally) validated for a given class hypothesis functions and the employed learning algorithm. As a simple example consider the case of the matching kernel $\kappa_\delta(x, z) = 1 \Leftrightarrow x = z$ and $\kappa_\delta(x, z) = 0 \Leftrightarrow x \neq z$ which is complete and always correct though not usually not appropriate as it will not generalize well to unseen examples.

In essence, the former two notions relate to the ability of the kernel to separate hidden concepts well, while the latter relates to the kernel's generalization capabilities.

2.5 Performance Assessment

In this section, we shortly discuss the procedures for assessing the quality of estimated machine learning models. These procedures will also be used for assessing the results of the experiments reported in Sections 6 and 8.

2.5.1 Binary Classification

While the above procedure enables us to hopefully determine the optimal prediction function, the question arises how the eventual effectiveness of the classifier can be estimated and which evaluation measure should be used.

Estimation Procedures

The empirical error obtained on the training data is not a good estimate of the generalization error as it is biased towards the training sample and has been used to derive the prediction function in the first hand. The only way to get an idea of the size of the generalization error is to estimate it on a separate *test sample* that has been generated independently of the training sample and that has not been used for training. The approach of setting aside a certain subset of the available data is called a *hold out* approach.

Usually, the available data is limited and we want to use as much as possible for training to reduce the variance of the trained classifier. At the same time, the error estimate based on the test set suffers from the same problem. If the test sample is too small, the variance of the expected variance of the error estimate will be greater — so a tradeoff needs to be found. If enough data is available, the problem becomes less severe, but if data is limited other techniques have to be used. Often, a strategy called *cross validation* is applied. Here, all available data is partitioned into a certain number of folds. In multiple runs, classifiers are trained on all but one folds, each time leaving aside a different fold which is used as test set. Averaging over the results in each run, the error of a final classifier trained on the entire available data can be estimated. In the extreme case, where the number of folds is equal to the number of available training examples, we speak of a so-called *leave–one–out (LOO) estimate*.

Performance Metrics

All performance metrics can be estimated based on the observations on the test set. Given binary classification setting, four possible combinations of true and predicted class label can arise in total.

Definition 2.27. Given a classifier f trained on an independent training set one can partition the set into $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$ and further into the sets $\mathcal{S}^+ = TP \cup FN$ and $\mathcal{S}^- = FP \cup TN$ given by:

$$\begin{aligned}
TP &:= \{(x_i, y_i) \in \mathcal{S} \mid f(x_i) = 1 \wedge y_i = 1\} \\
FP &:= \{(x_i, y_i) \in \mathcal{S} \mid f(x_i) = 1 \wedge y_i = -1\} \\
TN &:= \{(x_i, y_i) \in \mathcal{S} \mid f(x_i) = -1 \wedge y_i = -1\} \\
FN &:= \{(x_i, y_i) \in \mathcal{S} \mid f(x_i) = -1 \wedge y_i = 1\}
\end{aligned}$$

called the sets of *true positive*, *false positive*, *true negative* and *false negative* classifications.

Based on this definition, different evaluation measures have been defined. Commonly used classification measures are the *classification error*, *precision*, *recall* and the F_β measure.

The most natural classification metric, the *classification error* measures the probability that the classifier will make a wrong decision which actually coincides with the notion of the empirical error in conjunction with the zero/one loss function.

Definition 2.28 (Classification Error)**.** The classification error of classifier f is estimated by:

$$\hat{R}(f, \mathcal{S}) := \frac{|FP| + |FN|}{|TP| + |FP| + |TN| + |FN|} \; .$$

Sometimes, the complementary measure of *accuracy* is used that measures the probability of the complementary event, namely that the classifier makes *right* decisions. Obviously, the accuracy is calculated by subtracting the error probability from one.

Although the classification error is an intuitive measure, it is not the used as the sole metric of interest. The reason is that in most settings the number of positive vs. negative test examples $|\mathcal{S}^+|$ and $|\mathcal{S}^-|$ are extremely uneven. Many real-world datasets have several thousands of instances of which only a small fraction carries a certain class label. In this setting, a trivial rejector, i.e. a simple classifier given by $f(x) = -1 \; \forall x \in \mathcal{S}$, that produces negative decisions only, may yield impressively low error rates, sometimes outperforming other classifiers. Comparing and tuning classifiers based on the error rate may therefore lead to adopting a classifier that behaves very much like the trivial rejector (Yang, 1999). As an alternative, the classification effectiveness is often measured in terms of the *precision* and *recall* metrics originating from information retrieval.

Definition 2.29 (Precision and Recall)**.** The precision and recall of classifier f are given by:

$$\begin{aligned} precision(f, \mathcal{S}) &:= \frac{|TP|}{|TP| + |FP|} \\ recall(f, \mathcal{S}) &:= \frac{|TP|}{|TP| + |FN|} \; . \end{aligned}$$

Precision thus estimates the probability that if the classifier judges an instance to belong to a certain class, this decision is correct. Precision therefore measures how "clean" the result for a certain class label is. Conversely, recall estimates the probability that if an instance actually belongs to the class in question, it will be detected as such by the classifier.

Unfortunately, neither precision nor recall are sensible measures of the classifiers performance if judged independently. Most often, though not always, precision and recall are inversely related, meaning that higher recall scores often come at the cost of worsening precision scores.

Example 2.9 (Pathological Classifiers)**.** Consider an extremely "conservative" binary classifier that judges only a single instance on a given test sample to belong to a certain class and that decision is correct. In this case, precision will obviously be 1.0, while recall may be quite low if more instances of this class actually linger around in the test sample. In contrast, a classifier that retrieves any instance as belonging to one class will achieve maximum recall at score 1.0, while precision may be quite low as other instances not belonging to that class have been retrieved as well.

As neither precision nor recall make sense in isolation it is desirable to evaluate a classifier based on a measure that combines both. A measure that is widely used for this purpose is the F_β function proposed by van Rijsbergen (1979).

Definition 2.30 (F_β measure)**.** Given precision and recall, the combined F_β measure of precision and recall is given by:

$$F_\beta(f,\mathcal{S}) = \frac{(\beta^2+1)\ \mathit{precision}(f,\mathcal{S}) \cdot \mathit{recall}(f,\mathcal{S})}{\beta^2\ \mathit{precision}(f,\mathcal{S}) + \mathit{recall}(f,\mathcal{S})} \ ,$$

where $\beta \in \mathbb{R}^+$ is a suitably chosen parameter.

The parameter β weights precision versus recall. Smaller values of β emphasize the contribution of recall, while higher values of β emphasize precision. It is common to assign equal weight to precision and recall by setting $\beta = 1.0$ in which case the formula simplifies to

$$F_1(f,\mathcal{S}) := \frac{2\ \mathit{precision}(f,\mathcal{S}) \cdot \mathit{recall}(f,\mathcal{S})}{\mathit{precision}(f,\mathcal{S}) + \mathit{recall}(f,\mathcal{S})} \ , \tag{2.30}$$

which is equivalent to the harmonic mean of precision and recall. Here, the smaller of precision and recall dominates the F_1 score, which is therefore maximized when precision and recall are equal or close to each other.

2.5.2 Multiclass Classification

We have so far only looked at the case of assessing the performance of binary classifiers where $\mathcal{Y} = \{+1, -1\}$. However, in many real world applications, we may encounter problems that require inputs to be mapped to a set $\mathcal{C}$ of available categories. While some supervised machine learning algorithms can directly be extended to work with this setting, many algorithms are intrinsically devised for binary classification and require more complex schemes to achieve this effect.

Multi-Label vs Single-Label Classification

In the simplest case, the task is to assign the inputs to *several* possible categories in $\mathcal{C}$. We will refer to this case as the *multi-class, multi-label setting*. The extension of a binary algorithm to this setting is commonly done by reducing the problem into $|\mathcal{C}|$ binary sub-problems. The subproblems are then trained, evaluated independently and finally processed into the calculation of the overall predictions.

Depending on the application context, additional constraints on the choice of target classes may be imposed. In a very common alternative setting, referred to as the *multi-class, single-label setting*, the number of classes to be assigned to a data instance is enforced to be exactly equal to one. The extension of a binary algorithm to this setting is not always possible or easy to conceive. Again, a common alternative consists in reducing the problem into several binary sub-problems and compose their individual decision into a joint classification. However, the problem is that the different binary classifiers may produce conflicting individual classifications that have to be resolved by the composite system. Several binary classifiers, such as the margin-based classifiers discussed in the next section, are capable of producing fine-grained confidence values on a numeric scale instead of pure binary decisions. In the case of such classifiers, the input instances are mapped to a real vector of confidence values formed by the outputs of the binary classifiers. The final target class is then computed from this vector by means of a decoding function. In the simplest case, the so called *one-vs-all* scheme, the class that achieves the maximal prediction confidence is considered positive, while the others are implicitly considered negative. More generic schemes for reducing the case to binary classification problems are the methods based on error correcting output codes (ECOCs) introduced by Dietterich and Bakiri (1995) and more recently extended by Allwein et al. (2000). In the case of SVMs, some approaches have directly extended the algorithm to the multi-class case. While the resulting models have the advantage of simultaneously optimizing all decisions, they are computationally very intensive.

In the sequel we will focus on binary classification only, which eases notation. The experiments presented in Parts III and IV, however, are often based on multi-class settings. According to the procedures described, the extension from binary classification is straightforward.

Micro and Macro Averaging of Performance Measures

The evaluation measures presented so far were defined for binary classifiers. However, in the case of multi-class, multi-label settings we are interested in the performance of a whole set of binary classifiers for a corresponding set of classes. In this case, the evaluation scores of the individual classifiers have to be averaged across class labels. All performance metrics we have encountered so far are of the general structural form of a fraction. Each performance metric q_i on some category $c_i \in C$ has the general form:

$$q_i = \frac{a_i}{b_i} \; . \tag{2.31}$$

Here, a_i refers to the expression in the numerator and b_i to the expression in the denominator of the respective performance metric.

Based on Expression (2.31), one can define the *micro-averaged* or *macro-averaged* scores as follows. Micro-averaged scores q_μ are obtained by averaging over all instances:

$$q_\mu = \frac{\sum_{i=1}^{|\mathcal{C}|} a_i}{\sum_{i=1}^{|\mathcal{S}|} b_i} \; . \tag{2.32}$$

The general interpretation of this averaging alternative is that micro-averaging corresponds to a per-instance average. The micro-averaged F_1 value is computed directly from micro-averaged precision and recall. In contrast to micro-averaging, the macro-averaged scores q_M are obtained by averaging over the different classes:

$$q_M = \frac{\sum_{i=1}^{|\mathcal{C}|} q_i}{|\mathcal{C}|} \; . \tag{2.33}$$

Here, the macro-averaged scores correspond to per-class averages. Note that the macro-averaged classification error equals the micro-averaged classification error.

The choice of the averaging scheme to be employed depends on the characteristics of class distribution in the test dataset. In macro averaging, all classes are treated equally, regardless of the numbers of positive instances in the different classes. If all classes are considered equally important, the macro-averaging scheme is usually the best choice but it is important to keep in mind that the individual figures on small classes can have a large effect. If the uneven distrubution of positive training documents is to be considered explicitly, the micro-averaging scheme is usually favoured although this bears the risk that classes with a grossly higher number of training documents can dominate the result. If classes are equally distributed, there is usually no particular reason to favour a particular averaging scheme.

The averages of the evaluation measures presented so far are most easily conceived for assessing the performance of multiple independent binary classifiers , i.e. multi-class, multi-label settings. In most of the experiments which we report on, we will adopt this view. For the case of a multiclass, single-label setting, i.e. a setting where the final result of multiple independent classifiers is post-processed by a selection function to enforce the assignment of a single category, these measures are used by some authors in analogy. In the case that the performance in an explicit multiclass, single-label setting, is of interest this thesis will however resort to explicitly reporting the *single-label adjusted error (accuracy)* as a measure of multiclass, single-label classification performance. These measures are not computed as an average over multiple categories but as the total fraction of wrong (correct) classification decisions.

2.5.3 Measuring Significance: the Paired T-Test

Besides simply reporting performance measures, statistical significance tests are useful in machine learning experiments to verify that an given change in performance is *significant*, i.e. that it is very unlikely that the change in question is only the result of random effects (Yang and Liu, 1999). This thesis will sometimes use a common siginficance test procedure devised by Gosset (1908) which is known as *(Student's) paired T-Test*

The test compares two systems based on the *paired* values of a the reference performance measure, e.g. F_1, in different experiments, e.g. on the same set of classes or training subsets which can be associated with one another among the different experiments. Specifically, given that two systems A and B were evaluated in m individual experiments, let a_i denote the performance value achieved by system A in the ith experiment and b_i the same for system B. Then, the values $d_i = a_i - b_i$ indicate the amount of deviation and d is a random variable with mean μ_d and unknown standard deviation σ_d. Let the sample average of the sequence of d_i for $i = 1 \dots n$ be denoted by $\overline{d}$ and the estimated variance by $\overline{\text{var}}(d)$.

The null hypothesis H_0 of the (one-sided) test is that $\mu_d = 0$, i.e. that on average there is no difference in the performance of the systems while the alternative hypothesis H_1 is $\mu_d > 0$, i.e. that system A yields higher performance values than the reference system B. The (one-tailed) p-value is then computed from the t-distribution with $n - 1$ degrees of freedom for the test statistic:

$$T = \frac{\overline{d}}{SE(\overline{d})} = \frac{\overline{d}}{\sqrt{\frac{\widetilde{var}(d)}{n}}}. \tag{2.34}$$

For large n (at least $n > 40$) the standard normal distribution approximates the t-distribution and may be used instead. The p-value measures the probability of committing a type I error, that is the probability of falsely rejecting H_0. Small p-values thus indicate substantial evidence against the null hypothesis. Following common statistical practice, the significance level $\alpha = 0.05$ is required for the claim that an improvement is *significant*. The significance level of $\alpha = 0.01$ is required for the claim that an improvement is *very significant*.

2.6 Summary

In this section, we have introduced the main building blocks of kernel-based learning. We have first reviewed the main mathematical concepts that underlie kernel-based machine learning. We have then introduced, step-by-step the machine learning setting both supervised and unsupervised. We have motivated the use of kernels in the context of the kernel perceptron and subsequently presented examples of other kernel-based learning techniques among which the SVM constitute the most important instantiation in practice. We have then, in more technical detail, investigated the concept of valid kernel functions and their characterization as positive semi-definite functions. Furthermore, we have looked at well known kernels and kernel closure properties and discussed the issue of appropriate kernel design. Finally, we have finally reviewed the techniques for assessing the performance of machine learning algorithms. In the next section, we will turn to the other main building block of this thesis, namely the concepts of knowledge representation and ontologies.

Chapter 3

Knowledge Structures

In this chapter, we introduce a number of basic concepts from the field of knowledge representation. Knowledge representation is a research area shaped by a variety of disciplines, theories and applications. As a branch of artificial intelligence, knowledge representation deals with mechanisms for representing factual knowledge as well as general ideas, associations, and perceptions in a formal manner for further processing by computer systems. Representing knowledge in an explicit form enables computer systems to automatically organize data records, draw conclusions, and answer queries about the available knowledge in a way similar to human reasoning (Russell and Norvig, 2003). Common to many of these perspectives is the notion of an ontology as a conceptual model of a particular domain of interest (see e.g. Staab and Studer (2004) or Grimm et al. (2007)). Several formalisms have been devised for knowledge representation which differ in the set of supported primitives and their formal grounding. In the remainder, we will use the term *ontology* to refer to knowledge expressed by means of a formalism that (i) supports at least a minimal set of core primitives, in particular the capability to organize knowledge by means of individuals and classes and (ii) are formally grounded in a logical theory. As a more general notion we will refer to *knowledge structures* as knowledge expressed by a potentially informal network structure.

This chapter is organized as follows. First, Section 3.1 reviews some basic notions related to the nature of knowledge representation. On the one hand, it tries to answer the question why computer science is occupied with knowledge representation in practical and scientific contexts. On the other hand, it shortly reviews how the topic of knowledge representation has typically been addressed and which basic concepts have been established in the field. Section 3.2 then presents a formal framework for knowledge structures that forms the basis for the subsequent exposition in this thesis. The framework is inspired by current Semantic Web standards, in particular the graph model of the Resource Description Framework (RDF) (Manola and Miller, 2004) and class-based description languages which form the basis of Description Logics (DLs) (Baader et al., 2003) and the related Web Ontology Language (OWL) (McGuinness and van Harmelen, 2004). However, the model is largely kept generic and is capable of characterizing a large set of views on the topic of knowledge representation. Throughout these sections we will sketch the relation to the relevant Semantic Web standards where appropriate. The chapter concludes with a short summary in Section 3.3.

3.1 Background

In recent years, there has been an increased interest in knowledge representation and ontologies (Staab and Studer, 2004). The reasons for the popularity of this topic are manifold, and we will begin with motivating this interest.

3.1.1 Motivation

We can roughly cluster the influences for knowledge representation in computer science as follows:

Classification of Resources Most notably in information organization and library science, we encounter a need to organize knowledge resources (e.g. files or library media) to facilitate browsing and retrieval. A common method is to organize resources in a formal classification structure. Such a structure, a *taxonomy*, is a system for coding and organizing similar resources according to their subject or other important characteristics. The classes of interest are usually arranged in a hierarchical manner which encodes a specialization/generalization relationship.

Lexical Resources In the study and formalization of natural language, *lexical taxonomies* aim at structuring the words and senses of a given language along different types (e.g. nouns, verbs etc.) and relating these via relations such as hypernymy (generalization), antonymy (opposition) or meronymy (part-relationship).

Metadata Descriptions Besides a classification into major classes as done with taxonomies, *metadata descriptions* allow for a detailed specification of the characteristics of an object (e.g. a document). Metadata descriptions correspond to records where an entity is described by a set of fixed attributes. Metadata systems are usually tightly coupled with a type system whereby the type of an entity determines the structure (schema) of its metadata record.

Information Integration Given related data sources stored according to differing metadata schemata, we often wish to enable a unified view on the data. We thus encounter a need to specify a global schema that describes the relations and mappings between the various schemata to be integrated.

Queries and Deductive Reasoning Along another line, the explicit articulation of domain knowledge often serves the purpose of answering queries about the represented knowledge. Thereby the system should also return *implicit* knowledge which was not explicitly articulated but can be inferred from the supplied knowledge. In contrast to inductive reasoning, as we have encountered in the previous chapter, this reasoning is *deductive*. Intuitively, for this reasoning to yield the desired results, we need to endow our representation with a formal semantics that precisely specifies the effects of a certain representation primitive on the resulting model.

While each of these clusters emphasizes a different aspect, they draw a useful picture of the main roots of knowledge representation.

3.1.2 Ontologies and Knowledge Structures

We will now look in closer detail at the basic notions of knowledge representation and at the notions of knowledge structures and ontologies in computer science.

Ontologies in Computer Science

The term *Ontology* itself denotes a philosophical discipline occupied with reasoning about the fundamental categories of what sorts or kinds of things there are in the universe. While the term *Ontology* as a name refers to this discipline, the term *ontology* as a common noun has slightly different interpretations in philosophy and in computer science. In philosophy, an ontology is a particular system of categories accounting for a certain view of the world.

Originating from the philosophical context, this terminology has been taken up and found widespread usage in diverse branches of computer science. Especially, ontologies have recently found much attention as the backbone of the *Semantic Web* (Berners-Lee et al., 2001). In this sense, ontologies are are formal engineering artifacts, designed to provide a common schema for storing data and for explicating the understanding of the knowledge in a domain of interest. The probably most prominent recent characterization of ontologies is given by Gruber (1993, own emphases):

> "An ontology is an *explicit* specification of a *conceptualization*. [...] In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) *with human-readable text* describing what the names mean, and *formal axioms* that constrain the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a logical theory."

An ontology thus comprises (i) a vocabulary of symbols used to describe the relevant domain and (ii) a number of assumptions about the meaning of the vocabulary entries. Studer et al. (1998) build upon this definition and further characterize the conceptualization to be *shared*, i.e. consensual among different stakeholders.

Classes of Ontologies

Guarino (1998) distinguishes ontologies and ontology languages according to their degree of formality. While *informal* ontologies are usually expressed in an ontology language which largely draws from the natural language descriptions of the employed primitives, a *formal* ontology is expressed in a formal ontology language, usually represented in first-order logic or fragments thereof. These languages give their constructs a clear mathematical interpretation. In the remainder, we will refer to informal ontologies and general knowledge representation models as *knowledge structures*. We will refer to *ontologies* only as advanced knowledge structures where knowledge is expressed by means of a formalism that supports a minimum number of modeling primitives and which is formally grounded in a logical theory.

Along another line, Guarino (1998) distinguishes ontologies according to the subject of conceptualization. Different types of ontologies that are commonly suggested include top-level ontologies, domain ontologies, task ontologies and application ontologies, each of which comes with a different level of generality and focus. While top-level ontologies describe very general concepts independent of the application domain (e.g. relating to the representation of space and time), domain ontologies describe the concepts related to a specific domain (e.g. medical knowledge).

3.1.3 Semantic Web Standards

Knowledge structures and ontologies have recently attracted attention as the building blocks of the envisioned Semantic Web (Berners-Lee et al., 2001; Shadbolt et al., 2006). Conceptually, the Semantic Web is seen as an extension of the World Wide Web as a universal medium for data, information, and knowledge exchange. On the Semantic Web, content can be expressed not only in natural language for presentation to human users, but also in a format that can be interpreted directly by computer systems, permitting them to easily find, share and integrate information.

Practically, the backbone of the Semantic Web is constituted by a set of standards developed under the lead of the World Wide Web Consortium (W3C). In particular, the *Resource Description Framework (RDF)*, *Resource Description Framework Schema (RDFS)* and *Web Ontology Language (OWL)* standards form the core of the Semantic Web. While RDF (Manola and Miller, 2004) constitutes a rudimentary graph-based data model, RDFS (Brickley and Guha, 2004) supports established knowledge representation paradigms and a distinction between schema and instance level. Initially without a formal interpretation, these languages have later been given formal semantics (Hayes, 2004). This paradigm is extended by OWL, an ontology language that supports a rich set of advanced modelling primitives and has a clear logical grounding in Description Logics (McGuinness and van Harmelen, 2004; Patel-Schneider et al., 2004; Horrocks et al., 2003). Furthermore, with *Simple Protocol and RDF Query Language (SPARQL)*, a query language for the RDF data is under development (Prud'hommeaux and Seaborne, 2007). A description of some older knowledge representation languages on the Semantic Web which have lost practical importance since the advent of OWL is given by Gomez-Perez and Corcho (2001).

In practice, the interaction with Semantic Web data is performed by means of *ontology management systems*. These systems are software frameworks that provide a set of components that allow to interact and work with ontologies and knowledge structures in practical settings (Bloehdorn et al., 2006d). Standard components of ontology management systems are ontology editors and/or ontology management APIs, reasoning engines and query interfaces.

3.2 Formalizing Knowledge Structures

In this section, we formalize the notions of knowledge structures and ontologies. This formalization will constitute the basis for the subsequent exposition in this thesis.

3.2.1 Elementary Knowledge Structures

We begin by introducing the very general notion of an (elementary) knowledge structure which will be extended and refined subsequently.

Syntactic Elements

Definition 3.1 (Knowledge Structure)**.** A *knowledge structure* is a four-tuple $\mathcal{K} := (\mathcal{E}, \mathcal{P}, \mathcal{D}, \mathcal{S})$ consisting of a set $\mathcal{E}$ of *entity names*, a set $\mathcal{P} \subseteq \mathcal{E}$ of *property names*, a set $\mathcal{D}$ of *data values*, and a set $\mathcal{S} \subseteq \mathcal{E} \times \mathcal{P} \times (\mathcal{E} \cup \mathcal{D})$ of *statements*.

Remark 3.1 (Data Values). Note that we have deliberately not further specified the nature of the data values in the (potentially infinite) set $\mathcal{D}$. In general, these can be of any basic data type such as strings or integers. We will sometimes refer to the set of data values of a specific data type as D. The set $\mathcal{D} = D_{\text{String}} \cup D_{\mathbb{R}} \cup D_{\mathbb{N}} \cup \ldots$ is then seen as the set of supported data values.

In a general sense, we will refer to *entities* as all ontology elements that can be referenced, such as entity names or data values such as a specific character string. In contrast, *statements* classify and relate entities and may constrain their logical interpretation. The structure of the statement triples is devised in analogy with the basic structures we find in natural language sentences. Correspondingly, the first argument of each statement (any entity name) is referred to as the *subject*, the second argument (any property name) as the *predicate*, and the third argument (any entity name or value of one of the admitted data types) as the *object*.

Remark 3.2 (Properties as Entities). Note that properties form a subset of the overall entities, i.e. they may as well themselves appear as subjects or objects of statements. Sometimes, this level of generality is restricted and the sets $\mathcal{E}$ and $\mathcal{P}$ are instead required to be non-overlapping.

Practical Use and Examples

Conceptually, the statements in a knowledge structure form a directed and labeled graph. Graphs of this kind are alternatively also often referred to as *semantic networks*. The model basically corresponds to the RDF language for representing information about resources. The main distinction is that RDF additionally includes a small number of advanced concepts like the specification of sets and lists and the use of unnamed entities called *blank nodes* (Manola and Miller, 2004) both of which are not relevant for the further exposition in this thesis. The following examples illustrate the explication of such knowledge structures.[1]

Example 3.1 (Bibliographic Metadata and Topic Hierarchies). The elements of the knowledge structure would, for example allow us to encode bibliographic information like in the following example:

```
pub100      dc:title        "Kernel Methods for Knowledge Structures"
pub100      dc:creator      person100
person100   foaf:name       "Stephan Bloehdorn"
pub100      dc:subject      topic110
topic110    skos:prefLabel  "Machine Learning"
topic110    skos:altLabel   "Statistical Learning"
```

The example references property names taken from some of the best known RDF-based metadata standards such as the Dublin Core (DC) (DCMI Usage Board, 2006), Friend of a Friend (FOAF) (Brickley and Miller, 2007) and Simple Knowledge Organisation Systems (SKOS) (Miles and Brickley, 2005) metadata elements, marked by the corresponding namespaces.

[1]Since RDF, Semantic Web standards use the concept of Uniform Resource Identifiers (URIs) (Berners-Lee et al., 2005) to express entity names. In analogy with the concept of namespaces as found in Extensible Markup Language (XML) (Bray et al., 2006) we will shorten the notation by using a colon to separate the vocabulary identifier from an element in the vocabulary. We will also use the simple RDF abstract syntax (Klyne and Carroll, 2004) where an RDF triple is conventionally written in the order subject, predicate, object.

Often, such knowledge structures are used to encode taxonomic information as in the following example.

Example 3.2 (Bibliographic Metadata and Topic Hierarchies (cont.)). Consider the previous bibliographic example and the following knowledge structure:

```
topic110    skos:broader     topic100
topic120    skos:prefLabel   "Knowledge Representation"
topic120    skos:broader     topic100
topic100    skos:prefLabel   "Artificial Intelligence"
```

The knowledge structure implements the common organization of topics via broader-narrower relations we find in library systems.

Knowledge structures, as introduced so far are only syntactic structures but do not specify how these structures are to be interpreted. Correspondingly, the meaning of the entities needs to be described outside of the formal structure, e.g. in natural language descriptions and their proper use has to be checked by humans or needs to be ensured procedurally in the relevant applications. Consider Example 3.2 above, where it is not clear whether the `skos:broader` relation between two topics implies that an assignment of the specific topic should also imply an assignment of the more general topic. Such an approach is thus prone to suffer from interpretations changing over time or differing among implementing applications. A further level of complexity is added when different knowledge structures are to be combined, where the correspondence of the language elements has to be inspected and aligned manually. Also, while it is possible to query knowledge structures to retrieve resources that match certain patterns within the graph, there is no way of automatically inferring implicit facts or detecting inconsistencies. In the next section, we will extend the basic knowledge representation model to match these requirements.

3.2.2 Ontologies and Formal Semantics

We have argued earlier that (formal) ontologies should specify a sufficiently rich set of language primitives and that these primitives should be grounded in a logic-based interpretation to allow to reason with the supplied knowledge. In the following, we look at ontologies as an extension of the general model of knowledge structures considered so far. This model is designed in the light of two considerations. Firstly, the model is in line with a whole family of class-based knowledge representation languages and can be easily extended towards expressive DLs. Secondly, it has a well-defined formal semantics in the sense of first-order logic interpretations which again seamlessly fits into the family of DLs.

Syntactic Elements

The main primitives of the model are *individuals*, *classes*, and *object/data type properties* which, from the perspective of mathematical logic correspond to constants, unary predicates, and binary predicates, respectively as well as the notion of subsumption as a basic inference task.

Definition 3.2 (Ontology)**.** An *ontology* is a knowledge structure in the sense of Definition 3.1 with the following additions. The set of entities $\mathcal{E}$ is partitioned into two subsets $\mathcal{E} = \mathcal{E}_{\mathcal{V}} \cup \mathcal{E}_{\mathcal{O}}$ called the *description language entities* and *domain entities*. The set of domain entities is partitioned further as follows: $\mathcal{E}_{\mathcal{O}} = \mathcal{I} \cup \mathcal{C} \cup \mathcal{P}_0 \cup \mathcal{P}_D$ whereby $\mathcal{P}_0 \cup \mathcal{P}_D \subset \mathcal{P}$. We call $\mathcal{C}$ the set of *class names*, $\mathcal{I}$ the set of *individuals*, $\mathcal{P}_0$ the set of *object properties*, and $\mathcal{P}_D$ the set of *data properties*. The set of description language entities contains the properties $\{\mathsf{instanceOf}, \mathsf{subClassOf}, \mathsf{subPropertyOf}, \mathsf{domain}, \mathsf{range}\}$. The set of admissible statements in $\mathcal{S}$ is constrained according to the partition:

$$\begin{aligned}
\mathcal{S} \subseteq\ & (\mathcal{I} \times \{\mathsf{instanceOf}\} \times \mathcal{C}) \cup && \text{(class instantiation)}\\
& (\mathcal{I} \times \mathcal{P}_0 \times \mathcal{I}) \cup (\mathcal{I} \times \mathcal{P}_D \times \mathcal{D}) \cup && \text{(property filler)}\\
& (\mathcal{C} \times \{\mathsf{subClassOf}\} \times \mathcal{C}) \cup && \text{(class subsumption)}\\
& (\mathcal{P}_0 \times \{\mathsf{subPropertyOf}\} \times \mathcal{P}_0) \cup && \\
& (\mathcal{P}_D \times \{\mathsf{subPropertyOf}\} \times \mathcal{P}_D) \cup && \text{(property subsumption)}\\
& (\mathcal{P}_D \times \{\mathsf{domain}\} \times \mathcal{C}) \cup (\mathcal{P}_0 \times \{\mathsf{domain}\} \times \mathcal{C}) \cup && \text{(property domain restriction)}\\
& (\mathcal{P}_0 \times \{\mathsf{range}\} \times \mathcal{C}) && \text{(property range restriction)}
\end{aligned}$$

The basic elements of this ontology model, in particular the separation of schema-level and instance-level, can be found in all popular ontology languages. Schema-level statements encode *intensional* knowledge, i.e. properties of groups or abstractions of individuals, expressed by means of classes, properties (as entities) and their respective subsumptions (as axiom statements). Instance-level statements encode *extensional* knowledge, i.e. properties of particular individuals, expressed by individuals (as entities) and class and property fillers (as axiom statements). Sometimes, the intensional knowledge is also referred to as the ontology (in a strict sense), while the extensional knowledge is said to constitute the *knowledge base*.

Conceptually, the syntactic elements introduced in the model above correspond to the main syntactic elements available in RDFS. However, RDFS is not restrictive about the formation of the statements thus making the semantics as specified for RDF and RDFS (Baader et al., 2003; Patel-Schneider et al., 2004) more complex.[2]

[2]In fact, the RDFS language is specified via a cyclical metamodel where the language elements are described by references to themselves. As such, this structure produces several problems for a class-based interpretation of RDFS.

Model-Theoretic Semantics

Given an ontology structure in the sense of Definition 3.2, meaning of the basic modeling primitives is formally defined via a model theoretic semantics, i.e. by relating the language syntax to a model in an interpretation domain. We sketch this approach in the following definition.

Definition 3.3 (Class-Based Semantics for Ontologies)**.** Given an ontology structure in the sense of Definition 3.2, a *model* consists of a *domain* Δ^I consisting of a non-empty set of objects, a *domain of data values* $\Delta^I_\mathcal{D}$ representing the data values corresponding to $\mathcal{D}$, and an interpretation function I which maps entities of the ontology to concrete entities or groups of entities in the domain(s). The built-in classes $\top$ (Top) and $\bot$ (Bottom) map to Δ^I, i.e. all individuals in the domain, and to the empty set of objects, respectively. In particular, the following mappings hold (for convenience, we also include the common notation used in DL literature which we will often use):

Elements	**Semantics**	
$ind_1 \in \mathcal{I}$	${ind_1}^I \in \Delta^I$	
$val \in \mathcal{D}$	$val = val^I_\mathcal{D}$	
$\mathsf{class}_1 \in \mathcal{C}$	$\mathsf{class}_1{}^I \subseteq \Delta^I$	
$\mathsf{prop}_1 \in \mathcal{P}_0$	$\mathsf{prop}_1{}^I \subseteq \Delta^I \times \Delta^I$	
$\mathsf{prop}_2 \in \mathcal{P}_D$	$\mathsf{prop}_2{}^I \subseteq \Delta^I \times \Delta^I_\mathcal{D}$	
Statements	**Semantics**	**DL-Notation**
ind_1 type class_1	$ind_1 \in \mathsf{class}_1{}^I$	$\mathsf{class}_1(ind_1)$
ind_1 prop_1 ind_2	$({ind_1}^I, {ind_2}^I) \in \mathsf{prop}_1{}^I$	$\mathsf{prop}_1(ind_1, ind_2)$
class_1 subClassOf class_2	$\mathsf{class}_1{}^I \subseteq \mathsf{class}_2{}^I$	$\mathsf{class}_1 \sqsubseteq \mathsf{class}_2$
prop_1 subPropertyOf prop_2	$\mathsf{prop}_1{}^I \subseteq \mathsf{prop}_2{}^I$	$\mathsf{prop}_1 \sqsubseteq \mathsf{prop}_2$
prop_1 domain class_1	$\mathsf{prop}_1{}^I \subseteq \mathsf{class}_1{}^I \times \Delta^I$	—
prop_2 domain class_1	$\mathsf{prop}_1{}^I \subseteq \mathsf{class}_1{}^I \times \Delta^I_\mathcal{D}$	—
prop_1 range class_1	$\mathsf{prop}_1{}^I \subseteq \Delta^I \times \mathsf{class}_1{}^I$	—

An interpretation is said to *satisfy* an ontology (or any set of axioms), if there are no contradictions in the interpretation. Such an interpretation is called a *model* of the ontology (or set of axioms). If there are no such interpretations, the ontology is said to be *inconsistent*. If an additional axiom $A = (\mathsf{e_i}, \mathsf{prop_j}, \mathsf{e_k})$ holds in all models of an ontology $\mathcal{O}$, this axiom is said to be *entailed* by the ontology and this situation is denoted $\mathcal{O} \models A$. Two ontologies (or sets of axioms) are said to be *equivalent* if they have the same set of models.

A basic reasoning problem for a given ontology $\mathcal{O}$ are the questions of *subsumption* of classes and properties, i.e. whether for two classes $\mathsf{class}_1, \mathsf{class}_2$

(properties $\mathsf{prop}_1, \mathsf{prop}_2$) we have that $\mathcal{O} \models (\mathsf{class}_1 \ \mathsf{subClassOf} \ \mathsf{class}_2)$ ($\mathcal{O} \models (\mathsf{prop}_1 \ \mathsf{subPropertyOf} \ \mathsf{prop}_2)$). The structure where each class is associated with its immediate sub- and superclasses is called the *subsumption hierarchy* . A second basic reasoning procedure of particular practical importance is the task of *instance retrieval*. Given a set $\mathcal{I}$ of individuals and a class class_i, the retrieval task is to find all individuals ind_j such that $\mathcal{O} \models (ind_j \ \mathsf{instanceOf} \ \mathsf{class}_i)$. This task is particularly interesting when it is possible to form complex class descriptions as will be discussed below.

Remark 3.3 (Interpreting Individual Names). Note that the semantics given above ignores the naming of individuals as a means for distinguishing the corresponding domain entities. Specifically, two differently named individuals may be mapped to one and the same entity by the interpretation function. Sometimes, the axioms equal and differentFrom with the obvious interpretations are introduced to account for this fact. Alternative semantics that postulate the inequality of individuals with different naming are said to follow the *unique names assumption*.

Remark 3.4. Note that the equivalence of two classes (or properties) is not explicitly introduced but trivially follows from subsumption, i.e. $\mathsf{class}_1 \sqsubseteq \mathsf{class}_2 \wedge \mathsf{class}_2 \sqsubseteq \mathsf{class}_1 \Leftrightarrow \mathsf{class}_1 \equiv \mathsf{class}_2$.

The most common paradigm for queries to such ontologies are *conjunctive queries*. Such a query corresponds to a conjunction of DL atoms which may contain distinguished or non-distinguished variables. The result corresponds to individuals which are valid fillers of the distinguished variables with the non-distinguished variables existentially bound. Practically, such queries can be encoded for example in terms of SPARQL queries by interpreting the basic graph matching capabilities on knowledge structures by the semantics of the ontology language.

3.2.3 Advanced Ontology Constructs

While the presented syntactic elements allow for basic arrangement of information and retrieval tasks, several applications require more expressive semantics which also allow for more interesting reasoning problems. Advanced DLs build upon the basic ontology model and provide additional modelling primitives. As an example, we here informally introduce the language constructs of the Description Logic $\mathcal{SHOIN}(\mathbf{D})$ which constitutes the basis for the most prominent variant of OWL, called OWL-DL. In particular, in addition to the introduction of atomic classes it is possible to build complex class descriptions based on atomic classes by means of a set of constructors:

- $\mathsf{class}_1 \sqcap \mathsf{class}_2$ (intersection, conjunction), denoting the set of individuals that belong to both class_1 and class_2,

- $\mathsf{class}_1 \sqcup \mathsf{class}_2$ (union, disjunction), denoting the set of individuals that belong to either class_1 or class_1,
- $\neg\, \mathsf{class}_1$ (complement, negation), denoting the set of individuals that do not belong to class_1,
- $\forall \mathsf{prop}_1.\mathsf{class}_1$ (universal restriction), denoting the set of individuals that are related via the (object) property prop_1 only with individuals belonging to the concept class_1,
- $\exists \mathsf{prop}_1.\mathsf{class}_1$ (existential restriction), denoting the set of individuals that are related via the (object) property prop_1 with some individual belonging to the concept class_1,
- $\geq n\, \mathsf{prop}_1$ and $\leq n\, \mathsf{prop}_1$ (qualified number restriction), denoting the set of individuals that are related with at least (at most) n individuals via the (object) property prop_1.
- $\{ind_1, \ldots, ind_n\}$ (nominal), denoting the set of individuals that are explicitly enumerated.

Note that for such descriptions, single triples are not sufficient any more. Instead, each of these descriptions requires a set of triples which together describe their structure. In practice, the blank node feature of RDF is usually used to avoid unnecessary naming of complex classes.

Based on all available class descriptions, in addition to the axiom statements introduced in Definition 3.2, $\mathcal{SHOIN}(\mathbf{D})$ allows the following statements:

- axioms for transitivity, symmetry and functionality denoted $\mathsf{transitive}(\mathsf{prop}_1)$, $\mathsf{symmetric}(\mathsf{prop}_1)$, and $\mathsf{functional}(\mathsf{prop}_1)$ stating that the (object) property prop_1 is transitive, symmetric, or functional,
- inversion axioms $\mathsf{inverse}(\mathsf{prop}_1, \mathsf{prop}_2)$ stating that the (object) property prop_2 is the inverse of the (object) property prop_1,
- class equality axioms $\mathsf{equivalentClass}(\mathsf{class}_1, \mathsf{class}_2)$, stating that the sets of individuals in class_1 and class_2 are identical,
- class disjointness axioms $\mathsf{disjoint}(\mathsf{class}_1, \mathsf{class}_2)$, stating that the sets of individuals in class_1 and class_2 are disjoint,
- individual (in)equalities $ind_1 \approx ind_2$, and $ind_1 \not\approx ind_2$, respectively, stating that ind_1 and ind_2 denote the same (different) individuals.

In cases of such more expressive DLs, the subsumption hierarchy comprises complex descriptions besides the actual atomic classes. The practical importance of $\mathcal{SHOIN}(\mathbf{D})$ stems from the fact that it allows for a fairly expressive modelling language while remaining computationally realistic as complete reasoning procedures can be devised (i.e. all conclusions can actually be computed) and the logic remains decidable. We deliberately omit further details on the logical foundations as these aspects are not relevant for the core work of this thesis. The interested reader is referred to the rich literature on DLs, especially to the comprehensive presentation by Baader et al. (2003) for further reference.

3.3 Summary

In this chapter we have briefly introduced basic concepts from the field of knowledge representation which we will rely on in the subsequent chapters. We have motivated the distinction of possibly informal knowledge structures and ontologies and formalized these notions in terms of a graph-based knowledge representation model and a simple class-based description language with possible extensions.

Both models are rooted in specific Semantic Web standards, namely RDF, RDFS, and OWL but abstract away from the technical details of these languages. For further literature on these topics, we refer to the books by Staab and Studer (2004) as well as Hitzler et al. (2008) or to the comprehensive introduction by Grimm et al. (2007).

Part II

Kernels for Entities in Taxonomic Structures

Chapter 4

Kernel Functions for Entities in Taxonomic Structures

The notion of *similarity* plays a key role in the cognitive models we build of the world. It provides a principle by which humans classify objects, form concepts, and make generalizations. The quantification of similarities between entities is thus also an important issue in artificial intelligence and information management (Rissland, 2006). It has found most attention in the context of natural language processing, information retrieval and information integration (Budanitsky and Hirst, 2006; McHale, 1998; Li et al., 2003). In particular, various similarity functions have been proposed in literature that reflect notions of similarity between entities in taxonomies based on the overall structural setup of these taxonomic knowledge structures.

In Chapter 2, we have seen that kernel functions can be regarded as a special class of similarity functions for which an implicit embedding into a (possibly unknown) vector space is possible. Within this thesis, in particular in Part III, we encounter the need for similarity functions on entities in taxonomic knowledge structures which comply with the formal requirement of being valid kernel functions. This chapter thus investigates the most prominent similarity functions of this type that have been proposed in literature. For each of them, it answers the question whether (or under which conditions) the respective similarity function also constitutes a valid kernel. On the one hand, this is the first comprehensive formal analysis of similarity functions in taxonomic knowledge structures with respect to this question. On the other hand, it also sets the ground for the methods to be discussed later on.

The chapter is organized as follows. In Section 4.1, we first introduce the general notion of a similarity function as well as some basic concepts needed for the subsequent exposition. In Section 4.2, we then study the properties of various set-based similarity functions. The results of this section provide the tools needed for the next section but will sometimes also be used for themselves in other parts of this thesis. In Section 4.3 we then study prominent taxonomic similarity functions. For each of these similarity functions, we (i) review their main intuitions and (ii) show whether they constitute valid kernels. As we will see, some of these functions are positive semi-definite and can thus be readily used as kernel functions (or as parameters for kernel functions) while a large proportion does not exhibit this property in the general case. We shortly review pointers to related work in Section 4.4 and conclude with a short summary and discussion in Section 4.5.

4.1 Defining Similarity in Taxonomic Structures

The most prominent group of similarity functions for entities in knowledge structures are those functions that rely on the taxonomic backbone of the knowledge structures. Before we start investigating the actual similarity functions, this section discusses the general notion of a similarity function and the notions related to the taxonomic backbone of a knowledge structure.

4.1.1 Similarity Functions

A similarity function provides a mapping from pairs of objects to a set of similarity values. While any set of ordered values could be used, it is common to quantify similarities in terms of positive real numbers. We formalize these intuitions mathematically using the notion of a similarity function, or similarity measure.

Definition 4.1 (Similarity Function)**.** A similarity function on a set $\mathcal{A}$ of objects is a real-valued function $\text{sim} : \mathcal{A} \times \mathcal{A} \mapsto \mathbb{R}^+$, that measures the degree of similarity between the two input arguments. We generally require that $\text{sim}(\cdot,\cdot)$ is symmetric, i.e. $\text{sim}(x,y) = \text{sim}(y,x),\ \forall x,y \in \mathcal{A}$ and that it obeys the maximality axiom $\text{sim}(x,y) \leq \text{sim}(x,x),\ \forall x,y \in \mathcal{A}$.

Conceptually, $\text{sim}(\cdot,\cdot)$ captures the degree of similarity of two entities, whereby any entity must be at least as similar to itself than to any other entity. Lin (1998) has summarized a number of widely accepted intuitions about the basis of a similarity function.

(1) The similarity between two objects is related to their commonality. The more commonality they share, the more similar they are.

(2) The similarity between two objects is related to the differences between them. The more differences they have, the less similar they are.

In addition to these intuitions the third common intuition about similarity formulated by Lin (1998) introduces the notion of a (globally) maximal similarity value and the concept of identity:

(3) A maximum similarity between two objects is reached when the objects are identical, no matter how much commonality they share.

To account for this additional requirement, we introduce the notion of a normalized similarity function.

Definition 4.2 (Normalized Similarity Function)**.** A normalized similarity function on a set $\mathcal{A}$ of objects is a symmetric and real-valued function: $\mathrm{sim} : \mathcal{A} \times \mathcal{A} \mapsto [0,1]$, that measures the degree of similarity between the two input arguments. Hereby, sim is maximal for identical objects, i.e. $\mathrm{sim}(x,y) = 1 \Leftrightarrow x = y,\ x,y \in \mathcal{A}$.

Note that the choice of the interval $[0,1]$ is only for convenience and can be altered by rescaling by any positive factor. More importantly, similarity functions of this type enforce a constant self-similarity among objects. While this assumption seems intuitive, its adequacy has been challenged in psychological literature, e.g. by Krumhansl (1978) and some of the similarity functions we encounter will nevertheless allow different levels of self-similarity, i.e. they comply with Definition 4.1 but not necessarily with Definition 4.2.

Remark 4.1. Intuitions (1) and (2) are best captured in the *feature contrast model* by Tversky (1977) in one of the seminal psychological treatments of the topic. Tversky proposed a family of similarity functions, in which joint features tend to increase the perceived similarity of two objects while feature differences tend to diminish perceived similarity. Specifically, in the feature contrast model, the similarity of two objects x and y is defined as

$$\mathrm{sim}(x,y) = \theta\,\mu(\mathcal{F}(x) \cap \mathcal{F}(y)) - \alpha\,\mu(\mathcal{F}(x) \setminus \mathcal{F}(y)) - \beta\,\mu(\mathcal{F}(y) \setminus \mathcal{F}(x))\,,$$

whereby $\mathcal{F}(\cdot)$ denotes the set of *features* of the argument objects, $\mu(\cdot)$ denotes some measure on these sets (sometimes also referred to as *salience function*) and the parameters α, β and θ are fixed positive real numbers. Note, however that this family of functions is more general than required by intuitions (1) and (2) as it allows for negative similarity values and, more importantly, for asymmetric similarity functions (in case of $\alpha \neq \beta$).

To account for normalized similarity functions, i.e. for intuition (3), Tversky has proposed the *ratio model* as a variation of his first model of similarity functions. Specifically, in the ratio model, the similarity of two objects x and y is defined as

$$\mathrm{sim}(x,y) = \frac{\mu(\mathcal{F}(x) \cap \mathcal{F}(y))}{\mu(\mathcal{F}(x) \cap \mathcal{F}(y)) + \alpha\,\mu(\mathcal{F}(x) \setminus \mathcal{F}(y)) + \beta\,\mu(\mathcal{F}(y) \setminus \mathcal{F}(x))}$$

whereby $\mathcal{F}(\cdot)$ again denotes the set of *features* of the argument objects, $\mu(\cdot)$ denotes the corresponding *salience function* and the parameters α and β are fixed positive real numbers.

In the following, we investigate popular taxonomic similarity functions which can often (though not always) be reduced to Tversky's families of similarity functions by means of a specific choice of the relevant features, the salience function or the other parameters.

4.1.2 Structural Similarity in Taxonomic Structures

A major line of work on similarity functions for entities in knowledge structures are those that rely on the taxonomic backbone of the knowledge structures. This backbone organizes entities along their perceived generality. The basic assumption behind this view is that the structural properties of the taxonomic backbone correlate well with the perceived similarity of entities in the knowledge structure, a view that has been supported by various psychological studies (Budanitsky and Hirst, 2006).

The taxonomic backbone is obtained by retaining only the structure of a directed acyclic graph (DAG) induced by a dedicated *taxonomic relation* from an existing knowledge structure. For any two entities e_1 and e_2 directly linked by the taxonomic relation, we will say that e_2 is *direct superconcept of* e_1 or vice versa that e_1 is *direct subconcept of* e_2. For any two entities e_1 and e_2 that can be linked by multiple consecutive taxonomic relations we will say that e_2 is *superconcept of* e_1 or vice versa that e_1 is *subconcept of* e_2. This formalization is deliberately generic to capture a wide range of linguistic resources, taxonomies and ontologies. For formal ontologies, the taxonomy is mostly associated with the class subsumption hierarchy, i.e. the arrangement of classes via subClassOf relations. For informal ontologies, the taxonomy is often based on a specific set of relations, such as the skos : broader relation. This approach is particularly relevant for knowledge structures that model the semantic dependencies between natural language words and their senses, a group of structures which we will refer to as *lexical knowledge structures*. In practice, we will focus our attention to WORDNET synsets and the hypernym/hyponym relations among them.

Example 4.1 (WORDNET)**.** Similarity between two words is often represented by similarity between entities in lexical knowledge structures such as WORDNET that are associated with the two words. WORDNET is a large lexical reference system and semantic network (Miller et al., 1990; Miller, 1995).[1] WORDNET organizes English nouns, verbs, adjectives, and adverbs into sets of cognitive synonyms called *synsets*, each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. While the original WORDNET database is coded in a proprietary format, a translation to an RDFS/OWL encoding exists (van Assem et al., 2006). The following knowledge structure illustrates a fragment of WORDNET:

```
sense100    wn20schema:inSynset     synset500
sense100    wn20schema:word         word700
word700     wn20schema:lexicalForm  "island"
synset500   wn20schema:hyponymOf    synset800
sense200    wn20schema:inSynset     synset800
sense100    wn20schema:word         word900
word900     wn20schema:lexicalForm  "land"
```

[1] http://www.cogsci.princeton.edu/~wn/

This fragment references two words, namely `word700` having the lexical form *"island"* and `word900` having the lexical form *"land"*. The former is part of a sense that links to the synset `synset500` while the latter has a sense that links it to the synset `synset800`. These synsets stand in a hyponym relation, such that meaning of the word *"island"* is at least in some sense a hyponym (i.e. a specialization) of meaning of the word *"land"*.

4.1.3 Basic Notions on Taxonomic Structures

The similarity functions to be introduced subsequently require a set of preliminary notions and assumptions. Most similarity functions assume the taxonomic structure to form a DAG with a unique topmost element. For taxonomies based on the subClassOf relation in class-based ontology formalisms, this root entity is clearly the class of all individuals (Top). For any other knowledge structure we thus introduce a dedicated *root entity* which becomes superconcept of all entities that are not equipped with outgoing superconcept edges. This is particularly true for the WORDNET noun hierarchy, which up to version 2.0 defined 9 distinct *unique beginner concepts* up to which each concept can be traced.

Another notion that will be used in the following is notion of the *semantic cotopy* of an entity (Maedche and Staab, 2002).

Definition 4.3 (Semantic Cotopy)**.** The upper, lower and overall *semantic cotopies* $SC^+(e_1)$, $SC^-(e_1)$, and $SC(e_1)$ of an entity e_1 in a taxonomy are respectively defined as:

$$\begin{aligned} SC^+(e_1) &= \{e_i | e_i \text{ is superconcept of } e_1\} \\ SC^-(e_1) &= \{e_i | e_i \text{ is subconcept of } e_1\} \\ SC(e_1) &= SC^+(e_1) \cup SC^-(e_1) \end{aligned}$$

Further important notions for the specification of the similarity functions are the *distance* of two entities, the *depth* of an entity as well as the *Lowest Super Ordinate (LSO)* of two entities .

Definition 4.4 (Distance in a Taxonomy)**.** By the *distance* $\mathrm{dist}(e_1, e_2)$ of two entities, we will refer to the length of the shortest path(s) between e_1 and e_2 that traverses a common superconcept.

Note that in a DAG there can, in general, be multiple different paths that yield the minimal distance.

Definition 4.5 (Depth in a Taxonomy)**.** The *depth* of an entity in a taxonomy is then defined as: $\mathrm{depth}(e_1) = \mathrm{dist}(e_1, \mathrm{root})$.

Based on this, the *Lowest Super Ordinate* of two entities (or, alternatively, their *most specific common subsumer*) refers to the entity with maximal depth that subsumes them both. Distances can be easily computed from the taxonomy's adjacency matrix using the Floyd-Warshall algorithm (Floyd, 1962) for all pairs of entities and their superconcepts. Note that in a taxonomic structures that correspond to perfect tree structures, the path of an entity to the root entity is unique. The LSO of two entities is thus unique and necessarily lies on the shortest path between them. On the other hand, for taxonomic structures that correspond to general DAGs, i.e. for those that permit multiple inheritance, these properties do not hold.

4.2 Analysis of Set-Based Similarity Functions

In line with Tversky's notions, many similarity functions are defined in terms of compositions into sets of characteristic objects and various set operations on these sets. In this section, we will introduce a number of basic results on the positive semi-definiteness of popular set based similarity functions — traditionally sometimes also referred to as set similarity coefficients.

Set Intersection Coefficient

As the simplest set-based similarity function, we first introduce the set intersection coefficient.

Definition 4.6 (Set Intersection Coefficient)**.** Let $\mathcal{X}$ be an arbitrary set of objects. The set intersection similarity function for two sets $\mathcal{A}_1 \subseteq \mathcal{X}$ and $\mathcal{A}_2 \subseteq \mathcal{X}$ is defined as $\mathrm{sim}_\cap(\mathcal{A}_1, \mathcal{A}_2) = |\mathcal{A}_1 \cap \mathcal{A}_2|$.

Proposition 4.1. *The set intersection similarity function of Definition 4.6 is positive semi-definite.*

The result follows directly from the representation as the inner product of two binary vectors, each element of which encodes the presence or absence of an element of $\mathcal{X}$.

Crossproduct Similarity

While the set intersection coefficient only allows to relate the elements of two sets by their identity, the crossproduct similarity compares them based on an embedded similarity function.

Definition 4.7 (Crossproduct Similarity Function)**.** Let $\mathcal{X}$ be an arbitrary set of objects and $\mathrm{sim}(\cdot, \cdot)$ a similarity function on these elements. The crossproduct similarity for two sets $\mathcal{A} \subseteq \mathcal{X}$ and $\mathcal{B} \subseteq \mathcal{X}$ is defined as:

$$\mathrm{sim}_{cp}(\mathcal{A},\mathcal{B}) = \sum_{a\in\mathcal{A}} \sum_{b\in\mathcal{B}} \mathrm{sim}(a,b).$$

Proposition 4.2. *If* $\mathrm{sim}(\cdot,\cdot)$ *is a positive semi-definite function, the set crossproduct similarity function of Definition 4.7 is positive semi-definite.*

Proof. Let $\phi \mapsto \mathcal{H}$ be the feature space mapping associated with the positive semi-definite function $\mathrm{sim}(\cdot,\cdot)$ into some feature space $\mathcal{H}$. We then have

$$\begin{aligned}\mathrm{sim}_{cp}(\mathcal{A},\mathcal{B}) &= \sum_{a\in\mathcal{A}} \sum_{b\in\mathcal{B}} \mathrm{sim}(a,b) = \sum_{a\in\mathcal{A}} \sum_{b\in\mathcal{B}} \langle\phi(a),\phi(b)\rangle \\ &= \left\langle \sum_{a\in\mathcal{A}} \phi(a), \sum_{b\in\mathcal{B}} \phi(b) \right\rangle.\end{aligned}$$

As the crossproduct similarity function can be reformulated as an inner product, it constitutes a positive positive semi-definite function. □

In such cases, we will usually refer to the crossproduct similarity function as the *crossproduct kernel*.

Jaccard Coefficient

The Jaccard coefficient is a statistic used for comparing the similarity and diversity of sample sets.

Definition 4.8 (Jaccard Coefficient)**.** Let $\mathcal{X}$ be an arbitrary set of objects. The Jaccard similarity coefficient for two sets $\mathcal{A}_1 \subseteq \mathcal{X}$ and $\mathcal{A}_2 \subseteq \mathcal{X}$ is defined as

$$\mathrm{sim}_{JAC}(\mathcal{A}_1,\mathcal{A}_2) = \frac{|\mathcal{A}_1 \cap \mathcal{A}_2|}{|\mathcal{A}_1 \cup \mathcal{A}_2|}.$$

Note that the Jaccard coefficient corresponds to the value of Tversky's ratio model when setting $\alpha = \beta = 1$. To analyze the positive semi-definiteness of this similarity function, we require the following result by Gower (1971, proof therein).

Proposition 4.3 (Gower (1971))**.** *Let* $f : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ *be a positive semi-definite function and let* $c \in \mathbb{R}$ *be a constant such that for all* $x \in \mathcal{X}$ *we have that* $f(x,x) < c$*. Then the function* $f' : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ *given by* $f'(x,z) = 1/(c - f(x,z))$ *is positive semi-definite.*

Proposition 4.4. *The Jaccard coefficient of two sets is a positive semi-definite function.*

Proof. For any set $\mathcal{A} \subseteq \mathcal{X}$ we define its complement as $\overline{\mathcal{A}} = \mathcal{X} \setminus \mathcal{A}$. Clearly, by our earlier reasoning in terms of vector representations, $|\overline{\mathcal{A}_1} \cap \overline{\mathcal{A}_2}|$ is bounded by $|\mathcal{X}|$. Clearly, we have that $|\mathcal{A}_1 \cup \mathcal{A}_2| = |\mathcal{X}| - |\overline{\mathcal{A}_1} \cap \overline{\mathcal{A}_2}|$. From Proposition 4.3, it follows that $1/(|\mathcal{A}_1 \cup \mathcal{A}_2|)$ is positive semi-definite. Together with Proposition 4.1 it follows that the Jaccard Coefficient is positive semi-definite (p.s.d.) as it is the product of two positive semi-definite functions. □

Dice Coefficient

The Dice coefficient is another popular statistic for comparing two sample sets.

Definition 4.9 (Dice Coefficient)**.** Let $\mathcal{X}$ be an arbitrary set of objects. The *Dice similarity coefficient* for two sets $\mathcal{A}_1 \subseteq \mathcal{X}$ and $\mathcal{A}_2 \subseteq \mathcal{X}$ is defined as

$$\text{sim}_{DICE}(\mathcal{A}_1, \mathcal{A}_2) = \frac{2\,|\mathcal{A}_1 \cap \mathcal{A}_2|}{|\mathcal{A}_1| + |\mathcal{A}_2|}\,.$$

Again note that the Dice coefficient corresponds to Tversky's ratio model when setting $\alpha = \beta = 0.5$. For the Dice coefficient, we simply state the result by Gower and Legendre (1986).

Proposition 4.5 (Positive Semi-Definiteness of Dice Coefficient (Gower and Legendre, 1986))**.** *The Dice coefficient of two sets is a positive semi-definite function.*

We shortly sketch the proof of Gower and Legendre (1986). The main idea is to look at the nominator and the denominator separately. Clearly, the nominator, i.e. $2\,|\mathcal{A}_1 \cap \mathcal{A}_2|$ is positive semi-definite as it is an instance of the intersection kernel (positivly scaled by 2). It is now sufficient to show that $(|\mathcal{A}_1| + |\mathcal{A}_2|)^{-1}$ is a positive semi-definite function. Gower and Legendre (1986) proof this by showing that, as $|\mathcal{A}_i|$ are non-negative the determinant of any matrix formed out of this second function has a non-negative determinant, a sufficient condition for the positive semi-definiteness of a matrix (Shawe-Taylor and Cristianini, 2004).

Introducing Measures

All set-based similarity functions can be extended to take into account (bounded) measures $\mu : 2^{\mathcal{X}} \mapsto [0, C]$ on the subsets of a reference set $\mathcal{X}$ with $\mu(\mathcal{X}) = C < \infty$ other than the basic set cardinality $\mu(\cdot) = |\cdot|$. The use of a different measure allows to extend the similarity functions introduced so far in such a way that different objects within the reference sets can be emphasized or deemphasized when computing the similarity functions. The results of the analysis above carry over to this setting, c.f. for example to the analysis by Gärtner (2005).

4.3 Analysis of Taxonomic Similarity Functions

We now study the most prominent similarity functions on entities in knowledge structures that rely on the taxonomic backbone for computing similarity values. We present both *structural* and *information theoretic similarity functions*, as well as combinations thereof. We focus on the main rationales behind these measures, pointing to the recent survey by Budanitsky and Hirst (2006) for detailed complementary information on the topic. For each of the similarity functions, we then investigate whether it constitutes a positive semi-definite function. For most similarity functions, we can show that they do not correspond to positive semi-definite functions in the general case by constructing appropriate counterexamples. However, these similarity functions can sometimes be shown to be positive semi-definite functions if the structure of the taxonomic backbone is appropriately restricted.

4.3.1 Path-Based Taxonomic Similarity

The main assumption behind path-based similarity functions is that the distance between to entities is the major indicator of their semantic similarity. The *inverted path length* can be seen as an example of a particularly simple way to compute semantic similarity between two entities in a taxonomy.

Definition 4.10 (Path-Based Taxonomic Similarity)**.** The similarity $\mathrm{sim}_{IPL}(\mathsf{e}_1, \mathsf{e}_2)$ of two entities e_1 and e_2 is defined as

$$\mathrm{sim}_{IPL}(\mathsf{e}_1, \mathsf{e}_2) = \frac{1}{(1 + \mathrm{dist}(\mathsf{e}_1, \mathsf{e}_2))^{\alpha}}$$

where $\alpha > 0$ specifies a decay factor.

Based on its first reported use by Rada and Bicknell (1989), this similarity function is also referred to as the *Rada measure*.

Proposition 4.6. *The similarity of two entities in a taxonomic structure according to the similarity function introduced in Definition 4.10, i.e. the Inverted Path Length similarity, is not positive semi-definite.*

Proof. The proposition is shown by counterexample. Consider the fragment of a taxonomic network in Figure 4.1. For the inverted path length similarity function with $\alpha = 1$, this gives rise to the following similarity matrix for the entities $\{A, \ldots, K\}$. We hereby order the matrix indices according to the alphabetical order of the variables.

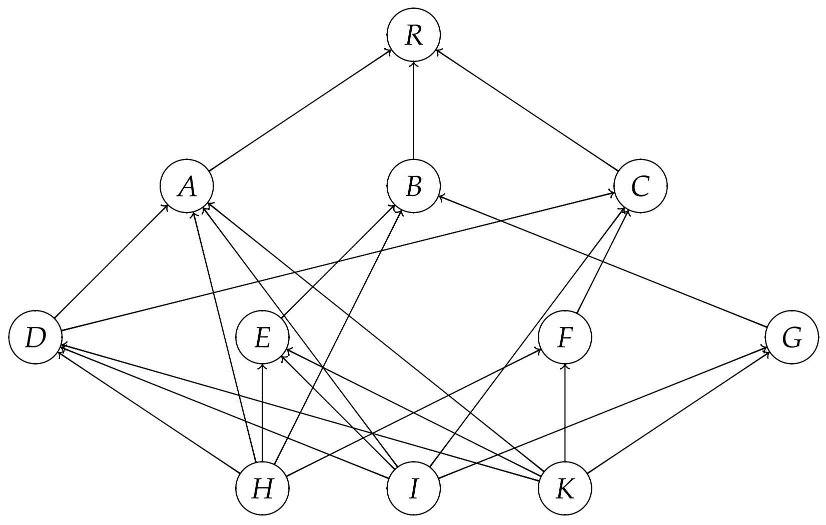


Figure 4.1 — Taxonomic network as counterexample for the Inverted Path Length Measure. The similarity matrix formed by restriction to entities $\{A, \ldots, K\}$ for the Inverted Path Length Measure is not positive semi-definite.

$$\begin{pmatrix}
1 & 1/3 & 1/3 & 1/2 & 1/4 & 1/4 & 1/4 & 1/2 & 1/2 & 1/2 \\
1/3 & 1 & 1/3 & 1/4 & 1/2 & 1/4 & 1/2 & 1/2 & 1/3 & 1/3 \\
1/3 & 1/3 & 1 & 1/2 & 1/4 & 1/2 & 1/4 & 1/3 & 1/2 & 1/3 \\
1/2 & 1/4 & 1/2 & 1 & 1/5 & 1/3 & 1/5 & 1/2 & 1/2 & 1/2 \\
1/4 & 1/2 & 1/4 & 1/5 & 1 & 1/5 & 1/3 & 1/2 & 1/2 & 1/2 \\
1/4 & 1/4 & 1/2 & 1/3 & 1/5 & 1 & 1/5 & 1/2 & 1/3 & 1/2 \\
1/4 & 1/2 & 1/4 & 1/5 & 1/3 & 1/5 & 1 & 1/3 & 1/2 & 1/2 \\
1/2 & 1/2 & 1/3 & 1/2 & 1/2 & 1/2 & 1/3 & 1 & 1/3 & 1/3 \\
1/2 & 1/3 & 1/2 & 1/2 & 1/2 & 1/3 & 1/2 & 1/3 & 1 & 1/3 \\
1/2 & 1/3 & 1/3 & 1/2 & 1/2 & 1/2 & 1/2 & 1/3 & 1/3 & 1
\end{pmatrix}$$

This matrix, however, has eigenvalues $\lambda' \approx (-0.001, \ldots, 4.463)$. As the smallest eigenvalue is negative, the matrix is not positive semi-definite. □

While this counterexample may not correspond to a typical setup in a taxonomic knowledge structure, it is one of the most compact examples that can be stated in a limited amount of space. In fact, various other counterexamples can be formed which correspond to more common situations in taxonomic knowledge structures and also exhibit larger negative eigenvalues. However, these examples require substantially more entities making them unsuitable for presentation in a limited amount of space.

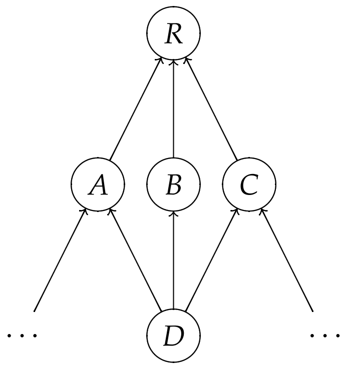


Figure 4.2 — Taxonomic network as counterexample for the Wu-Palmer, Resnik and Lin Measures. The similarity matrix formed by restriction to entities $\{A, B, C, D\}$ for the Wu-Palmer similarity function as well as – for appropriately chosen probabilities – the Resnik and Lin similarity functions is not positive semi-definite.

4.3.2 Taxonomic Similarity by Wu&Palmer

While the simplicity of the pure path-based similarity function is intriguing, it does not comply with the common intuition that concepts closer to the root of the semantic network should have a higher distance compared to concepts far away. Among many others, the similarity function introduced by Wu and Palmer (1994) tries to scale the similarity with respect to the depth of the entities and the LSO in the taxonomy:

Definition 4.11 (Taxonomic Similarity by Wu and Palmer (1994))**.** The similarity of two entities is defined as

$$\mathrm{sim}_{WP}(\mathsf{e}_1, \mathsf{e}_2) = \frac{2\ \mathrm{depth}(\mathrm{lso}(\mathsf{e}_1, \mathsf{e}_2))}{\mathrm{dist}(\mathsf{e}_1, \mathrm{lso}(\mathsf{e}_1, \mathsf{e}_2)) + \mathrm{dist}(\mathsf{e}_2, \mathrm{lso}(\mathsf{e}_1, \mathsf{e}_2)) + 2\ \mathrm{depth}(\mathrm{lso}(\mathsf{e}_1, \mathsf{e}_2))}.$$

Proposition 4.7. *The similarity of two entities in a taxonomic structure according to the similarity function introduced in Definition 4.11, i.e. the similarity function introduced by Wu and Palmer (1994), is not positive semi-definite.*

Proof. Again, the proposition is shown by counterexample. Consider the fragment of a taxonomic network in Figure 4.2. The similarity matrix of the entities $\{A, B, C, D\}$ is (again, matrix indices are in order of the entities):

$$\begin{pmatrix} 1 & 0 & 0 & 2/3 \\ 0 & 1 & 0 & 2/3 \\ 0 & 0 & 1 & 2/3 \\ 2/3 & 2/3 & 2/3 & 1 \end{pmatrix}.$$

For example, the matrix entry $\mathbf{M} = 2/3$ refers to the similarity of entity A and D. The Lowest Super Ordinate of these two entities is A at depth 1. The distances of the argument entities to the LSO are 0 for A and 1 for D, thus yielding $2 \cdot 1/(0+1+2\cdot 1) = 2/3$. This matrix, however, has eigenvalues $\lambda' \approx (-0.15, 1, 1, 2.15)$. As the smallest eigenvalue is negative, the matrix can not be positive semi-definite. □

Obviously, the finding that the similarity function by Wu and Palmer is not positive semi-definite in general, does not exclude the possibility that matrices restricted to specific sets of entities may nevertheless be positive semi-definite. Similarly, tighter assumptions in the taxonomic structure can enforce the positive semi-definiteness of the similarity function as shown in the following proposition.

Proposition 4.8. *The similarity of two entities in a taxonomic structure according to the similarity function introduced in Definition 4.11, i.e. the similarity function introduced by Wu and Palmer (1994), is positive semi-definite if the taxonomic backbone does not allow for multiple inheritance, i.e. if it is a perfect tree structure.*

Proof. Consider a representation of any entity e_i as a set $\mathcal{F}(\mathsf{e}_i)$ of edges on the unique path from this entity to the root entity. We then have that:

$$\begin{aligned}
\mathrm{sim}_{WP}(\mathsf{e}_1,\mathsf{e}_2) &= \frac{2\ \mathrm{depth}(\mathrm{lso}(\mathsf{e}_1,\mathsf{e}_2))}{\mathrm{dist}(\mathsf{e}_1,\mathrm{lso}(\mathsf{e}_1,\mathsf{e}_2)) + \mathrm{dist}(\mathsf{e}_2,\mathrm{lso}(\mathsf{e}_1,\mathsf{e}_2)) + 2\ \mathrm{depth}(\mathrm{lso}(\mathsf{e}_1,\mathsf{e}_2))} \\
&= \frac{2\ \mathrm{depth}(\mathrm{lso}(\mathsf{e}_1,\mathsf{e}_2))}{\mathrm{depth}(\mathsf{e}_1) + \mathrm{depth}(\mathsf{e}_2)} \\
&= \frac{2\,|\mathcal{F}(\mathrm{lso}(\mathsf{e}_1,\mathsf{e}_2))|}{|\mathcal{F}(\mathsf{e}_1)| + |\mathcal{F}(\mathsf{e}_2)|} = \frac{2\,|\mathcal{F}(\mathsf{e}_1) \cap \mathcal{F}(\mathsf{e}_2)|}{|\mathcal{F}(\mathsf{e}_1)| + |\mathcal{F}(\mathsf{e}_2)|}.
\end{aligned}$$

The Wu & Palmer similarity function thus has a representation in terms of the Dice coefficient, which is positive semi-definite according to Proposition 4.5. □

The proof relies on the finding that in a tree structure, there is only a single unique path between an entity and the root entity and that the Lowest Super Ordinate of two entities, necessarily lies on their respective paths to the root.

4.3.3 Information Content Based Similarity by Resnik

A different type of similarity functions tries to incorporate additional knowledge about the information content of a concept besides the structural setup of the taxonomic backbone. Resnik (1999) has argued that neither the individual edges nor the absolute depth in a taxonomy can be considered as homogeneous indicators of the "semantic content" of an entity. To overcome this problem, he introduces the notion of the probability $P(\mathsf{e})$ of encountering the entity e. This probability is typically estimated by the relative frequencies of the instantiations of the concept. For the case of lexical knowledge structures, e.g. WORDNET, it is common to obtain these frequencies by counting the number of lexicalizations of the concept in a corpus relevant for the domain under consideration. Here the counts of subconcepts equally contribute to their respective superconcepts. Resnik follows the argumentation of information theory in quantifying the *information content (IC)* of an observation as the negative log likelihood of the probability. By means of the argument that *"one key to the similarity of two concepts is the extent to which they share information in common"* he proposes the following function for the similarity of two concepts.

Definition 4.12 (Taxonomic Similarity by Resnik (1999))**.** Let $P(\mathsf{e})$ be the probability of encountering e. The similarity $\mathrm{sim}_{RES}(\mathsf{e}_1, \mathsf{e}_2)$ of two entities e_1 and e_2 is then defined as $\mathrm{sim}_{RES}(\mathsf{e}_1, \mathsf{e}_2) = \mathrm{IC}(\mathrm{lso}(\mathsf{e}_1, \mathsf{e}_2)) = -\log\, P(\mathrm{lso}(\mathsf{e}_1, \mathsf{e}_2))$, i.e. as the information content of the Lowest Super Ordinate.

The similarity function by Resnik has been highly influential in the field of similarity functions for knowledge structures. However, it has been repeatedly criticized because it is not normalized and thus allows varying degrees of self-similarity. While a universal root concept having a probability of 1 will carry an information content equal to zero, rare concepts will carry high information content values. Self-similarity thus increases for more specific concepts and can in practical settings easily achieve very high values.

Proposition 4.9. *The similarity of two entities in a taxonomic structure according to the similarity function introduced in Definition 4.12, i.e. the similarity function introduced by Resnik (1999), is not positive semi-definite.*

Proof. Consider again the fragment of a taxonomic network in Figure 4.2 with the following probabilities of the entities: $P(A) = 0.5$, $P(B) = 0.5$, $P(C) = 0.5$, and $P(D) = 0.25$. Then the respective amounts of information content according to Definition 4.12 will be $\mathrm{IC}(A) = 1$, $\mathrm{IC}(B) = 1$, $\mathrm{IC}(C) = C$, and $\mathrm{IC}(D) = 2$. These values give rise to the following similarity matrix for the entities $\{A, B, C, D\}$ (matrix indices in order of the entities):

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix}.$$

This matrix, however, has eigenvalues $\lambda' \approx (-0.30, 1, 1, 3.30)$. As the smallest eigenvalue is negative, the matrix can not be positive semi-definite. □

Again, this result does not carry over to the case of tree structures. This result is due to Mavroeidis et al. (2005a) and we here repeat it for completeness.

Proposition 4.10. *The similarity of two entities in a taxonomic structure according to the similarity function introduced in Definition 4.12, i.e. the similarity function introduced by Resnik (1999), is positive semi-definite if the taxonomic backbone does not allow for multiple inheritance, i.e. if it is a perfect tree structure.*

Proof. Again consider a representation of any entity e_i as a set $\mathcal{F}(\mathsf{e}_i)$ of edges on the unique path from this entity to the root entity. Define a measure $\mu(\cdot)$ on the edges such that for each edge connecting an entity e_1 and its direct superconcept e_2, the measure returns the difference of the information content value of e_1 and e_2. We then have that

$$\begin{aligned} \mathrm{sim}_{RES}(\mathsf{e}_1, \mathsf{e}_2) &= -\log P(\mathrm{lso}(\mathsf{e}_1, \mathsf{e}_2)) \\ &= \mu(\mathcal{F}(\mathrm{lso}(\mathsf{e}_1, \mathsf{e}_2)) \\ &= \mu(\mathcal{F}(\mathsf{e}_1) \cap \mathcal{F}(\mathsf{e}_2). \end{aligned}$$

This means that the Resnik similarity function can be represented as in terms of the set intersection coefficient for an appropriate fixed choice of the measure $\mu(\cdot)$. The positive semi-definiteness of the set intersection coefficient has been stated in Proposition 4.1. □

Again, the proof relies on the fact that the Resnik similarity function relies only on the information content of the unique LSO of two entities and that this value can be determined by looking only at the common path elements of the two elements on their shortest paths to the root.

4.3.4 Information Content Based Similarity by Lin

Based on Resnik's proposal, Lin (1998) derived a theoretically well motivated similarity function given by:

Definition 4.13 (Taxonomic Similarity by Lin (1998))**.** The similarity $\mathrm{sim}_{LIN}(\mathsf{e}_1, \mathsf{e}_2)$ of two entities e_1 and e_2 is defined as

$$\mathrm{sim}_{LIN}(\mathsf{e}_1, \mathsf{e}_2) = \frac{2 \log P(\mathrm{lso}(\mathsf{e}_1, \mathsf{e}_2))}{\log P(\mathsf{e}_1) + \log P(\mathsf{e}_2)}.$$

As an extension to the similarity function proposed by Resnik, it includes the information content of the compared entities as a means for normalization.

Proposition 4.11. *The similarity of two entities in a taxonomic structure according to the similarity function introduced in Definition 4.13, i.e. the similarity function introduced by Lin (1998), is not positive semi-definite.*

Proof. The result follows directly from the fact that the Wu-Palmer similarity function is not positive semi-definite. Consider again the fragment of a taxonomic network in Figure 4.2 with the following probabilities of the entities: $P(A) = 0.5$, $P(B) = 0.5$, $P(C) = 0.5$, and $P(D) = 0.25$. But then the respective amounts of information content will be $\mathrm{IC}(A) = 1$, $\mathrm{IC}(B) = 1$, $\mathrm{IC}(C) = C$, and $\mathrm{IC}(D) = 2$. These correspond directly to the distances of the entities to the root entity in the Wu-Palmer similarity and thus give rise to the identical similarity matrix which, however, has been shown to be indefinite. □

Again, this result is confined to knowledge structures that allow multiple inheritance.

Proposition 4.12. *The similarity of two entities in a taxonomic structure according to the similarity function introduced in Definition 4.13, i.e. the similarity function introduced by Lin (1998), is positive semi-definite if the taxonomic backbone does not allow for multiple inheritance, i.e. if it is a perfect tree structure.*

The proposition directly follows as a result of Proposition 4.8, i.e. from the positive semi-definiteness of the Wu & Palmer similarity function, when used with the same similarity function for the sets of edges as used in the case of the Resnik similarity function in the proof of Proposition 4.10.

4.3.5 Taxonomic Overlap and Related Similarity Functions

Maedche and Staab (2002) introduce a notion of the similarity that compares two entities in a taxonomy by means of their intensional semantics, i.e. by comparing them via all other entities they directly relate to by means of taxonomic relations.

Definition 4.14 (Taxonomic Overlap)**.** The similarity $\mathrm{sim}_{TO}(\mathsf{e}_1, \mathsf{e}_2)$ of two entities e_1 and e_2 is defined as

$$\mathrm{sim}_{TO}(\mathsf{e}_1, \mathsf{e}_2) = |\,\mathrm{SC}(\mathsf{e}_1) \cap \mathrm{SC}(\mathsf{e}_2)|\,.$$

Depending on the application context, the taxonomic overlap can be varied by restricting the semantic cotopies to superconcepts (i.e. $\mathrm{SC}^+(\cdot)$) or to subconcepts (i.e. $\mathrm{SC}^-(\cdot)$).

Note again that the cardinality of the intersection can be calculated by means of an inner product if each set is represented as a binary vector of its entries. This representation as a vector leads to an alternative explicit formulation by considering a weighting function for the entities in the semantic cotopy. Weighting can be useful to limit the effect of entities which reside far away from both reference concepts. In theory, any of the similarity functions we encounter in this section can be used for this purpose.

Definition 4.15 (Weighted Taxonomic Overlap)**.** The similarity of two entities is defined as

$$\mathrm{sim}_{TOW}(e_1, e_2) = \sum_{e_i \in \mathcal{K}} [e_i \in SC(e_1)][e_i \in SC(e_2)] \, \mathrm{sim}(e_1, e_i) \, \mathrm{sim}(e_2, e_i) \,.$$

The notation $[\cdot]$ refers to indicator functions evaluating to '1' if the backeted expression is true or to '0' otherwise. Again, the weighted taxonomic overlap can be varied by restricting the semantic cotopies to superconcepts (i.e. $SC^+(\cdot)$) or to subconcepts (i.e. $SC^-(\cdot)$). Hereby, $\mathrm{sim}(e_i, e_j)$ is an arbitrary weighting function.

Proposition 4.13. *The taxonomic overlap similarity functions of Definition 4.14 and 4.15 constitute positive semi-definite functions.*

The result follows from Proposition 4.1 or directly from the alternative formulation as an inner product.

Both variants of the taxonomic overlap are not normalized and may thus yield varying levels of similarity. In their original formulation, Maedche and Staab (2002) thus normalized the taxonomic overlap. We here report only on the unweighted variant of this similarity function.

Definition 4.16 (Normalized Overlap (Maedche and Staab, 2002))**.** The normalized taxonomic overlap of two entities is defined as

$$\mathrm{sim}_{TO}(e_1, e_2) = \frac{|\,SC(e_1) \cap SC(e_2)|}{|\,SC(e_1) \cup SC(e_2)|} \,.$$

Depending on the application context, the taxonomic overlap can be varied by restricting the semantic cotopies to superconcepts (i.e. $SC^+(\cdot)$), to subconcepts (i.e. $SC^-(\cdot)$).

Proposition 4.14. *The normalized taxonomic overlap of two entities of Definition 4.16 is a positive semi-definite function.*

The result follows directly from the positive semi-definiteness of the Jaccard Coefficient in Proposition 4.4.

4.4 Related Work

With the increased interest in ontologies and formal knowledge structures, the measurement of semantic similarity of entities in taxonomic structures has become a popular topic in the last decade (Budanitsky and Hirst, 2006). For large taxonomies like WORDNET, the computation of many of the structural components of these similarity functions requires substantial computational resources which have only recently become available. Note that the use of the taxonomic similarity functions which we have discussed is not constrained to informal knowledge structures. For example, Borgida et al. (2005) discuss the use of these similarity functions in the context of complex DL-based ontologies. Their goal is to generalize these earlier efforts for defining similarities for atomic concepts in order to obtain a way for assessing similarity between complex concept descriptions.

The positive semi-definiteness of the intersection coefficient and the crossproduct similarity are well known in the kernel community (see e.g. Shawe-Taylor and Cristianini (2004)). Results on the positive semi-definiteness of the Dice and Jaccard coefficients have been reported by Gower (1971), Gower and Legendre (1986) as well as Zegers (1986). However, these results are not well known in the kernel community. To the best of my knowledge, the only reference for an analysis of the positive semi-definiteness of a taxonomic similarity function is the proof of the positive semi-definiteness of the Resnik similarity function in tree structures by Mavroeidis et al. (2005a).

This exposition in this chapter has covered the best known taxonomic similarity functions. Future work will also look at less popular similarity functions like e.g. the *conceptual density* of Agirre and Rigau (1996) which was primarily designed for word sense disambiguation purposes or the similarity functions proposed by Li et al. (2003).

4.5 Summary and Discussion

In this chapter we have introduced and analyzed the popular similarity functions in taxonomic knowledge structures. Our exposition constitutes the first comprehensive investigation of these similarity functions in terms of the question whether they are positive semi-definite. On the one hand, we have seen that the multiple inheritance structure we find in most ontologies and (lexical) knowledge structures, e.g. WORDNET affects the positive semi-definiteness of many classical similarity functions. The positive semi-definiteness of these similarity functions, and thereby their suitability as kernel functions or parameters for kernel functions, can be ensured only if the background taxonomies conform to perfect tree structures. On the other hand, similarity functions of the families of taxonomic overlap do not show this limitation and will thus be the primary focus of our treatment in Part III in the context of Semantic Smoothing Kernels and Semantic Syntactic Tree Kernels.

Part III

Kernels for Semantic Smoothing in Text Mining

Chapter 5

Designing Kernel Functions for Semantic Smoothing

The overall topic of this thesis is the design of kernel functions that exploit declarative knowledge encoded in formal knowledge structures during learning. This chapter investigates an instantiation of this topic, in particular how (lexical) knowledge structures can provide complementary background knowledge in text mining scenarios. The chapter introduces and investigates two families of kernel functions, namely Semantic Smoothing Kernels (SSKs) and Semantic Syntactic Tree Kernels (SSTKs), that are capable of using such knowledge structures for alleviating the problems of variability in natural language. On the one hand, Semantic Smoothing Kernels build on the standard Vector Space Model (VSM) representation but can take the varying degrees of semantic similarity between individual terms into account. On the other hand, Semantic Syntactic Tree Kernels extend this concept towards representations that also mirror the syntactic structure of the textual input. For both types of kernels we incorporate the findings of Chapter 4 for choosing adequate notions of terms similarity based on the available lexical knowledge structures.

This chapter is organized as follows. We begin by reviewing the background for this chapter in Section 5.1. Specifically, this section introduces the Vector Space Model which constitutes the basic representation paradigm in text mining. It then analyzes some of the deficiencies of the VSM model which in turn motivate the design of alternative kernel functions. Section 5.2 investigates a family of kernel functions, referred to as Semantic Smoothing Kernels, which take build on the VSM representation but can take the variability of language into account. This section introduces the formal model, gives examples and provides an analysis of the properties of SSKs. While SSKs explicitly consider semantic aspects, they still neglect any syntactic relationships within the input texts. This aspect is taken up in Section 5.3 by developing a family of kernels called Semantic Syntactic Tree Kernels which incorporate the concepts of semantic smoothing into tree kernels, a class of kernels designed to work directly on the syntactic structure of the textual input. Critical to both groups of kernels is the choice of semantic smoothing parameters. In Section 5.4, we therefore investigate the requirements for the smoothing parameters that are (i) effective by maintaining a useful interpretation and (ii) valid by ensuring that the resulting SSKs or SSTKs maintain the kernel property. For both types of kernel functions, the results from the previous chapter determine the choice of semantic smoothing parameters that can be used. Section 5.5 points to various trails of related work while Section 5.6 concludes with a short summary and a discussion of the presented concepts. We demonstrate the application of both types of kernels in an extensive set of experiments on different datasets in the subsequent chapter.

5.1 Text Mining and the Vector Space Model

By *text mining* we refer to the application of machine learning technology to the organization and analysis of textual data (Feldman and Dagan, 1995; Hotho et al., 2005), a field that has attracted considerable interest in the last decade. In their recent book, Feldman and Sanger (2006, own emphases) characterize this field as follows:

> "In a manner analogous to data mining, text mining seeks to *extract useful information* from data sources *through the identification and exploration of interesting patterns*. In the case of text mining, however, the data sources are document collections, and interesting patterns are found not among formalized database records but in the *unstructured textual data* in the documents in these collections."

Text mining distinguishes itself from other machine learning settings mainly along two dimensions: the *type of tasks* and the *type of data preprocessing*.

5.1.1 Text Mining Tasks

Text mining tasks relate to structuring textual content according to (i) its perception by human users, or (ii) its linguistic properties. Text mining thus naturally overlaps with other computer science disciplines that deal with the processing of natural language such as information retrieval (Baeza-Yates and Ribeiro-Neto, 1999), web mining (Chakrabarti, 2002) as well as information extraction and corpus-based computational linguistics (Manning and Schütze, 1999).

Text mining tasks can be classified according to the *instance type* considered. For some tasks the items of interest are larger text units such as whole documents, paragraphs or sentences. Examples are *text classification (TC)* (Sebastiani, 2002) or *text clustering* which automate the classification or grouping of texts into sets of thematic or functional categories (Sebastiani, 2002). For another group of tasks, the items of interest are single or multiple sub-sentence expressions such as phrases, expressions or single words. Here, examples are Part–of–Speech (POS)-tagging, term clustering, e.g. for the automatic induction of concept hierarchies (Cimiano et al., 2005), semantic role labeling (Giuglea and Moschitti, 2006) or information extraction (Manning and Schütze, 1999). Along another line, text mining tasks can be distinguished by the *descriptive features* to be exploited as input. In the case of larger text units, it is common to use the text itself as the main source of information. In the case of sub-sentence expressions, it is common to use the surrounding context, e.g. the remaining text within a sentence or text windows of a predefined width as input. Finally, tasks can also be distinguished according to their *purpose*. Many classical text mining tasks such as text classification, text clustering or text summarization aim at the *organization* of textual documents as a classical knowledge management task (Marwick, 2001). Other tasks like information extraction (IE) or ontology learning (OL) (Maedche and Staab, 2001; Maedche, 2002; Cimiano, 2006) aim at the extraction of structured knowledge from unstructured textual input.

Text mining tasks need not be necessarily designed to operate in isolation. Often, they are part of a complex processing pipeline. As an example, consider the case of question classification (QC) (Metzler and Croft, 2005) that will be of interest as an evaluation scenario in Chapter 6. The QC task is a text classification task that aims at detecting the semantic type of the sought-after answer to a natural language question, e.g. whether it asks for a person or for an organization. The resulting classification information is then exploited for locating and extracting the right answers in question answering (QA) systems which requires further processing.

5.1.2 Fundamentals of the Vector Space Model

Text mining distinguishes itself from conventional data mining primarily by a set of specific preprocessing operations. These operations are responsible for transforming the unstructured textual input into a vector representation. Of course, in a kernel-based setting, some of these steps can alternatively be shifted into the kernel function and we will discuss some of these approaches later on.

Textual data comes with an intuitive feature representation, the *Vector Space Model representation*, sometimes also referred to as the *Bag-of-Words (BOW) representation*, that has been used early information retrieval (IR) (Salton et al., 1975). This representation builds on the hypothesis that the meaning of a text fragment can be approximated via the individual words it contains. Each word that occurs in the training collection thus becomes a feature. The resulting representation of a document is consequently referred to as a *word vector*. Widdows (2004) provides a good and comprehensive overview over the main rationales behind this representation. The main design decisions that affect the setup of the feature vectors in this paradigm are captured by two questions: (i) Which lexical units should be considered as features? (ii) What numerical values should be assigned to the dimensions of the feature vector such that they adequately mirror the importance of the respective lexical units in the textual input? In the following, we sketch the basic steps towards the VSM representation by answering both questions and name some of the most popular variants. Throughout this chapter, we denote the lexical units (terms) as $l_1, l_2, \ldots \in \mathcal{L}$ and the VSM vectors of textual inputs as $x, z, \ldots \in \mathcal{X}$.

Term Extraction

The *term extraction-* or *indexing* phase focuses on identifying the relevant lexical units that make up the dimensions in the word vector space. The resulting set of terms $l_1, l_2, \ldots \in \mathcal{L}$ is stored in a *lexicon* or *term dictionary*, usually together with basic occurrence statistics.

Assuming that the textual input is available as plain text, characters are usually converted to lower case in a first step. The aim of *tokenization* is then to determine adequate word boundaries and to split the text into chunks of terms. In the simplest case, tokenization rules only need to spot split characters such as whitespaces and punctuation marks. Depending on the language, the tokenization rules may however be more subtle. For example, in the English language a term may be a single word or a linguistic expression composed of multiple words that has a distinct meaning. Composite terms are usually referred to as *multi-word expressions* or *collocations* and need to be detected by advanced rules, usually by comparison with a reference dictionary.

Not all words necessarily contribute to the perceived document content. Such words, especially function words like articles and prepositions are called *stopwords* and are usually removed. For example, for the English language, a list of 571 stopwords which has originally been developed for the SMART system is commonly used.[1] Similarly, numbers are often removed as well as part of the stopword removal step.

In most languages, the same word may appear in different morphological variants. Most of these variations are inflections and reflect the contextual situation of a word in the sentence like plural forms or cases. *Lemmatization* is the activity of reducing words to their word stems. It is common to use language-specific lemmatization dictionaries or a set of advanced reduction rules. In an alternative paradigm called *stemming*, the reduction to base forms is governed by coarse grained heuristics which approximate a proper lemmatization. As an example, by virtue of simple morphological rules, most words in the English language can be stemmed easily by stripping off common suffixes. Porter (1980) has devised a popular and successful stemming algorithm for the English language based on this observation.

Example 5.1 (VSM Preprocessing)**.** Consider the preprocessing of the sentence *"SAP agrees 4.8-billion-euro buyout of Business Objects"*.

[1]The SMART stopwords list is available at `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`.

- Tokenization and lowercase normalization would yield the term set {*"sap"*, *"agrees"*, *"4.8"*, *"billion"*, *"euro"*, *"buyout"*, *"of"*, *"business"*, *"objects"*}.
- The entries *"4.8"* and *"of"* would be removed as stopwords.
- The lemmatization step would alter the verb entry *"agrees"* and, assuming that no special handling of collocations and named entities is in place, the noun entry *"objects"*.
- The resulting bag of term features would thus become {*"sap"*, *"agree"*, *"billion"*, *"euro"*, *"buyout"*, *"business"*, *"object"*}.

Term Weighting

In the simplest case, feature vectors contain the binary indicator variables '1' or '0' to indicate the presence or absence of a term in some text item. This representation has its roots in early approaches of information retrieval where search queries were seen as logical expressions and a document was considered a result candidate if its assignment of Boolean variables to the terms made the expression true. The major drawback of this representation is that all terms that occur in a document are considered equally important.

Weighting schemes aim at addressing this issue. For example, the vector entries may correspond to the absolute term frequencies in the respective document. However, using absolute term frequencies seems intuitive but has empirically produced suboptimal results. The reason is that common words which tend to occur in many documents are not well suited for discriminating documents in term space. Various advanced weighting schemes have been proposed to account for this situation. The most popular approach which has empirically shown superior performance is to inversely relate the term weight to the number of documents in which the term occurs by discounting the absolute term frequencies with a so called *Inverse Document Frequency (IDF)* resulting in the combined *Term Frequency Inverse Document Frequency (TFIDF)* weight (Salton, 1989).

Definition 5.1 (Term Frequency Inverse Document Frequency)**.** Given a set $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ of terms in the reference corpus and a set $\mathcal{X} = \{x_1, x_2 \ldots x_m\}$ of documents, let $\mathrm{tf}(l_i, x_j)$ be the absolute frequency of term l_i in document x_j. Then the function $\mathrm{tfidf} : \mathcal{L} \times \mathcal{X} \mapsto \mathbb{R}$ with:

$$\mathrm{tfidf}(l_i, x_j) = \mathrm{tf}(l_i, x_j)\ \mathrm{idf}(l_i) = \mathrm{tf}(l_i, x_j)\ \log \frac{|\mathcal{X}|}{|\{x_k \in \mathcal{X} | x_k \,\mathrm{contains}\, l_i\}|}$$

denotes the *TFIDF score* of term l_i in document x_j.

Extensions of the TFIDF scheme incorporate document length normalization to account for different lengths of input documents, e.g. by means of the mathematical vector normalization. In a kernel-based setting, this effect can be achieved implicitly by means of the cosine kernel modifier (c.f. Proposition 2.13). A study on the effects of various variants of these term weighting schemes in TC settings was conducted by Leopold and Kindermann (2002).

Since its early use by Salton et al. (1975) and others, the VSM representation has found widespread use in information retrieval and text mining. It still constitutes the predominant choice for most text mining tasks and the main source of this success is certainly the good performance in various text mining tasks at an appealing level of simplicity. However, various caveats can be made about the comparatively simple modeling of natural language phenomena in the VSM representation which we will review in the following together with an analysis of the impact of these deficiencies.

5.1.3 Deficiencies of the Vector Space Model — Semantic Aspects

The basic assumption behind the VSM representation is that it yields a "semantic" space in which geometric patterns can express regularities which correlate well with the perceived meaning of the input texts. In this space, each term contributes independently to the overall meaning of a document. Technically speaking, this representation implies that different terms correspond to mutually orthogonal dimensions in the resulting vector space.

The adequacy of this representation is challenged by situations in which two textual inputs, although perceived as conveying the same or similar content, only share a small amount of terms or no terms at all. In the following, we will approach the roots of this situation, historically termed the *vocabulary mismatch problem*, in terms of lexical semantics (Cruse, 1986).

Terms vs. Term Senses

According to the viewpoint of lexical semantics, words denote things or concepts in the world. This notion is best reflected by the well known *semiotic triangle* or *meaning triangle* of Ogden and Richards (1923).[2] Unfortunately, there is no one-to-one corre-

[2]The semiotic triangle is part a theory that explains the interconnections between symbols (e.g. words), concepts (senses) and referents (real world things and phenomena). According to this theory, symbols, e.g. verbal expressions in different languages, are used for communication. These symbols are not at all identical with concepts (thoughts) or things (referents) themselves. However, symbols are used to refer to concepts, thereby evoking the associated thought on the one hand. The evoked concept then, as we have said before, refers to a whole group of things. Thus the connection between a word and a thing is indirect as there is not necessarily any observable or direct relationship between a symbol and a certain thing (referent). Whether and how this link can be completed

spondence between the individual lexical units which we find in natural language and the meaning they are supposed to convey. Along this line, two problems are specifically apparent and relevant for the VSM representation, namely the phenomena of *synonymy* and *lexical ambiguity*.

Synonymy refers to situations where two or more different terms carry an identical meaning. In a strict sense, two expressions are considered *synonymous* if one can mutually substitute the two expressions in any sentence without changing the truth value of that sentence. As a classical example consider the synonymous terms *"car"* and *"automobile"*. In a looser sense, two expressions can be considered *synonymous within a certain context*, if there exists a context such that the mutual substitution in this context does not alter the truth value. The representation of texts via sets of terms in the VSM paradigm may pull the contribution some senses apart and distribute it across its different lexical representations. Note that this problem already exists with basic spelling variants (e.g. *"organization"* vs. *"organisation"* which a human reader would not even consider to be different terms.

On the other hand, lexical ambiguity refers to the converse phenomenon, namely that one and the same term or expression may have different meanings, usually in different contexts. Humans have the ability to differentiate between the different senses as part of the cognitive process of natural language understanding. Psychological studies suggest that this is done by relating the word to the discourse context, which may, however, apparently be quite small (Ide and Véronis, 1998). In the context of the VSM, lexical ambiguity tends to depresses the quality of the representation as different senses are conflated to a common dimension in feature space. In practice, the impact of ambiguous terms may, however, vary. On the one hand, lexical ambiguity becomes manifest in situations of *homonymy*, i.e. in situations where the various senses of a word are really distinct. In contrast, situations of *polysemy* are characterized by a set of possible senses for a given term that are, however, related because they share a common origin.

Relation among Term Senses

The problems we have mentioned so far mostly relate to the inadequacy of the VSM representation with regard to the lexical surface forms. As a more subtle variant of synonymy we also need to look at the similarity among *senses*. Psychology and linguistics have extended the concept of synonymy on the term level with a weaker form of the concept, namely *semantic similarity*. According to this view, senses are related to one another according to various degrees of semantic similarity and for the indication of a certain topic or the description of a certain situation, a wide set of concepts may appear relevant. In fact, native speakers tend to actively exploit the

depends on the interpreting process of both sender and receiver of the symbol.

offered variability in the choice of their wording. Even in a restricted scenario in a study by Furnas et al. (1987), two people choose the same main keyword for a single well-known object less than 20% of the time.

A common notion is that semantic similarity of senses mainly relates to the (linguistic) contexts in which words occur (Harris, 1968; Charles, 2000). At the same time, this perception primarily correlates with the amount of common "semantic content" by means of common *generalizations*. Generalizations cognitively organize concepts, i.e. word senses, along increasing levels of generality. Basic concepts at the highest levels of these structures are distinguished as elementary units that maximize cognitive economy while lower level concepts refine these basic categories (Rosch et al., 1976). In linguistics, generalization is captured in the notion of *hypernymy* as a semantic relation between concepts. Intuitively, hypernymy is asymmetrical whereby its inverse relation is referred to as *hyponomy*. Furthermore, hypernymy is transitive thereby generating a hierarchical semantic structure. Studies indicate that many of the semantic similarity measures which are based on the hypernym–hyponym structure of lexical taxonomies like WordNet (i.e. measures of the type we have encountered in Chapter 4) correlate well with the similarity perception by human subjects (Budanitsky and Hirst, 2006, 2001).[3]

In the following, we continue our previous example to illustrate the impact of semantic similarity on the VSM representation.

Example 5.2. Consider again the sentence *"SAP agrees on 4.8-billion-euro buyout of Business Objects."* we have introduced in Example 5.1 and another sentence *"IBM announces acquisition of its PC division by Lenovo."*. Assuming that both documents belong to the positive class of a classification problem aiming at the detection of financial news on acquisitions in the IT sector, the distinct though highly similar terms *"acquisition"* and *"buyout"* would be the sole indicators of this class.

Practical Impact

Despite these shortcomings, text classification systems have achieved good generalization results using the basic VSM representation or variants thereof in many settings. This is particularly true in text classification and text clustering tasks that aim

[3]In a strict sense, we only cover the concept of *literal similarity*. This can be distinguished from iconic, analogical, or connotative similarity which do, however, not impact the vocabulary mismatch problem in a way as literal similarity does. Along another line, Resnik (1999) has made a good point in distinguishing the more general concept of *semantic relatedness* from *semantic similarity*. The notion of relatedness can encompass various kinds of relationships other than common hypernyms, e.g. meronymy, antonymy or functional association. Consider the example given by Resnik to demonstrate this distinction: *"car"* and *"gasoline"* might seem to be more closely related than *"car"* and *"bicycle"* but the latter pair would certainly be more similar.

at the automated dispatching of text documents along their primary topics (Joachims, 1998; Yang, 1999).

A common intuition is that text classification on the basis of the pure VSM representation works well in those cases where the categories of interest can be characterized by the use of distinctive keywords and where sufficient data is available to recognize them during training. Joachims (2001) has theoretically analyzed this behaviour in a formal model of text classification with Support Vector Machines (SVMs). The most relevant findings of this analysis are as follows. On the one hand, the analysis shows that there is usually a large number of keywords that, with varying degrees of occurrence frequencies, can be used as class indicators. On the other hand, the considered text mining tasks are characterized by redundancy, i.e. usually several indicative words occur together in a document. In line with common intuition, Joachims (2001) shows that the expected error can be bounded in terms of several characteristics of the dataset. In particular, the bound becomes tighter, i.e. it suggests better generalization capabilities, if (i) the discriminative power of groups of terms increases, (ii) discriminative features occur more frequently and (iii) the level of redundancy increases.

A crucial result of this analysis is, however, that the bounds may loosen in situations of insufficient training data. On the one hand, an insufficient number of training instances introduces a higher level of variance both for the occurrence frequencies of term sets and their discriminative power. The smaller number of stable cues also affects the level of redundancy.

In summary, in those cases where training data is scarce and/or sparse, a more adequate data representation should reduce the variability and lead to patterns which are more stable than those based on the pure VSM representation. The approach we will take in the following is based on complementing the VSM representation by a-priori knowledge about the semantic similarities of terms by means of a specifically designed kernel function. We thereby perform a kind of "semantic smoothing" that alleviates the problems of the heterogeneity in the use of terms.

5.1.4 Deficiencies of the Vector Space Model — Syntactic Aspects

Up to now, we have covered the question in how far the lack of information on the mutual relation between terms affects the VSM model. Except for this shortcoming, we have implicitly accepted the validity of the VSM in which terms or senses are treated as the main constituents of the overall meaning of a text fragment.

Considering Syntactic Aspects

In natural language sentences, terms do not occur in isolation. The regularities in word order and phrase structure are governed by syntactic rules. The syntactic structure of natural language sentences is usually described according to the formal *grammar* of the respective language. The grammar provides a precise description of how terms as basic units of language can be transformed and combined to yield valid natural language sentences (Allen, 1995). Hereby, syntactic structures can be associated with the meaning of the overall sentences.

Example 5.3. Consider the two text snippets (i) *"They named Reims as an interesting tourist attraction."* and (ii) *"What are the names of tourist attractions in Reims?"*. Ignoring stopwords and morphology, both texts would be represented almost identically but obviously have distinct individual meanings.

Depending on the context, different syntactic patterns may be particularly relevant for a given text mining problem. Typical examples are phrases that group words into larger meaning-bearing units. As an example consider the noun phrase *"tourist attractions in Reims"* in the second sentence in the example above consisting of the head noun *"attractions"*, the modifier *"tourist"* and its prepositional phrase *"in Reims"*. Similarly, verb phrases are important meaning bearing units. In the first sentence in the above example, the verb *"to name"* takes *"Reims"* as its object. But also collocations, phrasal verbs or other expressions that are usually not considered individually in the VSM model may contribute to a distinct meaning of two sentences (Allen, 1995; Manning and Schütze, 1999).

Practical Impact

For scenarios of classification and clustering based on the overall document topic, the isolation of keywords in VSM-type representations appears to be sufficient. For example, while the two sentences in Example 5.3 have distinct meanings, their VSM representations would probably suffice if the task was to find classification rules for the topic *"tourism in France"*.

On the other hand, structural patterns may be highly relevant in more advanced tasks. They are particularly relevant when the text mining task is to find common patterns for constituents or other parts of the sentence. For the case of Example 5.3, the syntactic structures intuitively make a difference in a classification task that aims at detecting qualitative statements about cities. Other examples of tasks for which syntactic structure as a source of information has proved to be indispensable are semantic role labeling (Giuglea and Moschitti, 2006) or question classification (Metzler and Croft, 2005).

5.2 Semantic Smoothing Kernels

In the preceding section, we have argued that the representation of textual input according to the VSM does not adequately take into account various semantic and syntactic details of natural language. In particular, the VSM representation in conjunction with plain dot products suffers from the heterogeneous use of terms as it ignores knowledge about the pairwise semantic similarities of terms.

The aim of this section is to show how such knowledge, as for example encoded in the topological relations within semantic structures like WORDNET, can be incorporated within kernel functions. This allows the learning algorithm to relate distinct but similar features during kernel evaluation. In this model, we still assume the basic validity of the representation of a textual input as a vector of terms. That is, a representation that views isolated terms as basic units of feature representation, regardless of their mutual positions and relations within the text. As argued earlier, apart from the semantic deficiencies, this assumption is usually not critical for topical classification or clustering of textual data items. We will first introduce the general concept of SSKs and then investigate its properties.

5.2.1 A General Model for Semantic Smoothing Kernels

Based on the setting of the VSM model and our general characterization of kernel functions in Section 2.4, we are now able to introduce the concept of Semantic Smoothing Kernels.

Definition 5.2 (Semantic Smoothing Kernel)**.** The Semantic Smoothing Kernel for two VSM vectors $x, z \in \mathcal{X}$ is given by

$$\kappa(x,z) = x'\mathbf{S}z = \sum_{i=1}^{|\mathcal{L}|}\sum_{j=1}^{|\mathcal{L}|} x_i z_j \mathbf{S}_{ij}.$$

Whereby $\mathbf{S}$ is a $|\mathcal{L}| \times |\mathcal{L}|$ square symmetric and positive semi-definite matrix.

Hereby, the entries $\mathbf{S}_{ij}$ represent the semantic similarity between the terms l_i and l_j, i.e. the respective dimensions of the input space $\mathcal{X}$ of dimensionality $|\mathcal{L}|$.

Siolas and d'Alche Buc (2000) for the first time considered a Gaussian kernel function on text documents that effectively used a similar setup like SSKs for distorting the Euclidean distance between the original term vectors. Besides our work, Basili et al. (2005a) and Mavroeidis et al. (2005b) have recently considered kernel functions of this type. In the following exposition, we aim provide a principled and comprehensive analysis of these kernels while elaborating on other work in this direction in Section 5.5.

Validity Requirements

The requirement for $\mathbf{S}$ to be a positive semi-definite matrix ensures that the overall kernel functions retains validity. This finding is summarized in the following proposition.

Proposition 5.1 (Positive Semi-definiteness of Smoothing Matrices)**.** *For any square symmetric matrix* $\mathbf{S}$*, a function of the form* $\kappa(x,z) = x'\mathbf{S}z$ *is a valid kernel if and only if* $\mathbf{S}$ *is positive semi-definite.*

Proof. First consider the case of a given positive semi-definite smoothing matrix $\mathbf{S}$. Then, according to Definition 2.15, $\mathbf{S}$ has only non-negative eigenvalues and can thus be decomposed as $\mathbf{S} = \mathbf{P}'\mathbf{P}$. Consequently, we have that $\kappa(x,z) = x'\mathbf{S}z = x'\mathbf{P}'\mathbf{P}z = \langle \mathbf{P}x, \mathbf{P}z\rangle$, thus verifying that the kernel function has a representation as a dot product and is thus indeed positive semi-definite.

Now conversely assume that $\kappa(\cdot,\cdot)$ is a valid kernel, but $\mathbf{S}$ is not positive semi-definite. In this case, $\mathbf{S}$ has at least one negative eigenvalue. Let $\lambda < 0$ be this eigenvalue and let $x \neq \mathbf{0}$ be the corresponding eigenvector. We then have $\kappa(x,x) = x'\mathbf{S}x = \langle x'\mathbf{S}, x\rangle = \langle \lambda x, x\rangle = \lambda\langle x, x\rangle < 0$, thereby clearly contradicting the positivity property of inner products (compare Definition 2.8). Thus, $\kappa(\cdot,\cdot)$ can not be a valid kernel function. □

As the matrix $\mathbf{S}$ is positive semi-definite, we can equally view its components as evaluations of a kernel function $\kappa_{\mathcal{L}}(\cdot,\cdot)$ itself. This kernel function is then defined on the set $\mathcal{L}$ of lexical units (terms), i.e. $\kappa_{\mathcal{L}}(l_i, l_j) = \mathbf{S}_{ij}$. As such, assuming that terms can be mapped to entities in lexical knowledge structures such as WORDNET, all (taxonomic) similarity functions which constitute positive semi-definite functions can be used as smoothing parameters. We will discuss the choice of appropriate smoothing parameters in Section 5.4 while the experimental analysis of the suitability of these measures will be the topic of Chapter 6. For the general analysis in this section, it is sufficient to assume that the supplied smoothing parameters are valid (i.e. positive semi-definite) and encode a useful notion of semantic similarity.

Word Sense Ambiguity Issues

We have argued in Section 5.1 that lexical similarity holds between word senses, not between terms. Strictly speaking, a smoothing matrix $\hat{\mathbf{S}}$ based on lexical knowledge structures would thus encode the similarities among a set of concepts $c_1, c_2, \ldots \in \mathcal{C}$. The disambiguation of word senses is the topic of *word sense disambiguation* (Ide and Véronis, 1998). It would of course be possible to conduct the word sense disambiguation as part of the preprocessing phase and thus just work with *sense vectors*.

In the following we will maintain a somewhat simplified view on the issue. Specifically, we assume the existence of a somewhat informal disambiguation operator $\delta : \mathcal{L} \times \mathcal{C} \mapsto 0, 1$ to map from terms to one or several concept representations. Such an operator can just as well be written as a 0/1 matrix $\mathbf{D}$ of dimension $|\mathcal{C}| \times |\mathcal{L}|$ whereby $\mathbf{D}_{ij} = 1$ indicates that the concept c_i has a lexical representation in the term l_i. The actual smoothing matrix is then assembled as $\mathbf{S} = \mathbf{D}'\hat{\mathbf{S}}\mathbf{D}$. Note that disambiguation usually requires consideration of the occurrence context of a word such that this type of disambiguation is clearly a simplifying assumption. Disambiguation heuristics which could be implemented in this form would be (i) the association of a given term with all matching concepts or (ii) the association of a given term with its most frequent sense in a lexical taxonomy. In the experiments in Chapter 6 we will solely consider the second case. This decision is on the one hand rooted in practical considerations and the observation that even state-of-the-art word sense disambiguation systems yield substantial error rates (Ide and Véronis, 1998) and thus might introduce random errors. On the other hand, it reflects the finding that a large fraction of lexical ambiguity can be attributed to cases of polysemy, not homonymy. As an example consider the two most prominent senses for the term *"book"*, according to WORDNET (Miller et al., 1990; Miller, 1995): (i) *"(book) — a written work or composition that has been published (printed on pages bound together)"* and (ii) *"(book, volume) — physical objects consisting of a number of pages bound together"*. Intuitively, these senses are highly related, and either one is likely to be a proper representative of the intended meaning in a textual input.

5.2.2 Interpretation of Semantic Smoothing Kernels

So far, we have discussed the formal adequacy of the kernel. We will now have a closer look at its *interpretation*. In summary, we will see that SSKs implicitly perform a linear transformation of the input data which aims at providing better representation. There are two main perspectives along which the results of the Semantic Smoothing Kernel can be interpreted.

Considering Contributions of Similar Terms

The first perspective becomes clear when looking at the right side of the SSK formula in Definition 5.2 and comparing it with the definition of the ordinary inner product. While the inner product is the sum over products of the term weights of *identical* terms in the two argument vectors, the SSK additionally considers contributions of all pairs of *distinct* terms. However, the amount of the contribution of a given pair of terms to the overall kernel value is governed by the factors $\mathbf{S}_{ij}$, i.e. the similarity of the corresponding terms. In some cases, these factors may be small or even zero such that they do not change the original result very much. On the other hand, for pairs of highly similar terms, they may alter the results substantially. This interpretation characterizes the altered text similarity most intuitively.

Example 5.4 (SSKs – Interpretation)**.** Consider three simple VSM vectors $x' = (1,0,0)$, $y' = (0,1,0)$, and $z' = (0,0,1)$ Clearly, the inner product evaluations between these vectors yield values of $\langle x,y \rangle = \langle x,z \rangle = \langle y,z \rangle = 0$. Now consider the smoothing matrix

$$\mathbf{S} = \begin{pmatrix} 1 & 0.75 & 0 \\ 0.75 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

According to this smoothing matrix, the first and the second term are related by a factor of 0.75. These could, for example be the two terms *"acquisition"* and *"buyout"* in the dataset introduced in Example 5.1. In contrast, some other third term, is not related to the first two. Here, the evaluations of the SSK yield values of $\kappa(x, y) = 0.75$ on the one hand and $\kappa(x, z) = \kappa(y, z) = 0$ on the other. Although x and y employ different terminology, their terminology is still similar such that $x_1 y_2 \mathbf{S}_{12} = 0.75$ produces a positive contribution during kernel evaluation.

Implicit Feature Mapping

While this first perspective is intuitive, it does not directly reveal the feature space underlying the kernel function. A second alternative perspective becomes clear when looking at the reasoning in the proof of Proposition 5.1. Conceptually, it has shown that $\mathbf{S}$ can always be decomposed as $\mathbf{S} = \mathbf{P}'\mathbf{P}$. This decomposition reveals the underlying feature mapping as a linear transformation of the type $\phi(x) = \mathbf{P}x$. In general, the transformation matrix $\mathbf{P}$ may have $\mathcal{L}$ columns and any number m of rows, thus providing a linear transformation of the input document into a feature space of dimensionality m. Any such linear transformation can be thought of as a shifting of weights among dimensions or even the introduction of new dimensions. This approach is similar to what is known in IR as a *query expansion*. We will review some of these approaches in the context of related research work in Section 5.5.1.

Canonical Mapping Of course, for a given smoothing matrix $\mathbf{S}$ different matrices $\mathbf{P}$ may constitute valid decompositions and the resulting feature spaces may vary in dimensionality. However, despite the multitude of possible mappings for a given smoothing matrix $\mathbf{S}$, there is a canonical mapping in terms of the eigen-decomposition of $\mathbf{S}$ which mirrors the basic structure of all possible mappings and also defines the *effective dimensionality* of the resulting feature space. Let $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}' = \mathbf{V}\sqrt{\mathbf{\Lambda}}\sqrt{\mathbf{\Lambda}}\mathbf{V}'$ be the eigen-decomposition of $\mathbf{S}$. Then, setting $\mathbf{P} = \sqrt{\mathbf{\Lambda}}\mathbf{V}'$ we get an understanding of the fundamental transformation steps.

First, the left multiplication of the original VSM vectors by $\mathbf{V}'$ transforms the original VSM vectors into the eigenspace of $\mathbf{S}$. As $\mathbf{V}$ is an unitary matrix, this kind of operation only yields a rotation of the vectors, leaving inner products and distances between vectors unchanged. In the eigenspace, the multiplication by the diagonal matrix $\sqrt{\mathbf{\Lambda}}$ yields a reweighting of the new dimensions by the amount of the root of

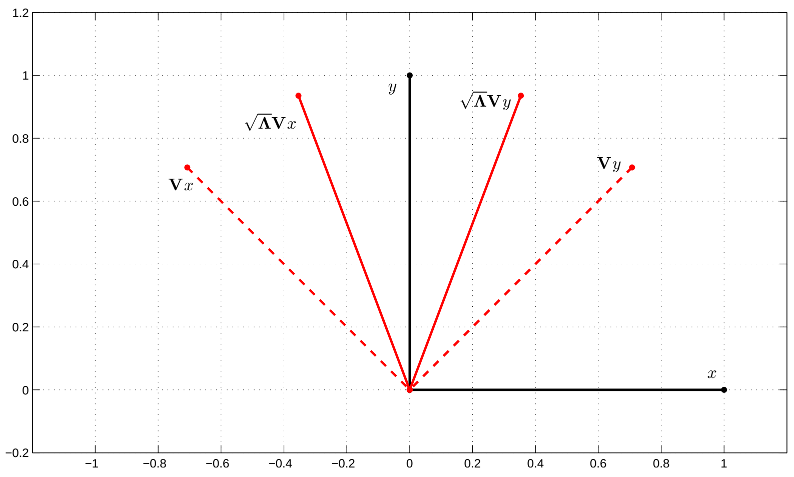


Figure 5.1 — Illustration of the eigenspace transformation for Semantic Smoothing Kernels in Example 5.5. The first transformation corresponds to a rotation of the VSM vectors, the second transformation to a rescaling of the new dimensions. The two vectors thus move closer together.

the respective eigenvalues. It is this second step that effectively distorts the original document similarities.

Example 5.5 (SSKs – Interpretation (cont.))**.** Recall the smoothing matrix **S** introduced in Example 5.4. For the purpose of visualization, we will look only on the first and the second dimension which stand in a similarity relation. The eigen-decomposition of the matrix yields the eigenvector, eigenvalue pairs:

$$\mathbf{V} \approx \begin{pmatrix} -0.71 & 0.71 \\ 0.71 & 0.71 \end{pmatrix} \quad \mathbf{\Lambda} = \begin{pmatrix} 0.25 & 0 \\ 0 & 1.75 \end{pmatrix}$$

The transformation of two vectors $x' = (1, 0)$ and $y' = (0, 1)$ by $\mathbf{V}$ yields the vectors $\tilde{x}' = x'\mathbf{V} \approx (-0.71, 0.71)$ and $\tilde{y}' = y'\mathbf{V} \approx (0.71, 0.71)$. Figure 5.1 illustrates the original vectors, as well as the situation after the rotation. Obviously, the two vectors are still orthogonal. However, the new dimensions have switched some of the weight. The square roots of the eigenvalues are given by $\sqrt{\lambda'} \approx (0.5, 1.32)$ and the rescaling of the dimensions in the rotated space, yield the new vectors $\hat{x}' = x'\sqrt{\mathbf{\Lambda}}\mathbf{V} \approx (-0.35, 0.94)$ and $\hat{y}' = y'\sqrt{\mathbf{\Lambda}}\mathbf{V} \approx (0.35, 0.94)$. Again, Figure 5.1 illustrates the resulting vectors which are now not orthogonal any more but have moved *closer* together such that their inner product becomes $\langle \hat{x}, \hat{y} \rangle = 0.75$ as computed above.

Linear Dependence of Dimensions A special situation arises in the case that the smoothing matrix $\mathbf{S}$ is not of full rank. For example, this situation can arise in the case of two terms being perfect synonyms. In this situation their respective rows and columns in $\mathbf{S}$ become identical and thus linearly dependent. In such cases, only $\text{rank}(\mathbf{S})$ eigenvalues of $\mathbf{S}$ will be positive, while the remaining ones will be equal to zero. In the light of the discussion above, this means that the transformation $\mathbf{P}$ will map two input dimensions entirely onto one dimension while the rescaling of the second dimension by zero will eliminate it entirely.

Interpreting Varying Degrees of Self-Similarity

Note that the setting we have discussed so far does not require that the entries in $\mathbf{S}$ are normalized to the $[0, 1]$ interval and in particular it does not require constant self-similarity (c.f. Section 4.1.1). As a result, entries on the main diagonal may vary.

To illustrate the effect of this situation, consider some positive semi-definite smoothing matrix $\mathbf{S}$ and let $\mathbf{S}_{diag}$ be a diagonal matrix obtained from $\mathbf{S}$ by keeping the entries on the main diagonal and setting all other entries to zero. We now have that

$$\sqrt{\mathbf{S}}_{diag}^{-1}\mathbf{S}\sqrt{\mathbf{S}}_{diag}^{-1} = \hat{\mathbf{S}} \text{ or, equivalently,}$$

$$\mathbf{S} = \mathbf{S}_{diag}\hat{\mathbf{S}}\mathbf{S}_{diag}.$$

Hereby, $\hat{\mathbf{S}}$ is again a normalized smoothing matrix and the diagonal matrix $\sqrt{\mathbf{S}}_{diag}$ can be seen as providing a *reweighting* of the original input features.

5.2.3 Complexity and Optimization

We now shortly analyze the complexity of SSKs in comparison with the conventional inner product on the one hand and with an explicit feature transformation as discussed in the previous section. Using basic multiplication as base operation, we will refer to the complexities in Big O notation, which describes the asymptotic upper bound of the worst case running time (Knuth, 1997). Of course the analysis in terms of the Big O notation means that we abstract away from any constant optimizations that can influence the run-time behaviour in practical settings.

Training Time

We assume a VSM representation in a vector space of dimensionality n and a set of m training instances. During training of a kernel-based learning algorithm, the kernel function has to be evaluated on all pairs of training instances. This activity is inherently quadratic in the number of training instances, i.e. in $O(m^2)$. The inner product is linear in the dimensionality, i.e. it requires $O(n)$ multiplication operations.

The resulting complexity of the plain inner product kernel thus amounts to $O(n\,m^2)$ as n and m grow.

We now turn to the case of SSKs. In the naive formulation of Definition 5.2, each kernel evaluation requires one vector-by-matrix multiplication which is in $O(n^2)$ and one vector-by-vector multiplication. The overall training complexity for SSKs thus requires $O(n^2\,m^2)$ operations as n and m grow. As we have seen above, instead of the naive computation of the SSK function, it is possible to perform the explicit feature transformation of the type $\phi(x) = \mathbf{P}x$. This leads to shifting the computational burden of vector-by-matrix multiplication to a one-time effort before training and working with the transformed instances afterwards. Similarly, it is possible to cache results of $\mathbf{S}x$, i.e. the first matrix-by-vector computation. In both cases, the complexity of the initial transformation is $O(m\,n^2)$ while the actual training now requires only $O(m^2\,n)$ operations as it is sufficient to perform a vector-by-vector multiplication for each pair of items by using the precomputed and cached version $\mathbf{S}x$ for one of the argument vectors. This yields a total complexity of $O(m\,n\,(m+n))$ as n and m grow.

Although the asymptotic worst case behaviour of both approaches, the plain dot product as well as SSKs, is in the same order of magnitude as n and m grow jointly, the situation is different for fixed training set sizes m, where the variation in the dimensionality still incurs a quadratic performance for SSKs compared to a linear variation for plain dot products. As a result, we note that the increased expressiveness of SSKs naturally incurs an overhead during training. However, as training is a one-time batch activity in operational settings and labelled training data is generally scarce, this finding will not affect the applicability of the kernel in most practical settings.

Test Time

We now turn the attention to the analysis of the complexity of SSKs compared to linear kernels during the execution of the trained classifier in an operational setting. At test time, the kernel functions need to be evaluated between a limited number of dual parameters, e.g. the support vectors in the case of SVMs, and each test data point. Note again that, each kernel evaluation requires one vector-by-matrix multiplication and a subsequent vector-vector multiplication. It is thus possible to perform the vector-by-matrix multiplication $\mathbf{S}x$ on the dual parameters in a one-time effort. In fact, as the basic transformation underlying SSKs is inherently linear, the linear discriminant function of Equation 2.4 in dual form can be optimized even more through the rearrangement:

$$\begin{aligned} f(x) &= \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \kappa(x_i, x) + b\right) \\ &= \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i (x_i' \mathbf{S} x) + b\right) \\ &= \operatorname{sign}\left(\left\langle \sum_{i=1}^{n} \alpha_i y_i \mathbf{S} x_i, x \right\rangle + b\right) \qquad (5.1) \end{aligned}$$

In either case, the initial computations require only a constant amount of time during initialization of the algorithm. However, each evaluation of a test data point now remains in $O(n)$, such that the overall complexity given m test data points is $O(m\,n)$ as if we had used the plain inner product. As a result, the initial overhead vanishes as more test data points are considered and has the same complexity as the plain dot products on the original VSM vectors thus making the use of SSKs feasible in practical settings.

5.2.4 Extensions

The basic model of Semantic Smoothing Kernels can be extended in various ways which we shortly discuss in the following.

On the one hand, as discussed in Section 2.4.2, it is possible to embed a Semantic Smoothing Kernel within a kernel modifier, e.g. a Gaussian kernel or a polynomial kernel to allow for an additional nonlinear mapping of the original input features.

On the other hand, it is possible to combine multiple SSKs by using the sum of their respective smoothing matrices. As a special case of this situation consider the combination of the traditional dot product on the VSM representation and some smoothing kernel:

$$\kappa(x, z) = \alpha\langle x, z\rangle + \beta x' \mathbf{S} z = x'(\alpha \mathbf{I} + \beta \mathbf{S}) z.$$

The combined kernel thus represents the original input vectors in their original VSM representation and in their transformed space at the same time without requiring any additional resources in terms of space or computation.

As a particularly interesting variant of this setting consider the case where the lexicon $\mathcal{L}$ explicitly indexes collocations, i.e. composite terms like *"machine learning"*. This is a desirable variant as it reduces word sense disambiguation problems and may yield more relevant similarity mappings (e.g. between *"machine learning"* and *"knowledge discovery"*). For backing-off the effect of the common BOW representation where collocations are not handled explicitly, we can design a "lexical" smoothing matrix which contains entries $\mathbf{S}^{lex}_{ij}$ which denote the absolute number of individual terms common to the lexical units l_i and l_j. The linear combination of this matrix $\mathbf{S}^{lex}$ and another semantic similarity matrix $\mathbf{S}$ at the same time combines collocation handling, the ordinary VSM representation and semantic smoothing.

5.3 Semantic Syntactic Tree Kernels

With Semantic Smoothing Kernels we have investigated a technique that allows us to exploit knowledge about semantic dependencies between terms within the traditional VSM paradigm. However, as we have argued earlier, the perception of natural language by humans draws not only from the employed terminology and the associated meanings but also from the syntactic relationship between terms. For example, the information whether a given noun resides in subject or object position of a particular verb or how it is modified by adjectives provides useful information about the overall semantics of the text fragment under investigation.

The basic VSM representation does not offer any means to account for the syntactic relations among terms. Of course, it is possible to manually design extensions of the basic VSM feature space that comprise additional syntactic features such as phrases, POS information and the like. However, the manual design of such syntax-oriented features is cumbersome and requires substantial experience (Moschitti and Basili, 2004). Furthermore, the number of such features can easily grow to a level that cannot be handled any more in a straightforward way.

As an approach to deal with such problems, *tree kernels* have been proposed as a means for implicitly working on large sets of syntactic features. Since tree kernels were proposed by Collins and Duffy (2002) they have proved to be a useful technique for quickly modeling syntactic information for various tasks in natural language processing such as syntactic parsing (Collins and Duffy, 2001), relation extraction (Zelenko et al., 2003; Zhang et al., 2008) and semantic role labeling (Moschitti, 2004). However, despite their success in these settings, tree kernels suffer from the lack of knowledge about semantic relationships in a similar manner as the VSM representation. Finding principled techniques for addressing issues of syntactic structure and lexical semantics at the same time is thus a promising research line.

The aim of this section is to investigate how concepts of semantic smoothing as encountered with SSKs can be incorporated into tree kernels. We first introduce a few basic notions about the representation of a text as a parse tree and then explain the basic model of tree kernels. We then introduce Semantic Syntactic Tree Kernels (SSTKs) as a technique for jointly exploiting syntax and semantics. We deduce the general model by motivating the feature spaces of SSTK, show how these kernels can be computed efficiently and then investigate their properties.

5.3.1 Context: Tree Kernels for Parse Trees

Tree kernels constitute a prominent example of kernel functions for structured data. As all kernel functions of this type, they bluntly discard the representation of their inputs as vectors. Rather than representing textual inputs as vectors like in the VSM representation considered so far, tree kernels work directly on the representation of the input sentences in terms of their *parse trees* which mirror their inner syntactic structure.

Parsing and Parse Trees

We have mentioned earlier that the syntactic structure of natural language sentences is described according to the formal grammar of the respective language (Allen, 1995; Manning and Schütze, 1999). The analysis of natural language texts by *parsing* aims at recovering their syntactic structure according to the chosen grammar paradigm. Parse trees are classical representations of the syntactic structure of natural language sentences for a given context-free grammar.

Various paradigms for natural language grammars exist whereby *phrase structure grammars* and *dependency grammars* constitute the most prominent examples. These grammars adhere to different representations but share the common goal of making the dependency relations between terms and phrases explicit. In the following, we will be occupied solely with the latter type, namely phrase structure grammars. The interested reader is referred to the study by Moschitti (2006) for a comparison of tree kernels based on the different parsing paradigms. Phrase structure grammars are, at least at their core, context-free, i.e. of Chomsky type 2. Parse trees are a convenient representation of natural language sentences with respect to the grammar. However, a problem of all realistic grammars is that for a given token more than one production may be possible, thus resulting in a conflict during parsing. Most modern parsers are therefore at least to some extent of statistical nature. Relying on a corpus of training data annotated with proper parses, they exploit information about the probabilities of productions within a given context (Manning and Schütze, 1999, and references therein).

As an example, Figure 5.2 shows a parse tree of the question sentence *"What are the names of the tourist attractions in Reims?"*. The interior tree nodes of a parse tree are labeled with *non-terminal symbols* of the grammar such as sentences, sentence fragments, phrases and POS categories like nouns or verbs. For the non-terminal symbols, we use the commonly used identifiers of the Penn Treebank Project (Santorini, 1990; Bies, 1995) which are also summarized in Appendix B. These nodes, together with their children are associated with a grammar production rule such as `[NP` $\rightarrow$ `[DT NN NNS]]`. On the other hand, the leaf nodes are labeled with *terminal symbols* of the grammar which correspond to the actual terms. We will also refer to the direct

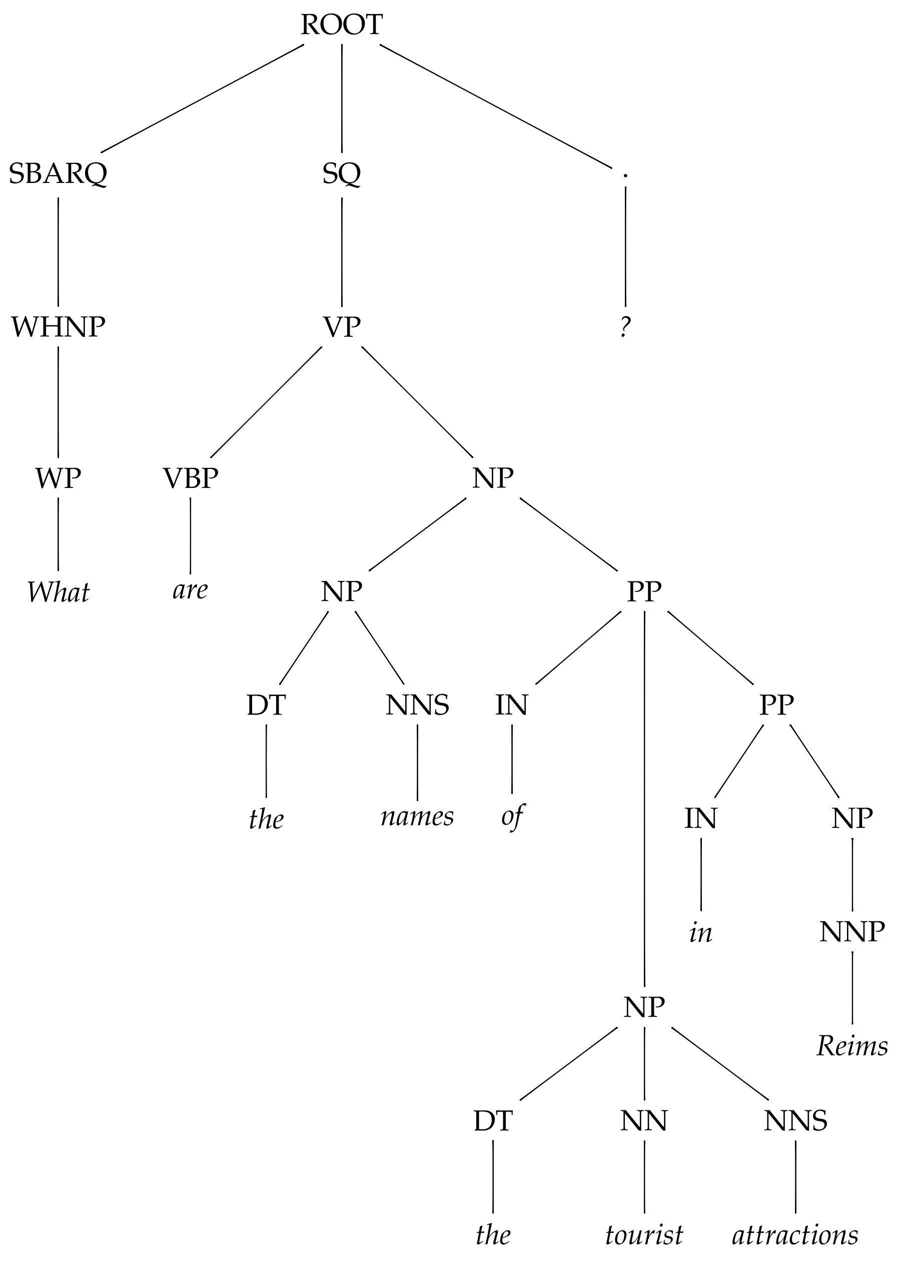


Figure 5.2 — Example of a parse tree for the question *"What are the names of the tourist attractions in Reims?"* according to the Penn Treebank tagset (c.f. Appendix B).

parents of leaf nodes, i.e. their POS categories, as *pre-terminal symbols*. The different fragments of the parse tree capture information about the overall meaning of the sentence under investigation such as the information that the prepositional phrase *"in Reims"* extends the basic noun phrase *"the tourist attractions"*. The exploitation of all this information during learning is the main motivation behind tree kernel functions.

Definition 5.3 (Parse Tree)**.** A parse tree a rooted and ordered tree structure whose nodes are labelled by the symbols of the chosen grammar. Parse trees are denoted as $t_1, t_2, \ldots \in \mathcal{T}$ and the set of nodes in tree t_i is denoted as $n_1, n_2, \ldots \in \mathcal{N}(t_i)$. We denote the *label* of an arbitrary node n_i as $\text{label}(n_i)$. As usual, we define a tree as a connected graph with no cycles. For each tree t_i, a particular node, denoted $\text{root}(t_i)$, takes the role of a designated root and is labelled by the start symbol of the corresponding grammar. Edges within the tree thus have a natural orientation, away from the root. Nodes that can be reached from a given node along this direction via a single edge are called its children. Figure 5.2 shows the example of a typical parse tree. For interior nodes these labels correspond to the non-terminal symbols of the grammar while for leaf nodes they correspond to the actual terms. We use the notation $\text{child}(n_i)$ to refer to the ordered set of children of an interior node n_i, i.e. the right side of the corresponding grammar production rule. Hereby $\text{child}(n_i, k)$ denotes the k-th child of node n_i.

Tree Kernel Feature Spaces

We now introduce the basic model of tree kernels, mostly in the spirit of Collins and Duffy (2002). The basic idea behind tree kernels is to represent trees in terms of substructures (fragments) of the argument trees. We first formalise this notion.

Definition 5.4 (Tree Fragments and Equivalence)**.** The set of connected subsets of nodes of a given tree t_i is referred to as the set of *tree fragments* and is denoted $f_1, f_2, \ldots \in \mathcal{F}(t_i)$. Depending on the chosen tree kernel paradigm we will sometimes impose additional constraints on the structure of these fragments. Again, the topmost node of a given tree or tree fragment t_i is denoted $\text{root}(t_i)$. Two trees or tree fragments f_i and f_j are said to *match* if they are isomorphic to each other with respect to their labels and their connections and we denote this equivalence as $f_i \sim f_j$. For a given set of trees this relation thus induces a set of equivalence classes of tree fragments which we denote as $f_1^\star, f_2^\star, \ldots \in \mathcal{F}^\star$.[4]

Tree kernel functions implicitly represent tree structures in a n-dimensional vector space whereby each dimension corresponds to a particular equivalence class of tree

[4]Note that the indices of the tree fragment equivalence classes correspond to some arbitrary but fixed numbering. In particular, $f_i^\star$ needs not be identical with the class of tree fragments equivalent with a particular fragment f_i.

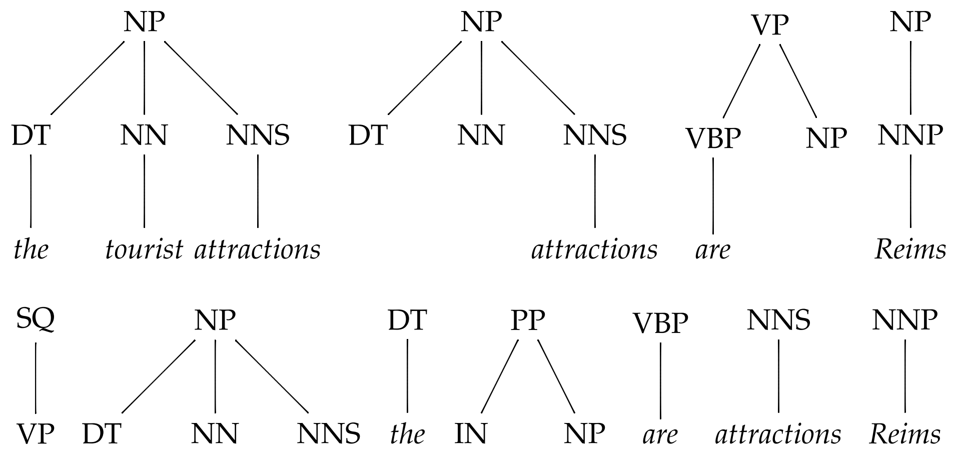


Figure 5.3 — Sample of substructures of the parse tree for the sentence *"What are the names of the tourist attractions in Reims?"* in Figure 5.2.

fragments. For each fragment class, the respective vector entry denotes the number of times fragments of this class occur in the tree, i.e. the entry will be positive if the tree contains such a fragment and it will be zero otherwise. Similar to the VSM for words, tree kernels thus generate hypotheses in a high-dimensional vector space and the implicit feature representations is very sparse.

Different kernel functions have been proposed that impose different restrictions on the type of fragments that are considered. As an example, Vishwanathan and Smola (2003, 2004) consider only proper subtrees, i.e. substructures that are characterized by expanding an arbitrary internal node of the original tree until the leaf nodes. As an alternative, the original proposal by Collins and Duffy (2002) does not require the full expansion up to the leaves. However, it requires that if an expansion takes place, the grammar production rule must be executed fully at the respective level. On the contrary, the partial tree kernel proposed by Moschitti (2006) considers an even bigger class of fragments by relaxing the requirement that the right side of a production rule must be fully represented in the fragment.

In the remainder, we will focus on the second setting above, i.e. the original proposal by Collins and Duffy (2002) which has empirically produced the best results. Figure 5.3 shows some of the resulting substructures for the example parse tree of Figure 5.2 on page 111. We will return to this issue in Section 5.3.5 where we discuss variants of the basic setting.

Computation of Tree Kernels

Naturally, when studying how to define appropriate kernel functions, trees need not be explicitly represented as vectors of the predefined features. Instead, the dot product of the respective feature vectors is computed directly from the two argument trees.

Let us for a moment assume that in the feature vectors each tree fragment class would be represented by the absolute number of matching fragments in the argument tree. The dot product could then be interpreted as the number of matching tree substructures in both argument trees and the task of the tree kernel function would be to efficiently count the number of these matches. Note that the number of such fragments of a single tree can be obtained by evaluating the kernel function between the tree with itself.

As a first step to understand the computation of tree kernels consider the indicator function:

$$[n_i \rhd f_k^\star] = \begin{cases} 1, & \text{if } \exists f_\ell : \text{root}(f_\ell) = n_i \text{ and } f_\ell \in f_k^\star \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

Clearly, the k-th dimension of the feature vector $\phi(t_i)$ can then be written as

$$\phi(t_i)^k = \sum_{n_j \in \mathcal{N}(t_i)} [n_j \rhd f_k^\star]. \tag{5.3}$$

This reasoning leads us directly to the definition of the tree kernel function.

Definition 5.5 (Tree Kernel (Collins and Duffy, 2002))**.** Given two trees t_1 and t_2, the *tree kernel* is defined as as:

$$\kappa_{\mathcal{T}}(t_1, t_2) = \sum_{n_i \in \mathcal{N}(t_1)} \sum_{n_j \in \mathcal{N}(t_2)} \Delta(n_i, n_j),$$

whereby

$$\Delta(n_i, n_j) = \sum_{f_k^\star \in \mathcal{F}^\star} [n_i \rhd f_k^\star][n_j \rhd f_k^\star].$$

Consequently, the value of Δ is equal to the number of matching fragments rooted at nodes n_i and n_j. Obviously, the naive enumeration over all tree fragments in the computation of $\Delta(n_i, n_j)$ is computationally problematic as the number of substructures that need to be considered grows exponentially in the number of nodes of the input trees. However, as the major source of a substantial speed-up, the tree kernel formulation by Collins and Duffy (2002) now allows to efficiently compute $\Delta(n_i, n_j)$ by means of Procedure 5.1. This recursive computation of $\Delta(n_i, n_j)$ yields the same result as the its formulation in Definition 5.5.

Procedure 5.1
Tree Kernels — Computation of Δ (Collins and Duffy, 2002)

```
Input: nodes $n_i$,$n_j$
1: if label$(n_i) \neq$ label$(n_j)$ or $\exists k$ : label(child$(n_i, k)) \neq$ label(child$(n_j, k)$) then
     /* nodes have different productions */
2:     $\Delta(n_i, n_j) \leftarrow 0$
3: else if $n_i, n_j$ pre-terminal symbols then
     /* nodes have identical productions and represent pre-terminal symbols */
4:     $\Delta(n_i, n_j) \leftarrow 1$
5: else
     /* nodes have identical productions but do not represent pre-terminal symbols */
6:     $\Delta(n_i, n_j) \leftarrow \prod_{k=1}^{|\,\text{child}(n_i)|} (1 + \Delta(\text{child}(n_i, k), \text{child}(n_j, k))).$
       /* recursive call to computation of Δ */
7: end if
8: return $\Delta(n_i, n_j)$
```

As the second step to understanding the computation of the tree kernels, the equivalence with this procedure can be verified as follows. The first two cases in Procedure 5.1 are obviously correct. The third recursive computation follows because for a given pair n_i and n_j the number of common fragments is the product of all variants that can occur at each child node of the production. For each such child node, there is the choice of stopping the expansion at the non-terminal symbol at this child or consider any of the equivalent sub-fragments at that child. Thus there must be $1 + \Delta(\text{child}(n_i, k), \text{child}(n_j, k))$ variants at the k-th child.

Decay Factors for Tree Height

Usually, there will be far more tree fragments of larger depth than of smaller depth. To account for the specificity of trees of different sizes, it makes sense to downweight the contribution of larger tree fragments to the kernel. It is thus common to introduce a decay factor λ, which can be added by modifying the lines (4) and (6) of Procedure 5.1 as follows:

line 4: $\Delta(n_1, n_2) \leftarrow \lambda,$

line 6: $\Delta(n_i, n_j) \leftarrow \lambda \prod_{k=1}^{|\operatorname{child}(n_i)|} (1 + \Delta(\operatorname{child}(n_i, k), \operatorname{child}(n_j, k))).$

The decay factor $\lambda < 1$ penalizes larger tree structures by giving them less weight in the overall summation. This corresponds to implicitly weighting the vector entries $\phi(t_i)^k$ by $\sqrt{\lambda^{\operatorname{size}(f_k^\star)}}$. Hereby, $\operatorname{size}(f_k^\star)$ corresponds to the number of grammar production rules in the fragments in $f_k^\star$. For the case of $\lambda = 1$, the kernel result corresponds exactly to the number of common fragments.

Furthermore, the value of $\kappa_{\mathcal{T}}(\cdot,\cdot)$ will still depend very much on the size of the trees. It is thus common to normalize tree kernel functions by using the cosine normalization modifier to equal out any scaling effects of different tree sizes.

5.3.2 Feature Space Design for Semantic Syntactic Tree Kernels

We will now introduce the family of Semantic Syntactic Tree Kernels. To motivate these kernel functions, we will first analyze how a feature space that considers both, syntax and semantics, should be set up. We will use this reasoning to guide the design of Semantic Syntactic Tree Kernels whereas the practical computation of these kernel functions will be discussed subsequently.

Motivation

The basic tree kernels introduced in the previous section rely on the intuition of counting all common substructures of two trees. However, like in the case of the VSM representation, the variability in natural language leads to undesired deficiencies. If two trees have substructures that represent the same syntactic phenomenon but employ different but related terminology at the leaves, these substructures will not yield any contribution. As in the case of single VSM dimensions, this situation is an evident drawback from the viewpoint of lexical semantics. Furthermore, it appears particularly problematic when considering that many tree fragments will contain several terminal symbols such that unintended mismatches are even more likely.

Example 5.6. Consider the fragments (i) `[NP [DT [`*"the"*`] NN [`*"tourist"*`] NNS [`*"attractions"*`]]]` and (ii) `[NP [DT [`*"the"*`] NN [`*"tourist"*`] NNS [`*"sights"*`]]]` on the one hand and the fragment (iii) `[NP [DT [`*"a"*`] NN [`*"police"*`] NNS [`*"car"*`]]]]` on the other hand. According to the basic tree kernel, none of these fragments would be matched (although selected subfragments would). However, from the viewpoint of the intended meaning, the fragments (i) and (ii) should be more similar than fragments (i) and (iii) or (ii) and (iii), respectively.

Considering Semantic Similarity of Tree Fragments

We now look at techniques to simulate an effect as envisioned in Example 5.6. The proposed approach takes up the ideas of semantic smoothing which we have encountered in Section 5.2 in the context of the VSM representation and recasts these concepts in the context of tree kernels.

In analogy with the SSK formulation, we are now interested in also accounting for *partial matches* between tree fragments during kernel evaluation. Let us for the moment assume the existence of a positive semi-definite similarity function $\kappa_{\mathcal{F}^\star}(\cdot,\cdot)$ defined on tree fragment classes. As the function needs to be positive semi-definite, it can equivalently be seen as a kernel function on tree fragment classes. Interpreting the resulting similarities as smoothing factors in a similar manner as in the SSK model, we now define the overall SSTK as the sum over the evaluations of $\kappa_{\mathcal{F}^\star}$ over all pairs of tree fragment classes in the argument trees. Technically, this means changing the summation in the second formula of Definition 5.5 as suggested by the following definition.

Definition 5.6 (Semantic Syntactic Tree Kernel (SSTK))**.** Given two trees t_1 and t_2, we define the *Semantic Syntactic Tree Kernel* as:

$$\kappa_{\mathcal{T}}(t_1, t_2) = \sum_{n_i \in \mathcal{N}(t_1)} \sum_{n_j \in \mathcal{N}(t_2)} \Delta(n_i, n_j),$$

whereby

$$\Delta(n_i, n_j) = \sum_{f_k^\star \in \mathcal{F}^\star} \sum_{f_\ell^\star \in \mathcal{F}^\star} [n_i \rhd f_k^\star][n_j \rhd f_\ell^\star] \; \kappa_{\mathcal{F}^\star}(f_k^\star, f_\ell^\star).$$

Proposition 5.2 (Validity of Semantic Syntactic Tree Kernels)**.** *The kernel function on trees of Definition 5.6 is a positive semi-definite function and thus constitutes a valid kernel.*

The validity is readily verified via the formulation of the kernel as crossproduct kernel. Recall from Proposition 4.2 that such functions are valid kernels if the embedded function is a valid kernel as assumed here for $\kappa_{\mathcal{F}^\star}(\cdot,\cdot)$.

Conceptually, the computation of Δ is changed in that it does not only count all identical tree fragments rooted at the two argument nodes. Instead, if two tree fragments rooted at the arguments are different but somehow similar according to the similarity notion encoded in $\kappa_{\mathcal{F}^\star}(\cdot,\cdot)$, their similarity will also contribute to the evaluation of Δ and thus also to the evaluation of the overall kernel $\kappa_{\mathcal{T}}(\cdot,\cdot)$.

Designing Fragment Class Similarity

So far we have just assumed that $\kappa_{\mathcal{F}^\star}(\cdot,\cdot)$ is positive semi-definite and that it encodes some useful notion of similarity of tree fragment classes without discussing the actual setup of this function. We now formalise this similarity of tree fragments. First, we require that two tree fragment classes, to be similar, need to be compatible with one another.

Definition 5.7 (Fragment class compatibility)**.** Two fragment classes $f_i^\star$ and $f_j^\star$ are considered to be *compatible* or to *match partially* if they are isomorphic to each other with respect to node labels and their connections whereby the labels of terminal symbols in both fragment classes are replaced by a wildcard symbol for terminal nodes and we denote this situation by $f_i^\star \hat{\sim} f_j^\star$.

A partial match thus occurs when two fragment classes differ only by their terminal symbols, e.g. `[NNS [`*"attractions"*`]]`$\hat{\sim}$`[NNS [`*"sights"*`]]`.

Next, in line with our motivation in Example 5.6, we require that the similarity of two compatible *fragments* should be determined by the mutual lexical similarities of the *terminal symbols, i.e. the terms,* at corresponding positions within the fragments of the respective tree fragment classes.

The approach we take is based on modelling the similarity of two tree fragments as products of the similarities of the contained pairs of terms. Note that, as the tree fragments need to be compatible, they have the same number of terminal symbols at compatible positions. Conceptually, this means that for similarity measures normalized to the $[0,1]$ interval, the similarity of two tree fragments is above zero only if all matching pairs are similar to some extent. The fragment similarity is evaluated as the product of all semantic similarities of corresponding terminal nodes (i.e. sitting at identical positions). It is maximal if all pairs have a similarity score of 1 while it decays fast as soon as individual pairs of matching nodes have smaller similarity values.

Example 5.7. Consider the fragments (i) `[NP [DT [`*"the"*`] NN [`*"tourist"*`] NNS [`*"attractions"*`]]]` and (ii) `[NP [DT [`*"the"*`] NN [`*"tourist"*`] NNS [`*"sights"*`]]]` introduced in Example 5.6. The two fragments ar compatible as the structure of the non-terminal symbols is identical. Further, the terms *"the"* and *"tourist"* at terminal positions 1 and 2 are identical, i.e. they have (normalized) similarity of 1. The terms *"attractions"* and *"sights"* at position 3 are different but similar, e.g. at a similarity value of 0.75. The overall similarity of the two fragments is then $1 \cdot 1 \cdot 0.75 = 0.75$.

Remark 5.1. We here generally require that the employed similarities on the tree fragments are normalized to the $[0,1]$ interval. While it would be possible to also include non-normalized similarity functions with varying degrees of self similarity this would, in analogy to the analysis for SSKs lead to a reweighting of the argument trees.

To ensure the validity of the overall fragment class kernel, the employed term similarity needs to be a valid kernel on the terms itself. Immediately, we can note the correspondence of this requirement with the requirements for SSKs to employ positive semi-definite smoothing matrices. Taking up the notation introduced for Semantic Smoothing Kernel, we generally denote the term similarity kernel as $\kappa_{\mathcal{L}}(\cdot,\cdot)$. Based on this reasoning, we now formally define the discussed notion of the similarity of two tree fragment classes.

Definition 5.8. For two fragment classes $f_i^\star, f_j^\star \in \mathcal{F}^\star$, we define the *Tree Fragment Similarity Kernel* as:

$$\kappa_{\mathcal{F}^\star}(f_i^\star, f_j^\star) = [f_i^\star \tilde{\sim} f_j^\star] \prod_{\ell \in \text{termindices}(f_i^\star)} \kappa_{\mathcal{L}}(\text{label}(f_i^\star, \ell), \text{label}(f_j^\star, \ell))$$

where termindices$(f_i^\star)$ denotes the indices of term nodes in fragments of class $f_i^\star$ in some fixed order and label$(f_i^\star, k)$ denotes the label of the k-th node in fragments of class $f_i^\star$ according to this order (e.g. numbered from left to right).

Proposition 5.3 (Validity of the Tree Fragment Similarity Kernel)**.** *Given, that the embedded term similarity function $\kappa_{\mathcal{L}}(\cdot,\cdot)$ is positive semi-definite, the kernel function on tree fragments of Definition 5.8 is a positive semi-definite function and thus constitutes a valid kernel on tree fragments.*

Proof. Consider a vector space with dimensions corresponding to all possible tree fragments without any terminal symbols. Let a tree fragment be represented by a 0/1 vector with a single entry of 1 at the dimension that corresponds to its structure without the tree fragments. It is straightforward to see that the indicator function $[f_i^\star \tilde{\sim} f_j^\star]$ corresponds to an inner product in this space. The overall kernel is thus a product of kernel evaluations and, by Proposition 2.11 constitutes a valid kernel. □

As with SSKs, we obviously need to choose an appropriate notion of term similarity $\kappa_{\mathcal{L}}(\cdot,\cdot)$, i.e. a valid and adequate smoothing parameter. We discuss the choice of such term similarity kernels for both, SSKs and SSTKs in Section 5.4.

Remark 5.2 (Constrained Word Sense Ambiguity). Beside the novelty of taking into account tree fragments that are not identical it should be noted that in contrast to the application of lexical semantic similarities to simple term vectors in semantic smoothing kernels, the lexical semantic similarity is now constrained in the syntactic structures, which limit errors/noise due to incorrect (or not provided) word sense disambiguation.

Procedure 5.2
Semantic Syntactic Tree Kernels — Computation of Δ (Bloehdorn and Moschitti, 2007a)

```
Input: nodes $n_i$,$n_j$
1: if label($n_i$) $=$ label($n_j$) and $n_i, n_j$ pre-terminal symbols then
     /* nodes represent compatible pre-terminal symbols */
2:   $\Delta(n_i, n_j) \leftarrow \lambda\, \kappa_{\mathcal{S}}(\mathrm{label}(\mathrm{child}(n_i, 1)), \mathrm{label}(\mathrm{child}(n_j, 1))$
3: else if label($n_i$) $\neq$ label($n_j$) or $\exists k : \mathrm{label}(\mathrm{child}(n_i, k)) \neq \mathrm{label}(\mathrm{child}(n_j, k))$ then
     /* nodes have different productions and do not represent pre-terminal symbols */
4:   $\Delta(n_i, n_j) \leftarrow 0$
5: else
     /* nodes have identical productions and do not represent pre-terminal symbols */
6:   $\Delta(n_i, n_j) \leftarrow \lambda \prod_{k=1}^{|\,\mathrm{child}(n_i)|} (1 + \Delta(\mathrm{child}(n_i, k), \mathrm{child}(n_j, k)))$.
      /* recursive call to computation of $\Delta$ */
7: end if
8: return $\Delta(n_i, n_j)$
```

5.3.3 Efficient Computation of Semantic Syntactic Tree Kernels

In the formulation of the Semantic Syntactic Tree Kernel in Definition 5.6 we note a basic structure similar to the one in Semantic Smoothing Kernels. However, in contrast to SSKs, it is unrealistic to compute the kernel by summing over all pairs of explicit term fragment features. While we could use optimizations of the type we have encountered for SSKs, we have already noted for basic tree kernels that the explicit representation of trees as vectors of tree fragment classes is out of scope in any realistic scenario.

However, it turns out that it is possible to use a similar recursion structure for the computation of $\Delta(\cdot, \cdot)$ like the one for basic tree kernels to achieve the kernel evaluation without any explicit representation in terms of feature vectors. This alternative computation of Δ is summarized in Procedure 5.2.

Procedure 5.2 for the computation of $\Delta(\cdot, \cdot)$ exhibits a similar structure as Procedure 5.1. However, in contrast to the original formulation in Procedure 5.1, it does not return with a 0 value in all cases where the nodes of two labels differ. Rather, the result of this comparison depending on whether terminal nodes or interior nodes are compared. In the former case, the evaluation returns the result of the similarities of the respective terms. Only in the latter case, the evaluation yields a zero result.

The equivalence of this computation with the model of SSTKs introduced above can be verified as follows. The first and second case in Procedure 5.2 are obviously correct. The third recursive computation follows because for a given pair n_i and n_j the number of common fragments is the product of all variants that can occur at each child node of the production. Again, for each such child node, there is the choice of stopping the expansion at the non-terminal symbol at this child or consider any of the equivalent sub-fragments at that child. As soon as relevant child nodes are included which correspond to terminal nodes, the result may differ from the original formulation. For one pair of terms, their similarity is the sole similarity that affects the similarity of the corresponding tree fragment classes. As any additional terminal node enters the computation multiplicatively, this corresponds exactly to the definition of the tree fragment similarity kernel above.

5.3.4 Complexity

Given two trees with n and m nodes, the complexity of the basic tree kernels amounts to $O(nm)$. First, note that each pair of nodes needs to be considered for computation in the $\Delta(\cdot,\cdot)$ function. Using a naive evaluation, each computation of $\Delta(\cdot,\cdot)$ might require up to another $\min(n,m)$ operations for each pair of nodes. However, the problem of computing $\Delta(\cdot,\cdot)$ for all nodes has overlapping subproblems due to the recursive definition which reuses the computed values on lower-level nodes. Using ideas from the dynamic programming area, we can thus traverse the two argument trees t_1 and t_2 in post-order, thereby filling up a $n \times m$ matrix of precomputed values of $\Delta(\cdot,\cdot)$ yielding only another constant factor for look-ups and products for any higher-level tree node.

Collins and Duffy (2002) have pointed out that for the basic tree kernels, this worst case estimate may vary substantial from the average running times which can be described as linear in the number of node pairs n_i and n_j such that the productions at n_i and n_j are the same. In most settings, the number of nodes with identical productions is linear, so that the large majority of values of $\Delta(\cdot,\cdot)$ is 0. Thus the running time is close to linear in the size of the trees. Obviously, the possibility of partial matches at the level of terms will destroy some of the sparsity of this matrix and thus the near-linearity of the basic tree kernel computation. However, the worst case complexity remains in $O(n\,m)$.

5.3.5 Extensions

In this section we discuss a number of extensions of the basic model of Semantic Syntactic Tree Kernels.

Considering Other Tree Fragment Paradigms

In the beginning of this section we have discussed other classes of tree fragments which could be considered. In particular, Vishwanathan and Smola (2003, 2004) suggest to consider only proper subtrees. The feature representation for a parse tree should thus only consider substructures that are characterized by expanding an arbitrary internal node of the original tree until the leaf nodes and may not stop earlier as in the model of Collins and Duffy (2002) which we have considered so far.

For the implementation, this behaviour can be mimicked by changing Procedures 5.2 as follows:

$$\text{line 6: } \Delta(n_i, n_j) \leftarrow \lambda \prod_{k=1}^{|\operatorname{child}(n_i)|} (\Delta(\operatorname{child}(n_i,k), \operatorname{child}(n_j,k))).$$

The only difference is thus that the term "1 +" on the right hand side is omitted thus removing any contributions of fragments that end at an internal node.

Tuning Leaf Contribution

We have remarked that the parameter λ is well suited to reduce the contribution of larger tree structures. On the other hand, we want to allow fragments with many matching/similar leaves to contribute more to the overall kernel than fragments without leaves. For this purpose, we introduce an additional parameter, α in the computation of the Δ function by modifying step (1) of the above computation (i.e. leaf matching) as follows:

$$\text{line 2: } \Delta(n_i, n_j) \leftarrow \alpha\, \lambda\, \kappa_{\mathcal{S}}(\operatorname{label}(\operatorname{child}(n_i,1)), \operatorname{label}(\operatorname{child}(n_j,1))$$

It is readily verified that this corresponds to implicitly weighting the vector entries $\phi(t_i)^k$ by $\sqrt{\alpha^{|\{n_j \in t_i | \operatorname{label}(n_j) \text{ is terminal symbol}\}|}}$. While α could of course be used to further reduce the contribution of the leaves, we will typically select a value $\alpha > 1$ to allow for a stronger contribution.

Normalizing POS Tag Labels

In the context of tree kernels, it could also sometimes be useful to reduce the employed tag sets to simpler variants than commonly used, for example to distinguish only between basic word categories like nouns, verbs, adjectives, adverbs etc. but not between their precise subtypes. For example the commonly used Penn Treebank Tagset in Appendix B distinguishes between the tag `NN` (noun, singular or mass) and `NNS` (noun, plural). Correspondingly, we may not want to distinguish between terminal symbols that have a common base form, i.e. are only inflected variants of the same term. In these cases, one can regard a given label as equivalent to another label and denote this situation as $\text{label}(n_i) \sim \text{label}(n_j)$. Obviously, we can then redefine the notion of equivalence of two fragments and implement this variant by altering the first condition in Procedure 5.2 accordingly.

5.4 Choice of Smoothing Parameters

The central design choice for both types of kernels, SSKs and SSTKs is the employed smoothing parameter, represented as term similarity kernel $\kappa_{\mathcal{L}}(\cdot, \cdot)$ or, equivalently, as smoothing matrix **S** in the case of SSKs.

A straightforward choice would be to use any given similarity function defined on the taxonomic back-end of a knowledge structure like WORDNET as smoothing parameter. However, we have seen in Chapter 4 that many classical semantic similarity functions of this type are, at least in the general case, not positive semi-definite and can thus not be used directly as parameters in SSKs or SSTKs. This is particularly true for WORDNET, which contains numerous cases of multiple inheritance at several levels of the hypernym hierarchy. In the following subsections, we discuss the main available possibilities. In the next chapter, we will investigate these variants experimentally.

5.4.1 Investigating Positive Semi-Definiteness Case-by-Case

We have seen that various classical taxonomic semantic similarity functions are not positive semi-definite and it will thus often not be possible to use them as kernels or kernel parameters. Obviously, this does not exclude that under specific conditions a given similarity function might yield a valid kernel. One possibility would thus be to build up candidate smoothing matrices for a specific learning task based on such similarity functions and investigate their positive semi-definiteness in the specific case. Such an approach has been used by Basili et al. (2005b) in conjunction with the Conceptual Density measure of Agirre and Rigau (1996). However, this approach is laborious and may nevertheless eventually result in indefinite kernel functions while omitting any practical solution for this situation.

5.4.2 Fixing Indefinite Kernels

Another approach would be to "fix" a given matrix of smoothing parameters or the resulting kernel matrix in case that the required positive semi-definiteness can not be shown. Haasdonk and Bahlmann (2004) mention some techniques for achieving this effect. One solution is to modify the entries on the matrix diagonal. If these are sufficiently big, a symmetric matrix necessarily becomes positive semi-definite. Similarly, it would be possible to perform an eigendecomposition of the kernel matrix, set all negative eigenvalues equal to zero and reassemble the kernel matrix again. The problem with these and other approaches in this direction is that they are on the one hand laborious and on the other hand effectively yield new measures whose interpretation becomes unclear. Furthermore, the results will vary very much depending on the amount of "fixing" involved and will not necessarily relate any more to the original similarity entries.

5.4.3 Matrix Squaring for Arbitrary Measures

As a way to avoid the possibility of indefinite similarity matrices, authors like Siolas and d'Alche Buc (2000) have enforced the positive-definiteness of $\mathbf{S}$ by explicitly computing it from $\mathbf{S} = \mathbf{P}\mathbf{P}'$ whereby the information about the mutual similarities between all terms is now encoded in the matrix $\mathbf{P}$. Conceptually this approach maps each term to all other terms with a given similarity value. The actual similarity value that eventually feeds into the computation of the Semantic Smoothing Kernel is then the shared weight of all these terms. While this approach obviously ensures the validity of the kernel, the interpretation of the resulting Semantic Smoothing Kernel is less clear. In particular, the resulting smoothing matrix depends not only on the employed similarity function but also on the overall vocabulary as different vocabulary configurations may yield different entries within the squared matrix for the same pair of terms.

5.4.4 Using Similarity Functions based on Taxonomic Overlap

The set of taxonomic similarity functions which are inherently positive semi-definite, regardless of the setup of the knowledge structure or any parameters are the similarity functions of the family of taxonomic overlap measures.

Because the notion of determining the similarity of two terms by means of the amount of shared superconcepts is intuitive, measures based on the taxonomic overlap of superconcepts will be the focus of our subsequent experiments. Furthermore, the possibility to use weighted variants of the the original taxonomic overlap measure allows to combine the notions of the other similarity measures with those of taxonomic overlap.

5.5 Related Work

In this section, we review previous research work that relates to the topics presented in this chapter. We discuss both, traditional methods as well as methods based on the kernel paradigm.

5.5.1 Semantic Smoothing and Query Expansion

The problem of the variability of natural language lies at the core of text mining, natural language processing and information retrieval. The class of Semantic Smoothing Kernels thus bears connections to a variety of existing work in these fields.

Kernel Methods and Semantic Smoothing

As a first approach to incorporate notions of semantic smoothing directly into kernel functions, Siolas and d'Alche Buc (2000) used squared matrices of mutual inverted path lengths between terms to alter the basic Euclidean distance between two vectors. This metric was then used for k-Nearest Neigbhour (kNN) classification and with a Gaussian kernel. The intention to design an appropriate distance function boils down to Semantic Smoothing Kernels as introduced in this chapter when viewing the resulting distance as a kernel-induced distance function as discussed in Definition 2.25. Cristianini et al. (2002), take up the approach of Siolas and d'Alche Buc (2000) for motivating their own work on Latent Semantic Kernels (LSKs). These kernels are based on Latent Semantic Indexing (LSI), a dimensionality reduction technique we review below. Shawe-Taylor and Cristianini (2004) later discuss this setting in more detail.

Basili et al. (2005b, reprinted in (Basili et al., 2006)) take up the concept of SSK in conjunction with the Conceptual Density (CD) measure of Agirre and Rigau (1996). The basic setting is similar to the one considered in this thesis and has in fact, motivated the further joint work on SSKs in Bloehdorn et al. (2006b). In contrast to the work in this thesis, there is no analysis whether the employed CD measure is positive semi-definite in general. Basili et al. suggest to approach this issue by investigating the validity of the smoothing matrix on a case-by-case basis. Finally, Mavroeidis et al. (2005b) report on experiments with SSKs. Here, the setup of the smoothing matrix was determined by the shared amounts of hypernyms, whereby the number of hypernyms was restricted to fixed distances from the reference terms. This crisp setting is a special case of the similarity measures based on the smoothing parameters based on the (weighted) semantic cotopy of the reference terms as employed in this thesis.

Query Expansion and Conceptual Document Representations

In Section 5.2.2, we have seen that SSKs are strongly related to the explicit transformation of the feature vectors, at least as long as the transformation corresponds to a linear transformation of the type $\phi(x) = Px$. Traditionally, such approaches aim at the expansion of the VSM vectors or documents or search queries with additional terms derived from a thesaurus, taxonomy or ontology. Especially in the context of IR applications, this technique is therefore also referred to as *query expansion*. In contrast to the kernel-based setting as investigated in this thesis, these methods perform explicit transformations of the feature vectors and are thus not capable of incorporating optimization strategies like the ones considered in our formulation of SSKs.

Information Retrieval Several researchers have reported positive results concerning query expansion in the context of IR applications. In early work on the topic, Salton and Lesk (1971), found that expansion with synonyms improved performance, while using broader or narrower terms produced too inconsistent results for being actually useful. Wang et al. (1985) report that a variety of lexical-semantic relations improved retrieval performance. A comprehensive study by Voorhees (1994) indicated that query expansion is especially useful when queries are relatively short.

Text Clustering and Text Classification Similar techniques have also been applied in text classification and text clustering settings. Green (1999) uses WordNet to construct chains of related synsets from the occurrence of terms for document representation and subsequent clustering but does not evaluate performance of the approach in comparison with standard VSM representations. Other results from similar settings are reported by Scott and Matwin (1999) as well as Wang et al. (2003).

Work on text clustering by Hotho et al. (2003); Hotho (2004) as well as own prior work (Bloehdorn and Hotho, 2004) has shown promising results when using additional conceptual features extracted from manually engineered ontologies. de Buenaga Rodriguez et al. (1997) as well as Ureña et al. (2001) show a successful integration of the WORDNET resource for a document categorization task. However, their results are based on manually constructed conceptual representations, thereby alleviating the problem of word sense disambiguation (WSD) altogether.

Term Clustering and Automatically Derived Knowledge Structures

Along another line, various research endeavours have focused on the use of knowledge structures that were constructed automatically in an unsupervised process, e.g. by means of ontology learning techniques (Maedche and Staab, 2001; Cimiano, 2006).

Jing and Croft (1994) could show an improvement in information retrieval tasks by means of query expansion whereby the background knowledge resource was an automatically constructed association thesaurus. Similar successful approaches using automatically generated thesauri for query expansion are reported in literature (Park and Choi, 1996; Schütze and Pedersen, 1997).

Own prior work has has investigated the question of using automatically constructed knowledge structures for conceptual document representations in text clustering and classification (Bloehdorn et al., 2006c). The results are based on the taxonomy induction technique proposed by Cimiano et al. (2005) and show that such knowledge structures can achieve results that mimic much of the positive effects of manually constructed knowledge structures suggesting a road for a fully integrated life cycle between ontology learning from text and exploitation of the resulting knowledge structures within text mining (Bloehdorn et al., 2005).

Another group of methods try to archive a sophisticated input representation via word clusters thus abandoning the notion of a background ontology altogether. Here, terms are clustered based on their joint occurrence in a given set of documents and use cluster weights to represent the documents. For example Baker and McCallum (1998) or Bekkerman et al. (2001) cluster words based on the class label distribution for each word.

Latent Semantics

Finally, *dimensionality reduction techniques* are often used to mimic the behaviour of conceptual representations by computing kind of semantic concepts statistically from term co-occurrence information. In contrast to conceptual document representations driven by lexical knowledge structures, the concept-like structures are, however, not easily interpretable by humans. The most prominent approach is certainly the Latent Semantic Indexing (LSI) technique proposed by Deerwester et al. (1990)). We here review the main ideas.

Consider a corpus of documents in VSM representation, represented as a $|\mathcal{X}| \times |\mathcal{L}|$ matrix $\mathbf{X}$. Now consider the singular value decomposition (SVD) of $\mathbf{X}$ (c.f. Definition 2.16) as into $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}'$. Assuming that the singular values are ordered by by size, the first k singular values represent the best (in a least squares sense) rank-k approximation of the original data. The core idea of LSI is to work on such a lower rank approximation to reduce the variability of the terminology used. Conceptually, this is done by setting all but the largest k singular values to zero. Equivalently, the corresponding columns can be deleted from $\mathbf{V}$,$\mathbf{U}$ and $\boldsymbol{\Sigma}$ altogether, yielding the reduced matrices $\mathbf{V}_k$,$\mathbf{U}_k$ and $\boldsymbol{\Sigma}_k$ such that $\hat{\mathbf{X}} = \mathbf{U}_k\boldsymbol{\Sigma}_k\mathbf{V}_k'$. Note that as $\mathbf{V}$ is unitary, the column-reduced matrix has the properties $\mathbf{V}_k'\mathbf{V}_k = \mathbf{I}$ while in general $\mathbf{V}_k\mathbf{V}_k' \neq \mathbf{I}$. The evaluation of the dot product on all pairs of input vectors in the reduced representation now becomes $\hat{\mathbf{K}} = \hat{\mathbf{X}}\hat{\mathbf{X}}' = \mathbf{U}_k\boldsymbol{\Sigma}_k\mathbf{V}_k'\mathbf{V}_k\boldsymbol{\Sigma}\mathbf{U}_k' = \mathbf{U}_k\boldsymbol{\Sigma}_k\boldsymbol{\Sigma}_k\mathbf{U}_k'$. Cristianini et al. (2002) have used the paradigm of SSKs to motivate their work on the embedding of LSI into kernel functions.

A probabilistic variant of LSI is the Probabilistic Latent Semantic Indexing (PLSI) technique by Hofmann (1999) which has also successfully been applied for TC tasks (Cai and Hofmann, 2003).

5.5.2 Syntactic Representations

Along a line between syntax and semantics, Moschitti and Basili (2004) has investigated the possibility of incorporating syntactic information like n–grams or POS information for terms directly into the VSM representations for text mining tasks. However, the reported results indicate only an insignificant gain in classification accuracy. Most probably, this can be explained by the fact that the incorporated information was too shallow to yield an increase in performance as opposed to the far richer representation by means of tree kernels. On the other hand, all reported experiments related to problems of document classification by general topics, a situation where syntactic information is not directly expected to help.

For the case of alternative tree kernel functions, Zelenko et al. (2003); Culotta and Sorensen (2004) develop a parse tree kernel in the context of relation extraction. The tree kernel is recursively defined on dependency parse trees in a top-down manner, matching nodes from roots to leaf nodes. For each pair of matching nodes, a subsequence kernel on their child nodes is invoked, which matches either contiguous or sparse subsequences of nodes. In a recent study, Zhang et al. (2008) has provided evidence that proper convolution kernels (Haussler, 1999) like the ones covered in our exposition are best suited for relation extraction settings.

5.5.3 Other Kernel Paradigms for Text Data

Along a radically different line, Lodhi et al. (2002) and Vishwanathan and Smola (2003) have proposed the concept of *string kernels*. String kernels consider textual input solely as symbol sequences. The feature space of string kernels corresponds to the set of all (non-contiguous) substrings of symbols up to a specified length. The more substrings two textual inputs have in common, the more similar they are considered. Similar to tree kernels, these kernels require no explicit representation of the substrings. Instead, the kernel functions are computed by means of sequence alignment techniques. The combination with dynamic programming techniques makes the computation of string kernels efficient. On the other hand, string kernels are well suited to deal with languages for which no or little language resources exist and which do not lend themselves to easy preprocessing (e.g. lemmatization). At the same time, they are more robust to noise in the lexical surface forms such as e.g. spelling errors. While String kernels have shown surprisingly good results in text classification settings on selected datasets. However, these kernels require substantial amounts of training data and usually lack the full power of text representation based on terms.

5.6 Summary and Discussion

The successful exploitation of the mutual dependencies between textual data and knowledge structures is a long standing vision in text mining (Bloehdorn et al., 2005). In this chapter, we have introduced two types of kernel functions which provide such capabilities.

On the one hand, Semantic Smoothing Kernels (SSKs) allow to exploit the semantic similarity of terms in the context of the classical Vector Space Model (VSM). On the other hand, we have shown how to combine the merits of this type of semantic smoothing with the rich syntactic representations of tree kernels. The resulting family of Semantic Syntactic Tree Kernels (SSTKs) is the first principled approach in the direction of jointly addressing issues of lexical semantics and syntactic structure in kernel-based learning. For both types of kernels, we have provided an in-depth analysis of their properties in terms of interpretation and complexity as well as a discussion of extensions of the basic paradigms.

Both types of kernels require smoothing parameters which represent a useful notion of lexical semantic similarity and at the same time ensure the validity of the resulting kernels. The investigation of the positive semi-definiteness of various similarity measures on formal knowledge structures in Chapter 4 provides guidance on which of these can be used as smoothing parameters for the kernel functions proposed in this chapter. In the next chapter, we provide experimental evidence that the presented techniques improve upon the still prevailing VSM representation in conjunction with the conventional inner product as a similarity measure.

Finally it is worth to note that although this chapter as well as the experimental investigation in the next chapter are entirely focused on the case of textual data, the application of the paradigm of semantic smoothing by means of Semantic Smoothing Kernels could readily be extended to other types of data. The only requirement would be that the conventional vectorial features can be associated with formal structures that encode their mutual dependencies.

Chapter 6

Experiments with Kernel Functions for Semantic Smoothing

In the previous chapter, we have presented two types of kernel functions, namely Semantic Smoothing Kernels and Semantic Syntactic Tree Kernels. In different ways, both groups of kernel functions account for the variability in human language by incorporating background knowledge from lexical knowledge structures. In this chapter, we investigate the application of both types of kernels in a series of practical text mining experiments. As the experiments show, both approaches are effective extensions of the current state-of-the-art approaches. For both types of kernels we particularly investigate how the specific design choices for smoothing parameters affect performance.

This chapter is structured as follows. We shortly discuss the implementation of SSKs and SSTKs in Section 6.1. In Section 6.2, we report on experiments with SSKs on the well-known REUTERS-21578 text classification dataset. In Section 6.3, we report on experiments with SSKs on a text classification task for AMAZON book abstracts. Finally, Section 6.4, reports on experiments with SSTKs in the context of a question classification task. We conclude with a short summary and discussion in Section 6.5.

6.1 Implementation

As part of this thesis, both types of kernels, SSKs and SSTKs, have been implemented as software modules. The implementation of Semantic Smoothing Kernels is available as a plug-in for SVMLIGHT software by Joachims (1999), which constitutes one of the most well known implementations of the SVM classification algorithms.[1] The module also implements the optimizations discussed in the context of Section 5.2.3. Just as the core modules of SVMLIGHT, it is implemented in C++.

While this implementation provides a deep integration of Semantic Smoothing Kernels and SVMLIGHT, a second and more generic extension of SVMLIGHT provides an interface to kernel functions that are implemented in JAVA by means of the

[1]The SSK kernel plug-in for SVMLIGHT (V6.01) is freely available at `http://www.aifb.uni-karlsruhe.de/WBS/sbl/software/semkernel/` together with accompanying documentation.

Java Native Interface (JNI) Invocation API. Together with this extension, we provide JAVA-based implementations of SSKs and SSTKs as well as variety of other well-known kernel functions.[2] Both types of kernels, SSKs and SSTKs, as well as basic tree kernels and various other well-known kernel functions are part of a JAVA library for use within SVMLIGHT via the JNI kernel plug-in or for use with any other kernel-based software infrastructure.

6.2 Experiments with Semantic Smoothing Kernels on REUTERS-21578

In the first set of experiments, we analyze the behaviour of SSKs in a classical text classification setting, namely the classification of news texts according to their main topics.

6.2.1 Task and Dataset

Description of the Dataset

As a dataset, we use the well-known REUTERS-21578 collection (Lewis, 1991). The documents in the REUTERS-21578 collection are news texts collected from the REUTERS news wire in 1987 (Lewis, 1991).[3] These texts, mostly financial news, are annotated with one or several out of 135 thematic categories. As an example, Figure 6.1 shows three exemplary documents for the categories *"earn"*, *"crude"*, and *"ship"*. The corpus was used for the first time for the evaluation of the CONSTRUE system, a manually designed text classification system built for REUTERS by Carnegie Group (Hayes and Weinstein, 1991). Although the corpus has repeatedly been criticized because of inconsistent category assignments and a number of peculiar characteristics (some of which will be discussed below) it has since then become a *de-facto* standard for text classification experiments (Lewis, 1991; Yang, 1999; Yang and Liu, 1999; Sebastiani, 2002).

Training vs. Test Data

The "ModApte" split, based on the split by Apté et al. (1994), divides the REUTERS-21578 collection into 9,603 training documents, 3,299 test documents and 8,676 unused documents and forms the basis for the experiments in this section. In the exper-

[2]The JNI kernel plug-in for SVMLIGHT (V6.01) and the corresponding kernel library are available at `http://www.aifb.uni-karlsruhe.de/WBS/sbl/software/jnikernel/` together with accompanying documentation.

[3]The REUTERS-21578 collection is freely available at `http://www.daviddlewis.com/resources/testcollections/reuters21578/`.

COBANCO INC YEAR NET
SANTA CRUZ, Calif., Feb 26 -
Shr 34 cts vs 1.19 dlrs
Net 807,000 vs 2,858,000
Assets 510.2 mln vs 479.7 mln
Deposits 472.3 mln vs 440.3 mln
Loans 299.2 mln vs 327.2 mln
Note: 4th qtr not available.
Year includes 1985 extraordinary gain from tax carry forward of 132,000 dlrs, or five cts per shr.
Reuter

TEXACO CANADA LOWERS CRUDE POSTINGS
NEW YORK, Feb 26 -
Texaco Canada said it lowered the contract price it will pay for crude oil 64 Canadian cts a barrel, effective today.
The decrease brings the company's posted price for the benchmark grade, Edmonton/Swann Hills Light Sweet, to 22.26 Canadian dlrs a bbl.
Texaco Canada last changed its crude oil postings on Feb 19.
Reuter

AGENCY REPORTS 39 SHIPS WAITING AT PANAMA CANAL
WASHINGTON, Feb 26 -
The Panama Canal Commission, a U.S.government agency, said in its daily operations report that there was a backlog of 39 ships waiting to enter the canal early today. Over the next two days it expects —

	2/26	*2/27*
Due:	*27*	*35*
Scheduled to Transit:	*35*	*41*
End-Day Backlog:	*31*	*25*

Average waiting time tomorrow —

North End:	*13hrs*	*15hrs*
South End:	*4hrs*	*26hrs*

Reuter

Figure 6.1 — Example documents of the REUTERS-21578 collection. Each of the three documents belongs to a single dedicated category, namely the categories *"earn"* (news on company earnings), *"crude"* (news related to crude oil), and *"ship"* (news on shipping and ocean freight).

iments, only the 10 categories having the largest numbers of positive training documents were considered — a common setting for this collection (Sebastiani, 2002).

Characteristics of the Dataset

The REUTERS-21578 collection exhibits a number of peculiar characteristics which affect the focus of our experiments as well as the interpretation of the results. While the corpus has been repeatedly criticized because of these findings, authors like Debole and Sebastiani (2004) have argued that exactly this "dirty" nature makes the REUTERS-21578 collection interesting for text classification.

Multi-Label Setting with Dependent Classes The collection conforms to a *multi-label setting* where each document may belong to more than one category. However, the target categories are often not orthogonal to each other but rather show semantic relationships. For example, the classes *"money-fx"* (news related to foreign exchange) and *"interest"* (news related to interest rates) are strongly related. Naturally, such situations lead to a significant overlap in category assignment. For example, 141 of the 347 positive training documents for *"interest"* also belong to *"money-fx"*.

Inhomogeneous Distribution of Classes Even for the 10 largest classes, the number of available positive training documents differs substantially. For example, the largest category, *"earnings"* has $2,877$ positive training documents associated with it while the smallest of the 10 categories we consider, *"corn"*, has only 181 associated positive training documents. In all cases, when considering the common reduction to binary classification problems that was discussed in Section 2.2.2, the collection exhibits a dominance of negative documents. While the largest category, *"earnings"*, has a ratio of roughly 2 negative training documents per positive training document, the smallest of the 10 categories considered, *"corn"*, yields a ratio of 52 negative vs. one positive training documents. Table 6.2 summarizes these statistics.

Inhomogeneous Characterization of Classes Most importantly, the corpus is rather inhomogeneous with respect to the types of patterns that can guide the classification. While some categories can be characterized by a small set of frequent terms with high discriminative power, others require complex combinations of medium and low frequency terms, each of which alone only has limited discriminative power. As an example, consider the categories *"earn"*, *"crude"* and *"ship"*. We have seen examples of a document for each of these in Figure 6.1. For each of these categories, Table 6.1 lists the 5 most frequent terms and indicates the empirical probabilities that a document in which the term appears

Table 6.1 — Term occurrence statistics for three selected REUTERS-21578 categories. For each of the three categories *"earn"*, *"crude"*, and *"ship"*, the table lists the 5 most frequent terms that occur in documents of the respective class. For each class, the first column names the term, the second column indicates the empirical probability $P(l_i|c)$ that the term appears in a document of the respective class while the third column indicates the empirical probability $P(c|l_i)$ that a document in which the term appears belongs to the respective class. All numbers are percentages and computed on the training and test documents of the REUTERS-21578 "ModApte" split. The quantity $P(c|l_i)$ is highlighted for discriminative terms, i.e. if $P(c|l_i) > 0.75$.

earn (3,964 total)			crude (578 total)			ship (286 total)		
term	$P(l_i\|c)$	$P(c\|l_i)$	term	$P(l_i\|c)$	$P(c\|l_i)$	term	$P(l_i\|c)$	$P(c\|l_i)$
"cts"	73.4	**93.4**	*"oil"*	91.2	45.4	*"ship"*	47.9	62.6
"net"	66.9	**88.9**	*"barrel"*	50.0	**82.6**	*"shipping"*	38.1	67.3
"mln"	62.7	40.0	*"crude"*	48.1	**77.9**	*"oil"*	35.7	8.8
"shr"	59.7	**99.4**	*"dlrs"*	46.2	4.9	*"gulf"*	32.9	42.3
"qtr"	51.6	**99.1**	*"price"*	43.9	10.9	*"port"*	30.8	50.3

belongs to the respective class.[4] It is readily verified that the category *"earn"* can be described very well in terms of a few frequent and highly discriminative terms. For example, the terms *"cts"* and *"net"* appear in more than two out of three positive documents and are at the same time highly discriminative at conditional probabilities for the class given the terms at 93.4% and 88.9% respectively. As another extreme, the occurrence of terms is substantially less concentrated for documents of the category *"ship"*. Here, even the two most frequent terms occur in less then half of the documents while their discriminative power is only slightly above parity. The category *"crude"* can be seen as in the middle of the two. Here, we still find two terms with somewhat substantial discriminative power, *"barrel"* and *"crude"*, but they appear at most in every second document.

[4]Of course, this is a simplifying analysis as we only focus on the *positive* classification decisions while neglecting terms that are indicators of the *negative* classification. Obviously, these terms can also substantially aid the overall classification, especially in cases of non-overlapping classes. However, the analysis should suffice to illustrate the main points of the argument. The analysis also seems to be in line with the more comprehensive analysis of the *"earn"* category by Joachims (2001, 2002).

6.2.2 Experimental Setup

Focus of the Experiments

The popularity of the REUTERS-21578 collection mainly stems from its perception as an inhomogeneous, partly easy and partly challenging collection for text classification. Furthermore, its application context, the automated dispatching of news documents, exhibits substantial business relevance. These considerations and the characteristics of the dataset discussed above have shaped the focus of the experiments. In particular, the experiments in this section aim at investigating the performance of Semantic Smoothing Kernels in the light of the following aspects:

Small Datasets Based on the reasoning in Section 5.1.3, we expect that the introduction of prior semantic knowledge typically has a small effect when sufficient training data is available and, in situations of too much noise (e.g. as a result of word sense ambiguity effects), might even degrade the performance compared to the VSM kernel. Our experiments are thus restricted to quantifying performance gains in those cases where little training data is available.

Heterogeneous Target Classes Documents in categories like the *"earn"* category, which can be characterized almost undoubtedly by terms which occur with high frequency do not to utilize the full variability natural language offers. On the other hand, there are categories which do so. For example, for the category *"ship"*, the terms *"vessel"*, *"tanker"* or *"supertanker"* are highly indicative but occur only in the mid-frequency range. The reduction of the available training data will thus affect performance more in the latter case, a situation where SSKs are likely to exhibit most of their power. In the subsequent analysis, we thus undertake not only a global, but also a per-category analysis of the results.

Classical VSM Setting For all experiments, we focus on a setting that is as close as possible to the classical VSM setting. In particular, we avoid extensive linguistic preprocessing.

Preprocessing

The VSM representation of the REUTERS-21578 documents was generated based on the standard preprocessing steps, namely (i) single-word tokenization, (ii) removal of the standard stopwords for English defined in the SMART stopword list, and (iii) dictionary-based lemmatization. For lemmatization, we have used the lemmatization lexicon of the statistical parser LOPAR (Schmid, 2000).[5] This indexing step resulted in a total number of 23,754 distinct term features. All term features were weighted according to the standard TFIDF scheme.

[5] Available at `http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/LoPar.html`.

Table 6.2 — Document statistics for the REUTERS-21578 corpus. The first line (total (+/-)) lists the total number of documents per subset of the corpus. The subsequent lines list the number of positive training documents per category and subset.

	Training											**Test**
subset	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%	100%	100%
total (+/-)	96	192	288	384	480	576	672	768	864	960	9,603	3,299
earn (+)	29	58	86	115	144	173	201	230	259	288	2,877	1,087
acq (+)	17	33	50	66	83	99	116	132	149	165	1,650	719
money-fx (+)	5	11	16	22	27	32	38	43	48	54	538	179
grain (+)	4	9	13	17	22	26	30	35	39	43	433	149
crude (+)	4	8	12	16	19	23	27	31	35	39	389	189
trade (+)	4	7	11	15	18	22	26	30	33	37	369	117
interest (+)	3	7	10	14	17	21	24	28	31	35	347	131
wheat (+)	2	4	6	8	11	13	15	17	19	21	212	71
ship (+)	2	4	6	8	10	12	14	16	18	20	197	89
corn (+)	2	4	5	7	9	11	13	14	16	18	181	56

Training Subsets

As we aim to compare the performance in those cases where little training data is available, we prepared small subsets of the ModeApte training set. For each category, we randomly chose 1% - 10% of the positive training documents and the same fraction of negative training documents. To account for the inherent sampling variance, this approach was repeated 10 times for each of the 10 subset sizes resulting in a total number of 100 different training subsets per category. Table 6.2 summarizes the distribution of positive training documents per category and subset. Note that we have chosen fixed percental subsets of the original training corpus to mirror the original structure. For the analysis of the experiments, however, it is important to keep in mind that the uneven distributions of positive documents is thus also reflected in the subsets.

Smoothing Parameters

For all experiments, we used the noun hierarchy of the lexical database WORDNET (Miller et al., 1990; Miller, 1995) described in Example 4.1 as the taxonomic backbone for our experiments. For the setup of the smoothing matrices as discussed in Section 5.4 we used two variants:

- Unnormalized taxonomic overlap of superconcepts (c.f. Definition 4.14) and weighted taxonomic overlap of superconcepts (c.f. Definition 4.15), whereby we used the popular path- and information content (IC)- based measures of semantic similarity of Sections 4.3.1–4.3.4 as weighting schemes.

- Cosine-normalized variants of these with an additional term of "1" on the main diagonal of the resulting smoothing matrix **S** as discussed in Section 5.2.4.

Table 6.3 on page 139 summarizes the descriptions and coding of these configurations as well as those used in any of the other experiments. Frequency counts needed for the calculation of the measures making use of IC were obtained from the complete REUTERS-21578 collection. Furthermore, as discussed in Section 5.2.1, for mapping between terms and senses, we used a simple disambiguation strategy $\delta : \mathcal{L} \rightarrow \mathcal{C}$ whereby each term is mapped to its most frequent noun sense according to the usage statistics provided by WORDNET. Note that this approach implies an inherent word sense disambiguation side effect, both with respect to the respective Part–of–Speech as well as to the chosen noun sense. While this effect is likely to have a negative impact on the results, the error introduced by this approach is systematic. In the light of these considerations, the results can also be seen as a pessimistic estimate of the potential effectiveness given a perfectly disambiguated input. All pairs of entries for which this mapping failed were implicitly assumed to take the default values (i.e. zero and one for off-diagonal and diagonal entries respectively) during kernel evaluation. Note that this procedure (a so-called zero-extension of the smoothing matrix) always leaves the smoothing matrix in a positive semi-definite state.

6.2.3 Results

Binary classification experiments were then conducted for each category and each subset resulting in a total number of 1,000 individual experiments for each SSK configuration. While in each of these experiments the SVM classifier was trained using the respective subset, the corresponding testing was conducted using the full ModeApte test set as specified in Table 6.2. The soft margin parameter C that controls the influence of misclassified examples as discussed in Section 2.3.2 was set to $C = 0.1$ in all experiments. This decision was motivated by the findings in preliminary experiments that its variation had no or only minor effects for the results obtained on the subsets. This can be explained by the fact that all subproblems are linearly separable such that the value of C does not really affect the optimization problem of the soft-margin SVM.[6] In this section, we report on the results of the classification experiments.[7]

[6]Note that this finding is consistent with the analysis by Joachims (2002) which indicates that even on the *full* training data all but three of the 10 largest REUTERS-21578 categories are linearly separable.

[7]Note that the results reported in this section slightly differ from earlier experiments (Bloehdorn et al., 2006a,b). This fact needs to be attributed to different preprocessing which did not remove number

Table 6.3 — Configurations of smoothing parameters used in the experiments in this chapter. Descriptions of each smoothing parameter configuration together with its respective short code used in the remainder.

Short name	Description
vsm (string)	Term identity only – no smoothing, i.e. corresponding to plain dot products in the case of SSKs.
hyp–full	Smoothing similarities as unnormalized taxonomic overlap of superconcepts, c.f. Definition 4.14.
hyp–resnik	Smoothing similarities as weighted taxonomic overlap of superconcepts, c.f. Definition 4.15. Weighting between the reference concept and the respective superconcept via the similarity measure of Resnik (1999).
hyp–lin	As above, but using the similarity measure of Lin (1998).
hyp–wupalmer	As above, but using the similarity measure of Wu and Palmer (1994).
hyp–path1	As above, but using the inverted path length similarity measure with a decay factor of $\alpha = 1$.
hyp–path2	As above, but using the inverted path length similarity measure with a decay factor of $\alpha = 2$.
simsq–resnik	Squared smoothing similarities according to the similarity measure of Resnik (1999).
simsq–lin	As above, but using the similarity measure of Lin (1998).
simsq–wupalmer	As above, but using the similarity measure of Wu and Palmer (1994).
simsq–path1	As above, but using the inverted path length similarity measure with a decay factor of $\alpha = 1$.
simsq–path2	As above, but using the inverted path length similarity measure with a decay factor of $\alpha = 2$.
{}–norm*	Suffix indicating that any of the above configurations is modified by using normalized smoothing measures according to the cosine normalization.
{}–add1*	Suffix indicating that any of the above configurations is modified by increasing the weight of pairs of matching terms by increasing the values on the main diagonal of the smoothing matrix by 1.0 (c.f. Section 5.2.4).

Global Analysis

We first analyze the results on the full dataset from a birds-eye perspective in terms of the absolute macro F_1 values obtained over the different subsets of REUTERS-21578. These results are summarized in Table 6.4. The choice of the macro-averaging scheme is motivated by the fact that, due to the grossly uneven class distribution, micro-averaged scores would be dominated entirely by the largest categories *"earn"* and *"acq"*. We will account for the differences between the different classes by means of a differentiated analysis in the next section.

As we can see in table Table 6.4, all SSKs can achieve substantial improvements over the VSM baseline for the smallest subsets. Consistent with out prior analysis, the improvement diminishes as more training data becomes available and sometimes deteriorates compared to the VSM baseline.

For the first group of (unnormalized) SSK parameters, we find the largest improvements for the small datasets but at the same time, these settings are typically more unstable as more training data becomes available. The *hyp-full* scheme is most problematic in this respect. While it achieves good improvements for 1%–3%, results fall below the VSM baseline starting from 5%. The situation is similar for the *hyp-resnik* scheme which does not show any improvements starting from 7%. In both cases the mixed performance is most likely to be due to the fact that for *hyp-full* no weighting scheme is employed at all, thus leading to identical similarities for two terms just below their common Lowest Super Ordinate (LSO) and for two terms which also share this LSO but are situated far apart. This property is shared by the *hyp-resnik* scheme which also neglects the distance of a superconcept to the reference concept even though it attributes less weight to more general concepts far up in the taxonomy.

The remaining four schemes, which employ more fine-grained weighting schemes behave differently. For these configurations, the relative improvement over the *vsm* baseline is depicted in Figure 6.2. The *hyp-wupalmer* scheme accounts for the distances between a superconcept and the reference concept such that it achieves good results up to the 8% level. The most successful scheme for the smaller training sets is the *hyp-lin* scheme which is taken over by the *hyp-path1* and *hyp-path2* after the 5% level. This finding confirms the intuitive assumption on the desired structure of the weighting scheme as the *hyp-lin* scheme respects both the overall depth of the respective superconcept by virtue of the information content as well as the distance from the base concept by means of the difference in information content. For the interpretation of the *hyp-path1* and *hyp-path2* schemes it seems important to note that the superconcept weights decrease very fast, thus leaving most of the contribution to the pairwise similarities to superconcepts in the immediate neighbourhood of the reference terms. Correspondingly, the resulting smoothing matrices have shown sub-

items and to the different sampling strategy for the subsets. The general structure of the results is, however, consistent.

Table 6.4 — Macro-averaged F_1 results for different REUTERS-21578 subsets and different Semantic Smoothing Kernel configurations. All numbers are percentages. The three best results per subset are highlighted.

Configuration	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
vsm	29.5	40.7	49.5	54.5	57.9	60.6	63.9	65.3	67.0	68.5
hyp-full	36.4	44.7	51.2	55.0	57.4	59.7	61.9	62.9	64.6	65.4
hyp-resnik	**37.6**	**46.8**	53.0	56.8	59.3	61.2	63.2	64.0	65.4	65.8
hyp-wupalmer	**37.5**	**46.7**	**53.4**	**57.6**	60.3	62.5	64.6	65.5	66.9	67.7
hyp-lin	**39.8**	**48.5**	**54.9**	**59.0**	**61.5**	**63.5**	65.4	66.3	67.7	68.4
hyp-path1	34.6	46.3	**53.7**	**58.5**	**61.8**	**64.5**	**66.9**	**67.6**	**69.1**	**70.0**
hyp-path2	31.9	44.3	53.1	**57.6**	**61.2**	**63.7**	**66.5**	**67.6**	**69.4**	**70.4**
hyp-full-norm-add1	31.3	41.8	49.5	55.1	58.7	61.2	64.1	65.1	66.6	67.7
hyp-resnik-norm-add1	32.6	44.0	51.7	56.7	60.2	62.8	65.6	66.8	68.1	69.4
hyp-wupalmer-norm-add1	32.2	43.5	51.1	56.2	60.0	62.5	65.4	66.4	67.7	68.9
hyp-lin-norm-add1	32.4	43.4	50.9	56.0	59.7	62.3	65.3	66.2	67.6	68.7
hyp-path1-norm-add1	32.1	44.0	51.8	56.9	60.8	63.4	**66.4**	**67.6**	**69.0**	**70.3**
hyp-path2-norm-add1	30.4	42.7	51.6	56.2	59.9	62.6	65.5	66.8	68.7	69.9

stantially smaller similarity values for most pairs of terms. As a result, these schemes still allow to relate strongly related terms, but limit the effects of noise for weakly related terms. In contrast to the other schemes, the structure of the original VSM model is largely maintained. This analysis also explains that the improvements are small in case of little training data but are still maintained in situations of more training data.

For the second group of SSK parameters, where the smoothing parameters are normalized via the (nonlinear) cosine normalization and are implicitly complemented by the plain inner product on the original VSM, the situation is different. Except for the *hyp-full-norm-add1* scheme, which does not always improve compared to the plain VSM representation, all schemes consistently lead to improved results compared to the plain VSM setting. However, the improvements are in all cases less distinctive than for the unnormalized setting. While the unnormalized measures achieved absolute improvements between 2.4% (*hyp-path2*) and 10.4% (*hyp-lin*) for the 1% subset and up to 1.9% (*hyp-path2*) for the 10% subset, the improvement in the second group of measures always ranges between 1.0% and 3.3%. Similar to the analysis for the unnormalized *hyp-path1* and *hyp-path2* schemes, this can be explained by the fact that all variants are much "closer" to the original VSM representation.

Table 6.5 — F_1 results for the REUTERS-21578 categories *"earn"*, *"crude"*, *"grain"* and *"ship"*, different subsets and different Semantic Smoothing Kernel configurations, averages over 10 samples. All numbers are percentages. Results marked with one star are significantly better than the corresponding VSM baseline according to the (one-sided) paired T-Test at a significance level of $\alpha = 0.05$ while two stars indicate the same for $\alpha = 0.01$.

	"earn"			*"crude"*		
Configuration	1%	5%	10%	1%	5%	10%
vsm	90.8	95.3	96.0	19.4	62.2	72.3
hyp-full	83.3	92.7	94.3	**41.4	64.1	69.1
hyp-resnik	82.7	88.8	89.4	**40.8	*65.7	70.0
hyp-wupalmer	86.6	94.0	95.0	**40.2	*65.9	71.2
hyp-lin	87.4	94.1	95.1	**44.5	*67.2	71.7
hyp-path1	90.6	*95.5	96.3	**31.7	**66.5	72.4
hyp-path2	*91.6	**95.6	**96.3	**23.2	**64.8	72.0
hyp-full-norm-add1	90.3	95.1	96.2	**27.1	62.6	71.3
hyp-resnik-norm-add1	91.4	**95.5	*96.4	**24.6	63.7	71.9
hyp-wupalmer-norm-add1	91.2	95.4	*96.4	**25.7	63.5	72.1
hyp-lin-norm-add1	91.1	95.3	**96.4	**25.5	63.7	71.8
hyp-path1-norm-add1	91.8	**95.9	**96.5	**24.6	**64.6	72.5
hyp-path2-norm-add1	*91.5	**95.7	**96.3	**21.2	**64.0	72.5

	"grain"			*"ship"*		
Configuration	1%	5%	10%	1%	5%	10%
vsm	24.4	65.1	75.1	1.1	16.8	43.2
hyp-full	**44.8	68.0	75.7	**17.6	**39.9	**53.4
hyp-resnik	**49.5	*70.4	76.7	**10.6	**40.2	**55.4
hyp-wupalmer	**47.8	**71.1	*77.0	**13.8	**40.2	**54.5
hyp-lin	**52.4	**72.2	**78.2	**13.2	**42.3	**56.3
hyp-path1	**37.8	**72.0	**78.4	*5.2	**32.1	**51.7
hyp-path2	**28.8	**69.9	**77.6	1.5	**22.1	**47.4
hyp-full-norm-add1	**32.9	**67.7	75.8	**8.0	**31.3	**49.6
hyp-resnik-norm-add1	**34.9	**69.8	**77.7	2.8	**27.5	**49.8
hyp-wupalmer-norm-add1	**34.3	**69.6	**76.9	*4.2	**29.9	**50.2
hyp-lin-norm-add1	**34.6	**69.3	**76.9	*3.8	**29.2	**49.9
hyp-path1-norm-add1	**30.4	**69.2	**77.7	2.4	**26.2	**49.9
hyp-path2-norm-add1	**25.8	**67.8	**76.9	1.5	**19.7	**45.8

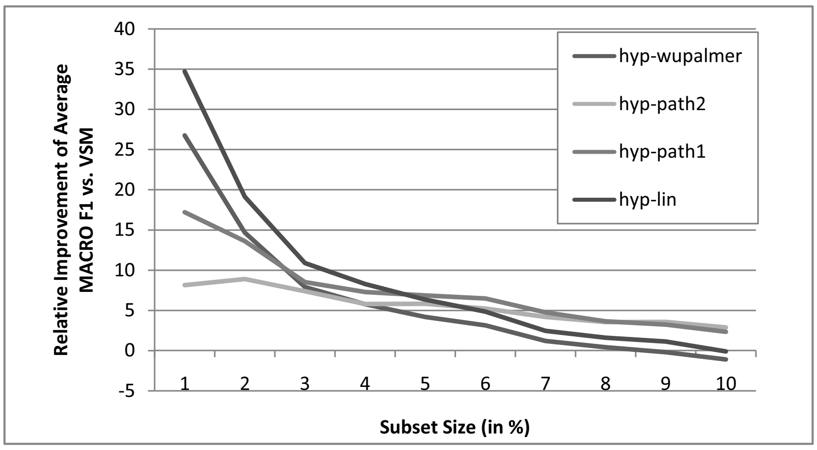


Figure 6.2 — Relative improvement of macro F_1 values on REUTERS-21578 subsets for Semantic Smoothing Kernels based on the *hyp-lin*, *hyp-wupalmer*, *hyp-path1* and *hyp-path2* schemes versus the *vsm* baseline.

Per-Category Analysis

The analysis in the previous section has focused on the averaged performance over all categories for the respective subsets. All categories were treated as equally relevant, regardless of the actual number of positive training documents and the specific characteristics of the individual classes. While this analysis allows to draw conclusions on the performance from the macro perspective, it blurs the substantial differences between the individual classes which we have noted in the prior analysis. This section thus provides on a more detailed analysis of the behaviour of the different classes. For space reasons, we focus on four exemplary categories. While each of the 10 classes has different characteristics and exhibits different results, these classes can be seen as roughly representative in terms of the distributions in the number of training documents and in their behavior results. Specifically, these are the categories *"earn"* (many positive training documents, poor performance), *"crude"* (medium number of positive training documents, performance varies between strong and poor for subsets), *"grain"* (medium number of positive training documents, good performance for all subsets), and *"ship"* (few positive training documents, very good performance for all subsets).

For each of these categories, Figure 6.3 plots the F_1 values achieved by the Semantic Smoothing Kernels on all subset samples against the VSM results on the same sample. Table 6.5 summarizes the macro-averaged results F_1 for these categories and the 1%, 5% and 10% subsets.

The *"earn"* category achieves fairly high F_1 values of over 90% even for the 1%

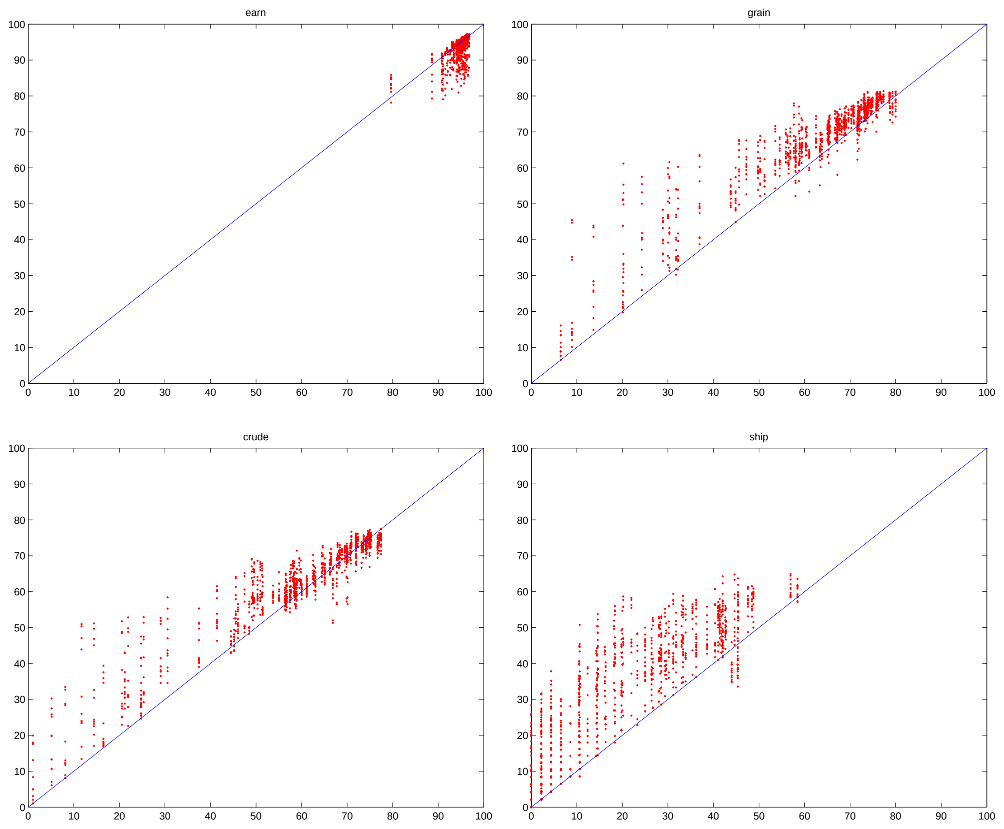


Figure 6.3 — Scatterplot of F_1 values on selected REUTERS-21578 categories. F_1 values for all SSKs configurations, subsets and samples are plotted against the corresponding F_1 value for the results of the plain VSM representation on the corresponding sample.

subset. One the one hand, this is certainly due to the large fraction of positive training examples. On the other hand, it also complies with the earlier analysis of indicative terms for this category. On this category, is generally tough for SSKs to compete against the VSM baseline. For the unnormalized smoothing parameters, only the path-based weighting schemes can compete with the VSM baseline while all other schemes even degrade performance. For the normalized measures with extra VSM weights, most SSKs achieve only slight (tough nevertheless sometimes statistically significant) improvements or assume similar values.

The situation changes for the *"crude"* category. Here, at the 1% subset, all SSK configurations yield substantial and significant improvements, especially for the "crisp", unnormalized smoothing parameters. For the 5% subset, the improvements begin to diminish but remain significant in many cases. At the 10% level, no SSK configuration can beat the VSM baseline any more. It appears that at this level, the number of training documents is sufficient to equal out any variation in the used terminology.

The categories *"grain"* and *"ship"* generally appear amenable for Semantic Smoothing Kernels. For *"grain"*, we find improvements for all subsets and all SSK configurations, in most cases substantial and in almost all cases statistically significant. The same situation applies for *"ship"*, even though the general level of the F_1 values for VSM representation and SSKs is very low, especially for the very small subsets. Due to the extremely small number of training documents in the 1% subset, these results should be considered with care.

6.3 Experiments with Semantic Smoothing Kernels on AMAZON Data

In a second set of experiments, we analyze the behaviour of SSKs for the classification of editorial reviews of scientific books according to the covered topic on the AMAZON collection (Ifrim and Weikum, 2006).

6.3.1 Task and Dataset

Description of the Dataset

The AMAZON dataset was compiled by Ifrim and Weikum (2006) and contains editorial reviews of books for sale by the online book retailer AMAZON.COM. The texts are organized into classes according to the categorization provided by AMAZON.COM. The dataset contains a total of 5,634 editorial reviews for books in the three categories *"Biology"* (2,047 reviews), *"Mathematics"* (2,258 reviews), and *"Physics"* (1,329 reviews).

Training vs. Test Data

The dataset does not provide a breakdown into training and test documents. We thus randomly selected 500 documents of each class which together formed the test collection for the experiments, the remaining documents formed the corresponding training collection.

Characteristics of the Dataset

The AMAZON dataset differs from the REUTERS-21578 dataset in various aspects. On the one hand, the vocabulary is richer and the language ambiguity is higher. On the other hand, the different categories are much more homogeneous with respect to their characteristics. While in the REUTERS-21578 dataset, as discussed above, various classes can be identified solely on the basis of a small number of discriminative keywords which occur frequently, all classes in the AMAZON dataset are characterized by a large number of topic-related scientific terms which occur at medium or low frequency. For example, the category *"mathematics"* is characterized sufficiently well by means of terminology as diverse as *"calculus"*, *"Fourier analysis"*, *"geometry"*, *"set theory"* etc. Clearly, all of these are branches of mathematics and their similarity is reflected by their neighbourhood in the WORDNET noun hierarchy.

6.3.2 Experimental Setup

Focus of the Experiments

In this experiment, we aim at investigating the performance of Semantic Smoothing Kernels in the light of the following aspects:

Small Datasets Similar to the previous experiments on the REUTERS-21578 dataset, we again aim at quantifying performance gains in settings where little training data is available.

Comparison of Different SSK Paradigms While we have investigated smoothing parameters based on the taxonomic overlap of superconcepts, we this time aim to also compare this representation to the performance of SSKs that use squared matrices of the semantic similarities defined between all terms.

Consideration of Compound Terms The AMAZON dataset is characterized by a substantial frequency of compound expressions (e.g. *"computer science"*) with distinct meaning. As such expressions are usually not affected by lexical ambiguity when mapping to term senses, we this time choose to use this kind of representation.

Preprocessing

We prepared the VSM representation of the documents based on the standard preprocessing steps. After initial tokenization we used the WORDNET dictionary and morphology module in conjunction with a sliding window of length 2 to detect compound expressions and to lemmatize the expressions. Standard stopwords for English defined in the SMART stopword list were again removed. This indexing step resulted in a total number of 38,530 distinct terms. All term features were weighted according to the standard TFIDF scheme.

Training Subsets

As we want to quantify performance gains in settings where very little training data is available, we again prepared subsets of the overall training. For each category, subsets of different sizes were prepared. The subsets were built by randomly choosing $5, 10, \ldots, 50$ as well as $100, 200$ documents of each category, whereby the target category contributed the positive training documents and the remaining two categories contributed the nagative training documents. This means that when we refer to a subset size, e.g. to the "10" subset, this corresponds to the subset of 3 times 10 documents ("3×10"). Like in the previous experiments, this technique was repeated 10 times for each subset size and for each target category to minimize random effects. Overall this procedure thus results in 80 training subsets per category. We further prepared a *single* sample of 500 documents per category.

Smoothing Parameters

As in the case of the REUTERS-21578 experiments, we use the noun hierarchy of the lexical database WORDNET (c.f. Example 4.1) as the taxonomic backbone. For the setup of the smoothing matrices as discussed in Section 5.4 we used two variants:

- Unnormalized taxonomic overlap of superconcepts (c.f. Definition 4.14) and weighted taxonomic overlap of superconcepts (c.f. Definition 4.15), whereby we used the popular path- and IC- based measures of semantic similarity of Sections 4.3.1–4.3.4 for weighting.

- Squared smoothing matrices of the popular path- and IC- based measures of semantic similarity discussed in Sections 4.3.1–4.3.4.

Frequency counts needed for the calculation of any of the similarity measures making use of IC were this time obtained from the Brown corpus (Kucera and Francis, 1967). Recall that Table 6.3 summarizes the descriptions of the smoothing matrix configurations used in the experiments. Again, we used the simple (and noisy) disambiguation heuristic of choosing the most frequent term sense for a given linguistic expression in the index. As the corpus contains a substantial fraction of terms which occur only in very few documents, we further restricted the off-diagonal matrix entries to terms which occur at least five times in the overall corpus. All pairs of entries which were ignored this way or for which the sense mapping failed were again assumed to take the default values (i.e. zero and one for off-diagonal and diagonal entries respectively) during kernel evaluation.

6.3.3 Results

Binary classification experiments were then conducted for each category and each subset resulting in a total number of 243 experiments for each configuration. In each run, the SVM classifier was trained using the respective subset and the corresponding testing was conducted using the full test set. The soft margin parameter C that controls the influence of misclassified examples as discussed in Section 2.3.2 was set to $C = 0.1$ in all experiments. Again, this decision was based on preliminary experiments that showed that its variation in the common $[0.1, 10]$ range had little effect for the obtained results, even though a perfect linear separation was not always possible.

Global Analysis

For each of the investigated configurations, Table 6.6 summarizes the absolute macro F_1 values obtained over the different subsets of the AMAZON dataset. The choice of the macro averaging scheme is motivated by the fact that the number of positive training examples is identical for all three target classes. As the three target classes are, in contrast to the REUTERS-21578 experiments, more uniform in their characteristics and the training subsets are chosen to contain equal numbers of all classes, we also include the results of the paired T-Test on an improvement of the macro-averaged F_1 based on the 10 samples per subset size and configuration (except for the 3×500 subset which was sampled only once).

Again we note that all types of SSKs achieve substantial and statistically significant improvements over the VSM baseline for the smallest subsets of up to 50 example documents per class (i.e. a dataset of 150 documents). Similar to the previous experiment on REUTERS-21578, the improvement diminishes as more training data becomes available and eventually deteriorates in some of the cases when compared to the VSM baseline.

However it is again crucial to distinguish the different types of configurations. When considering the representations based on the (weighted) taxonomic overlap of superconcepts we note that the *hyp-full* scheme performs well in the beginning but falls behind the competing configurations already at the 3×10 subsets. The *hyp-resnik* schemes is among the best performing weighting schemes until the 3×100 subset. The *hyp-lin* scheme and, slightly lagging behind, the *hyp-wupalmer* schemes can achieve performance gains up to the 3×200 subsets. The *hyp-path1* and *hyp-path2* schemes show only moderate tough still statistically significant gains compared to the VSM baseline at smaller subset sizes but remain stable also in situations of more available training data.

The results of SSKs based on squared full similarity matrices seem to support our earlier analysis that this kind of representation is likely to introduce too much noise and is thus inadequate in the general case. While the initial results on the 3×5

Table 6.6 — Macro-averaged F_1 results for different AMAZON subsets and different Semantic Smoothing Kernel configurations. All numbers are percentages. For subsets of 5 documents per category until 200 documents per category, values are macro averages over all classes and samples. Results marked with one star are significantly better than the corresponding VSM baseline according to the (one-sided) paired T-Test at a significance level of $\alpha = 0.05$ while two starts indicate the same but for the tighter significance level of $\alpha = 0.01$. For the subset of 500 documents per category, values are macro-averages over all classes (only one sample, no significance testing).

Configuration	5	10	20	30	40	50	100	200	500
vsm	16.7	24.9	39.9	50.2	58.0	63.0	73.5	78.9	82.0
hyp-full	**46.9	**53.3	**61.3	**66.3	**68.9	**71.7	**76.4	78.9	80.3
hyp-resnik	**44.2	**55.9	**64.0	**69.4	**72.3	**73.8	**77.6	78.8	72.4
hyp-wupalmer	**45.5	**54.4	**63.1	**68.4	**71.1	**73.4	**77.8	**80.0	81.4
hyp-lin	**44.2	**54.3	**63.4	**68.7	**71.4	**73.6	**77.9	**80.1	81.4
hyp-path1	**36.2	**50.0	**61.0	**67.5	**71.0	**73.6	**78.9	**81.5	82.5
hyp-path2	**20.5	**32.9	**50.6	**60.3	**66.5	**70.2	**77.3	**81.0	82.2
simsq-resnik	**46.8	**48.7	**56.8	**61.9	**64.7	*67.3	70.5	65.1	55.3
simsq-wupalmer	**46.1	**48.2	**55.9	**60.5	*63.4	*66.9	72.8	69.3	46.0
simsq-lin	**46.9	**48.4	**56.8	**61.3	**64.3	**67.6	73.4	76.4	56.6
simsq-path1	**48.6	**55.2	**64.1	**68.4	**70.9	**73.5	**78.2	77.5	40.7
simsq-path2	**41.0	**53.4	**63.5	**69.3	**72.1	**74.3	**78.2	77.5	82.3

subsets are a bit more favorable than for the smoothing parameters based on the (weighted) taxonomic overlap of superconcepts, the *sim-resnik*, *sim-wupalmer*, and *sim-lin* schemes fail to hold up this margin and begin to deteriorate. Nevertheless they show improvements compared to VSM and deteriorate only at subset sizes of 3×100 and above. The *sim-path1* and *sim-path2* schemes are consistently better than the more advanced weighting schemes but show the same general pattern compared to VSM and the smoothing parameters based on the (weighted) taxonomic overlap of superconcepts.

6.4 Experiments with Semantic Syntactic Tree Kernels for Question Classification

While the last two experiments have analyzed the behaviour of Semantic Smoothing Kernels, we now switch to Semantic Syntactic Tree Kernels. question answering (QA) has a long history within the framework of the TREC (Text Retrieval Conference) evaluation series.[8] This long tradition has produced a large question set used by several researchers which can be exploited for experiments on question classification (QC) (Metzler and Croft, 2005). As we will see, the setting of question classification is particularly sensitive to the syntactic structure of the input data which makes this setting interesting for the use of Semantic Syntactic Tree Kernels. In fact, tree kernels have shown to be effective means for question classification (Zhang and Lee, 2003; Moschitti, 2004; Quarteroni et al., 2007; Li and Roth, 2006) and main focus of the experiments will be to investigate the additional effects obtained through the use of Semantic Syntactic Tree Kernels.

6.4.1 Task and Dataset

Description of the Question Classification Task

QC (Li and Roth, 2002) aims at detecting the type of a question, e.g. whether the question asks for a person or for an organization which is critical to locate and extract the right answers in QA systems. According to (Li and Roth, 2002), we can define QC *"to be the task that, given a question, maps it to one of k classes, which provide a semantic constraint on the sought-after answer"*. A major challenge of QC compared to standard text classification settings is that questions typically contain extremely few words which make this setting a typical victim of data sparseness. At the same time, the syntactic structure of the sentence determines the relation between the statements in the sentence to the entity referred to by the *"Wh"*-question word. As this sough-after entity determines the classification of the question, the syntactic relations within

[8] `http://trec.nist.gov`

(SBARQ (WHNP (WP What)) (SQ (VP (VBP are) (NP (NP (DT the) (NNS names)) (PP (IN of) (NP (NP (DT the) (NN tourist) (NNS attractions)) (PP (IN in) (NP (NNP Reims)))))))) (. ?))

(SBARQ (WHNP (WP What)) (SQ (VP (VBZ 's) (NP (NP (DT the) (JJ only) (NN work)) (PP (IN by) (NP (NNP Michelangelo))) (SBAR (WHNP (WDT that)) (S (VP (VBZ bears) (NP (PRP his) (NN signature)))))))) (.?))

Figure 6.4 — Examples of Question Parse Trees from the TREC QC Dataset.

the sentence provide useful classification information and need to be considered adequately.

Description of the Dataset

The experiments in this section are based on a dataset that has initially been employed by authors like Li and Roth (2002) and Zhang and Lee (2003) and is freely available.[9] The free-text questions in the dataset are categorized according to different taxonomies of different granularities. The experiments reported on in this section consider the coarse grained classification scheme as described by Li and Roth (2002), consisting of 6 classes: *"Abbreviations"*, *"Descriptions"*, *"Entity"*, *"Human"*, *"Location"*, and *"Numeric"* (e.g. codes or dates).

Two examples of Question Parse Trees from the TREC QC Dataset are given in Figure 6.4. The example parse tree in Figure 5.2 on page 111 is also taken from this dataset. Again, refer to Appendix B for the full list of used annotations.

Training vs. Test Data

We use the standard benchmark setting devised by Li and Roth (2002). The training set consists of 5,500 questions which taken from the 4,500 English questions published by USC (Hovy et al., 2001), about 500 questions provided by Li and Roth (2002) and 894 questions from TREC 8 and TREC 9. The test set consists of 500 TREC 10 questions for testing.

6.4.2 Experimental Setup

Preprocessing

The reference term dictionary was compiled by tokenizing all questions sentences. To allow for a fair comparison the preprocessing was modified compared to the standard

[9] `http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`

VSM representation by *not* remove any stopwords. This decision is motivated by the intuitive finding that, as opposed to the standard topic-based classification, function words like *"What"* or *"Who"* are highly informative for the question classification task. Further, as in the experiments of Zhang and Lee (2003), terms were not lemmatized to their base forms for the VSM baseline. Note that we will shortly discuss the results for alternative features which are reported in literature (Li and Roth, 2002; Zhang and Lee, 2003) later on. Altogether, this preprocessing led to a total number of 8,410 distinct terms. For this VSM representation, the vector components were again weighted according to the standard TFIDF scheme. The question parse trees needed for the tree kernel based representations were obtained by running the parser developed by Charniak (2000).[10]

Smoothing Parameters

As in the previous experiments, we used the noun hierarchy of the lexical database WORDNET as the taxonomic backbone to compute the smoothing parameters for the Semantic Syntactic Tree Kernels. For the setup of the smoothing matrices as discussed in Section 5.4 we used one setting:

- Cosine-normalized variants of the taxonomic overlap of superconcepts (c.f. Definition 4.14) and of the weighted taxonomic overlap of superconcepts (c.f. Definition 4.15), whereby we used the popular path- and IC- based measures of semantic similarity of Sections 4.3.1–4.3.4.

Again recall that Table 6.3 summarizes the descriptions and coding of these configurations as well as those used in any of the other experiments. As in the previous experiments we used the mapping of terms to the most frequent noun sense associated with their base form as disambiguation heuristic. Statistics for calculating the information content were this time obtained from the Brown corpus (Kucera and Francis, 1967) as word frequency estimations on the TREC QC dataset itself would be rather unreliable due to its far smaller overall size. Again, pairs of smoothing parameters that were undefined were assumed to take the default values (i.e. zero for distinct and one identical terms respectively).

6.4.3 Results

The experimental setup was identical to the setup of Zhang and Lee (2003) as it contains the most comprehensive comparison of experiments on the TREC QC corpus. Binary classification experiments were conducted on each of the 6 question types of the coarse-grained classification scheme described above. To be able to compare with

[10] `http://www.cs.brown.edu/people/ec/#software`

the literature results, the binary classifiers were further combined according the one-vs-all multi-classification scheme. This means that for a given question, besides the classifications vs. all classes, a single class is selected for which the corresponding test instance produces the highest margin score of the binary SVMs. In all experiments, $C = 1$ was used as soft-margin parameter. Furthermore, the best cost-factor for positive vs. negative classifications in SVMs (Morik et al., 1999; Joachims, 2002) was preliminary determined on a small validation set.

The experiments compare plain kernels based on the VSM representation, the original tree kernel and a set of SSTK configurations with different term similarities. Based on the findings by Zhang and Lee (2003) and Quarteroni et al. (2007), the normalized tree kernels were in all cases additively combined with normalized VSM kernels. Furthermore, for the tree kernels, different values of the decay parameter λ and the leaf contribution parameter α (c.f. Section 5.3.5) were considered.

Analysis of the Results

For all configurations, this section reports on the F_1 scores for the binary classifiers. As the target classes are unevenly distributed in the test data, but at the same time there is are no class which would dominate the result, the reported averages are based on the micro-averaging scheme. Furthermore, to be able to compare with literature results, we also report the adjusted single-label accuracy after one-vs-all processing (c.f. Section 2.5.2) as eventually only one dedicated class should be assigned to a given question.

Table 6.7 reports the results of the experiments. The first column indicates the value of the α parameter of the tree kernel discussed in Section 5.3.5. The second column shows the type of term similarity kernel used together with the kernel function of SSTKs, whereby *tree kernel* indicates that the original tree kernel formulation is used. The remaining columns report the micro-averaged F_1 and the adjusted single-label accuracy.

While the variation of the λ parameter seems to have a minor importance, the improvement of the tree kernels seems to be largely consistent. We can note that default tree kernels can achieve up to 91.6% of multi-classification accuracy (for $\alpha = 2$). The above values can be improved considerably when we employ term similarity kernels.

In all cases, the additional semantic information in Semantic Syntactic Tree Kernels improves performance whereby the improvement is highest for the *hyp-wupalmer* scheme. In particular, the tree kernel based on the *hyp-wupalmer* similarity and parameters $\alpha = 2$ and $\lambda = 0.05$ achieves the highest multi-classification accuracy, i.e. 93.6%. For the α parameter, values over $\alpha = 2$ were also investigated but could not improve the results any further (although they did not harm, either).

Table 6.7 — Results for different parameter settings on the TREC QC dataset. Evaluation of different kernels on the question classification dataset for different values of α, different values of λ and different semantic smoothing kernels. Evaluation results for the VSM representation are included as baseline (obviously, the settings of λ don't apply here). For each combination, the table reports micro-averaged F_1 measure (μF_1) and adjusted single-label accuracy (A^*). All numbers are percentages.

		$\lambda = 0.05$		$\lambda = 0.01$		$\lambda = 0.005$		$\lambda = 0.001$	
		μF_1	A^*	μF_1	A^*	μF_1	A^*	μF_1	A^*
	vsm	81.5	89.2	81.5	89.2	81.5	89.2	81.5	89.2
$\alpha = 1$	*vsm$_n$ + tree kernel*	89.8	90.8	89.7	91.2	89.7	91.2	89.7	91.4
$\alpha = 1$	*vsm$_n$ + sstk hyp-full-norm*	91.0	92.6	90.6	92.6	90.6	92.6	90.7	92.6
$\alpha = 1$	*vsm$_n$ + sstk hyp-resnik-norm*	91.0	92.6	90.9	92.2	90.9	92.2	90.9	92.2
$\alpha = 1$	*vsm$_n$ + sstk hyp-lin-norm*	90.9	92.2	91.2	92.4	91.2	92.4	91.3	92.4
$\alpha = 1$	*vsm$_n$ + sstk hyp-wupalmer-norm*	91.0	92.6	91.2	92.6	91.2	92.6	91.3	92.6
$\alpha = 2$	*vsm$_n$ + tree kernel*	89.7	91.4	89.8	91.6	89.9	91.4	90.0	91.4
$\alpha = 2$	*vsm$_n$ + sstk hyp-full-norm*	91.2	93.2	91.5	93.0	91.6	93.0	91.6	93.0
$\alpha = 2$	*vsm$_n$ + sstk hyp-resnik-norm*	91.1	92.8	91.3	92.6	91.2	92.6	91.2	92.6
$\alpha = 2$	*vsm$_n$ + sstk hyp-lin-norm*	91.1	92.4	91.4	92.6	91.2	92.6	91.5	92.6
$\alpha = 2$	*vsm$_n$ + sstk hyp-wupalmer-norm*	91.6	93.6	91.5	93.0	91.4	92.8	91.4	92.8

Table 6.8 — Summary of literature results for the TREC QC dataset in terms of adjusted single-label accuracy (A^*). All numbers are percentages.

Source	Algorithm & Features/Kernel	A^*
Li and Roth (2006)	Sparse Network of Winnow on various manually constructed features	92.5
Quarteroni et al. (2007)	SVM on VSM	90.6
	SVM on tree kernel and VSM (individually normalized)	91.8
Zhang and Lee (2003)	SVM on VSM	85.8
	SVM on n-grams	87.4
	SVM on (modified) tree kernel	90.0

Comparison with Literature Results

As the evaluation setting of Li and Roth (2002) which was considered in this experiment has been used several times in literature, we shortly compare the results of this section with those reported in literature. Table 6.8 summarizes the best figures reported as well as a selection of figures on inferior representations.

First of all, note that the baseline results obtained using SVMs on the VSM representation vary between 85.8% (Zhang and Lee, 2003) (which is substantially worse than our baseline results, i.e. 89.2%) and 90.6% (Quarteroni et al., 2007) (which is slightly above our results). These differences could be due to a different preprocessing strategy but also due to effects relating to the setting of the C parameters or the cost-factor. In either case, the VSM representation should not serve as the relevant baseline as it ignores syntactic information.

More important are the results obtained using tree kernels. Zhang and Lee (2003) used a modified formulation of the original tree kernels by Collins and Duffy (2002) which directly integrates the VSM feature space into the tree kernel computation yielding an overall accuracy of 90.0%. This result is already outperformed by our basic configuration *vsm$_n$ + tree kernel* in the range of 90.8% – 91.6%. Similar to the results reported here, Quarteroni et al. (2007) report their best result for this setting at 91.8% multi classification accuracy.

Li and Roth (2006) report on the use of a different classifier infrastructure, namely Sparse Network of Winnow (Carlson et al., 1999) using a whole range of complex and manually constructed syntactic and semantic features. Despite this elaborate feature space design, the best multi classification accuracy reported there is 92.5% as compared to 93.6% in the experiments of this section. The superior behaviour of all the taxonomy-derived SSTKs thus suggests that they can accurately model complex feature spaces for QC tasks while requiring minimal user intervention.

6.5 Summary and Discussion

This chapter has reported on an extensive series of experiments based on the concepts of Semantic Smoothing Kernels and Semantic Syntactic Tree Kernels which were introduced in Chapter 5. In the following, we summarize the main findings:

- Semantic Smoothing Kernels are an effective means for incorporating conceptual background knowledge from lexical knowledge structures into the prevailing VSM representation to introduce a bias in situations where there is insufficient training data to build stable statistical models. Generally, the performance of SSKs is superior to the performance of the plain VSM model in cases where little training data is available. In situations of "sufficient" training data, SSKs are generally less effective and can even deteriorate the results when compared to the VSM baseline.

- The behaviour of SSKs varies with respect to the characteristics of the target class. If a given target classes can be characterized unambiguously by means of a set of terms which occur frequently, SSKs appear to be less effective while they show their potential in situations of heterogeneous use of language.

- Different SSKs configurations based on (weighted) taxonomic overlap differ with respect to the performance that can be achieved. While "crisp" weighting schemes like *hyp-full* and *hyp-resnik* can show good results for very small training sets, they soon fail to do so as more training data becomes available. Combined weighting schemes like *hyp-wupalmer* and *hyp-lin* can carry over good initial results to larger training subsets. Finally, "fine-grained" weighting schemes like *hyp-path1* and *hyp-path2* improve upon the baseline in virtually all cases, including situations of larger amounts of training data but can at the same time not achieve the high performance gains of the other schemes on very small training sets.

- One of the alternatives to using (weighted) taxonomic overlap for configuring the smoothing parameters of SSKs is to encode similarities according to a given similarity measure directly into a matrix and square this matrix for use as smoothing parameter. The experiments on the AMAZON corpus show that this approach can be effective as well when compared to the VSM baseline for smaller training subsets. However, this alternative configuration generally performs worse than the configurations based on (weighted) taxonomic overlap and sooner falls below the VSM baseline as more training data becomes available.

- Semantic Syntactic Tree Kernels are an effective way of incorporating background knowledge about lexical semantics with tree kernels for syntactic structure. This is a desirable property on tasks like question classification where syntactic structure is important but, at the same time, due to the short length of questions, data sparseness is also an issue. The experimental results show that SSTKs based on (weighted) taxonomic overlap do not only outperform the VSM baseline and ordinary tree kernel formulations but also benchmark results from literature that employ complex manually constructed syntactic and semantic features.

Open issues along the lines of work of this chapter include the following:

- An improved handling of word sense ambiguity. In all experiments in this chapter, we have used the default WSD strategy of considering the most frequent noun sense for a given term. In this light, the results reported in this chapter appear to be pessimistic estimates of a perfectly disambiguated sense-based representation. Mavroeidis et al. (2005b) have experimented with SSKs on disambiguated term vectors, but it is not clear in how far their (mixed) results can be attributed to their WSD scheme. In fact, authors like Sanderson (1994) have claimed that word sense ambiguity has only a minor effect on performance in IR and various text mining tasks and our results seem to support this finding. Furthermore, the impact of wrongly disambiguated terms is likely to vary substantially between the different experiments. For example, the compound indexing scheme used in the AMAZON experiments is likely to cause a substantial reduction in wrong disambiguation of relevant words. Similarly, for the experiments with Semantic Syntactic Tree Kernels, terms are constrained within their respective POS classes, thus minimizing the wrong assignment of noun senses to verb terms.

- The design of the smoothing parameters in the experiments was mostly restricted to measures of (differently weighted) taxonomic overlap and normalized variants thereof. While the analysis in Chapter 4 has shown that (most) taxonomic similarity measures are unsuited for this purpose because they are likely to yield indefinite similarity matrices, future work also needs to investigate the design of new similarity measures which comply with the requirement of positive semi-definiteness.

- We have seen in Chapter 5 that (unnormalized) smoothing parameters also introduce, in addition to the semantic smoothing, a reweighting effect on the basic term representation. Future work thus needs to investigate the interplay between weighting schemes like TFIDF and this effect.

- Future work also includes, at least for the case of Semantic Smoothing Kernels, measures which are based on *relatedness* measures as opposed to *similarity* measures considered here. Practically, these measures use not only the taxonomic backbone of the employed knowledge structure, but also take other, non-taxonomic, relations into account (Budanitsky and Hirst, 2006).
- Another direction of research is the use of automatically induced knowledge structures as taxonomic backbone for the computation of semantic similarity measures. Initial results in the context of explicit conceptual document representations, e.g. by Bloehdorn et al. (2006c) suggest that such structures can to some extent mimic the effect of manually constructed knowledge structures.

Part IV

Kernels for Mining Instance Data in Ontologies

Chapter 7

Designing Kernel Functions for Instance Data in Ontologies

The overall topic of this thesis is the design of kernel functions that are capable of working with declarative knowledge encoded in formal knowledge structures. The last chapter has covered one particular aspect of this setting, namely the question in how far knowledge structures can provide useful complementary knowledge for standard data representations — in our case with the specific focus on textual data. In this chapter we investigate another aspect of this setting, namely the case of learning with instances that are fully described within formal ontological structures. In particular, we analyze the primitives available within ontologies for modelling individuals and derive a set of corresponding kernel functions that can be tuned and combined for a particular learning setting. The chapter is organized as follows. Section 7.1 reviews the application context and motivates the work of this chapter together with an analysis of some example scenarios and the main requirements. Subsequently, Section 7.2 formally introduces the kernel design framework. It first motivates and introduces the individual kernel components and then discusses their composite structure. Where appropriate, the section provides examples of such kernels and discusses practical issues for kernel computations. While the kernel design framework is presented in terms of the generic class-based ontology model of Chapter 3, it is meant to be practically applied in the context of Web Ontology Language (OWL)-based ontologies. Section 7.3 shortly discusses implementation strategies for the kernel framework. Section 7.4 points to related work in the area of kernel functions and similarity measures for ontologies while Section 7.5 concludes with a short summary and a discussion of the presented concepts. We instantiate the kernel framework introduced in this chapter in a set of practical experiments in the subsequent chapter.

7.1 Learning with Formal Knowledge Structures

In this section, we review the application context and motivate the work of this chapter.

7.1.1 Motivation

In Chapter 3, we have introduced the concepts of knowledge structures and ontologies as important building blocks of the emerging Semantic Web (Berners-Lee et al., 2001; Shadbolt et al., 2006). Recall that the Semantic Web is the vision of the World Wide Web as a universal medium for data, information, and knowledge exchange where knowledge can be expressed formally in a format that can be interpreted directly by computer systems, permitting them to discover and integrate knowledge and to reason about it. Hereby, ontologies provide the vocabulary to formally describe and integrate data. The standardization of the Resource Description Framework (RDF) and the Web Ontology Language (OWL) (Horrocks et al., 2003) have led to an increasing amount of available reference ontologies and a rising number of semantic annotations. As of March 2008, the statistics of the Semantic Web search engine SWOOGLE[1] count a total of $1,359,231$ publicly available *"error-free pure Semantic Web documents"* containing a number of $606,394,051$ *"triples [that] could be parsed from all Semantic Web documents"*.

With the increased availability of data sources of this type, engineers of intelligent applications are facing the question how these data sources can be used in data mining scenarios. While the prevailing paradigm for working with formal ontologies is deductive reasoning, this view takes the paradigm of working with formal ontologies by means of inductive reasoning. The particular appeal of this approach lies in the fact that it enables the linking of the formal logical underpinning of current Semantic Web technology with machine learning techniques. This approach be seen characterized as learning *from* the Semantic Web — a complementary activity to work in ontology learning, i.e. learning *for* the Semantic Web (Buitelaar et al., 2005, and references therein). Despite early interest in the topic, also phrased *Semantic Web Mining* (Berendt et al., 2002) at that time, principled approaches for mining Semantic Web data by means of statistical machine learning methods are still missing.

Example 7.1 (Bibliographical Data)**.** Consider the case of a knowledge structure depicted in Figure 7.1 which describes the domain of research, including persons, institutions and publications. Assume a user, who has read some of the publications stored in the knowledge structure, some of which were judged positive, others negative. How could a statistical learning algorithm use the overall knowledge structure to learn a predictive model for publications liked or disliked by the author and thus help him to retrieve relevant publications faster?

The kernel framework proposed in this chapter thus serves two main purposes. On the one hand, it provides a general and unifying framework for mining Semantic Web data sources with kernel methods. Just as the Semantic Web aims to abstract

[1] `http://swoogle.umbc.edu/`

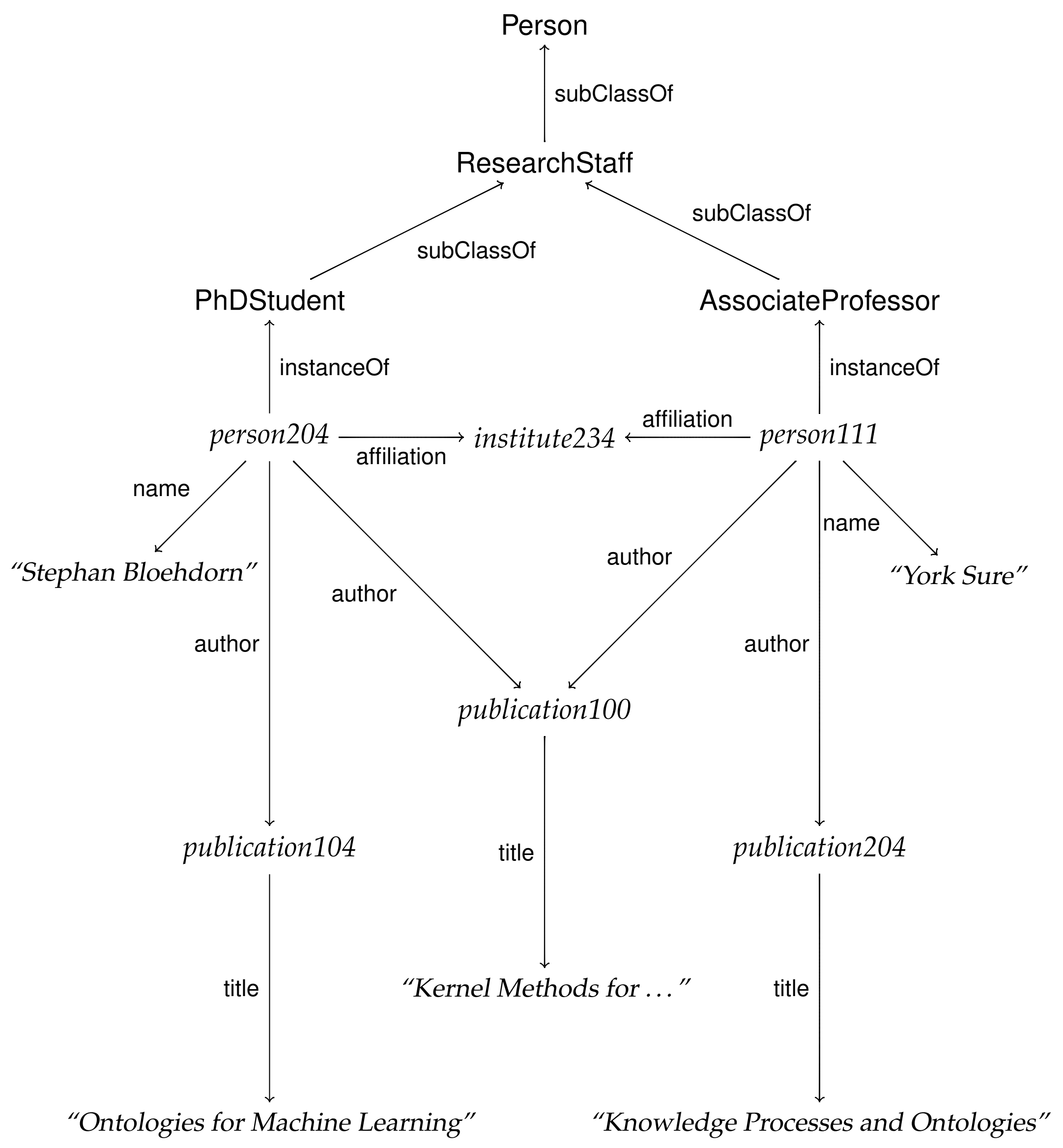


Figure 7.1 — Example ontology fragment. The fragment depicts a simplified fragment of the AIFB dataset, described according to the primitives of the SWRC ontology (Sure et al., 2005).

away from specific data formats and provide a general framework for data items, the family of kernel functions in this chapter aims to abstract away from special purpose kernel functions which are designed for a specific data structure. On the other hand, it provides a clean and flexible interface for integrating *intensional knowledge* into the learning process. Due to the formal semantics of these languages, kernel functions on Semantic Web data items can rely on deductive reasoning engines when comparing characteristics of two instances. As most data sources exhibit complex relationships, implicit domain knowledge specified within ontologies can provide valuable extra information.

7.1.2 Statistical Learning vs. Inductive Logic Programming

Learning with formal knowledge structures has traditionally been the domain of Inductive Logic Programming (ILP). The term was coined by Muggleton (1991), to describe the research area at the intersection of machine learning and logic programming (Lavrač and Džeroski, 1994; Muggleton and Raedt, 1994). Similar to the techniques discussed in this chapter, ILP uses logic as the uniform representation for examples and background knowledge. However, in ILP also the learned hypotheses are described in terms of formal logic. Specifically, given a first-order logic encoding of the known background knowledge and a dataset represented as a set of logical axioms, ILP systems try to derive a logic program which explains all the positive but none of the negative examples. Successful applications areas for ILP systems include the learning of structure-activity rules for drug design, prediction of protein structure and fault diagnosis rules for technical systems. The strongest point of ILP techniques is, however, that they provide an intensional description of the positive training examples in a classification task in terms of the logical language. The results can thus be interpreted naturally by knowledge engineers and incorporated directly into the ontological framework.

On the other hand, ILP techniques exhibit several problems when contrasted with the kernel-based paradigm taken in this thesis. First, the extreme computational requirements for searching optimal hypotheses have shown to limit the applicability of ILP algorithms in many practical settings. Secondly, ILP is occupied almost entirely with classification tasks, whereas the portfolio of kernel-based learning techniques naturally embraces all major types of learning problems, including besides classification tasks as regression, ranking, novelty detection, clustering, and dimension reduction. Thirdly, kernel-based learning techniques can be thoroughly analyzed in terms of statistical learning theory. Like all statistical learning algorithms, this allows for a principled handling of noise within the training data while the complexity of the resulting models can be controlled in terms of the norm in the implicit feature space, thus leading to better generalization behaviour.

The main ingredient for successful kernel-based learning is naturally the choice of an adequate kernel function. This can be seen as an alternative to propositionalization techniques (Kramer et al., 2001) which aim at the construction of propositional features for logic-based representations.

7.1.3 Requirements

In the following, we collect a number of requirements for kernel functions on instances in ontologies.

First of all, we have seen in Chapter 2 that kernel functions are characterized as the set of positive semi-definite functions. As a natural formal requirement a kernel framework for instance data in ontologies needs to ensure the validity (i.e. the positive semi-definiteness) of the resulting kernel functions under all parameter choices.

Secondly, the specific description of a given set of data items depends on the modelling decisions taken by the engineer(s) of the reference ontologies. Along the same line, the quality of a given kernel function will depend on the learning task at hand and on the available ontological structure that can be exploited. The kernel framework should thus not prescribe a specific kernel function but rather provide the user with the flexibility to design kernel functions with respect to the given task and data.

Thirdly, kernel functions should not rely only on the explicit assertions within the ontology but also on the intensional knowledge contained in the reference ontologies. The kernel framework should thus generally be based on all of the inferred knowledge.

7.2 Kernel Design Framework

In this section, we introduce a general framework for kernel functions on instance data as a formal and comprehensive foundation for mining ontological instance data. The basic principle of the family of kernels covered by this framework is the decomposition of instances into sets of kernel functions, each of which exploits another descriptive primitive. Along the common lines of interpretation of similarity we will consider two instances the more similar, the more common (or similar) characteristics they have. We look at the modelling primitives available in class-based ontology languages and discuss the respective implications for the similarity of instances. Based on these considerations, we suggest kernel functions that mirror the respective notion of similarity. For each component, we give the formal definition and discuss its interpretation. Depending on the learning task at hand and on the nature of the available data, this set of component kernels has to be specified declaratively by the kernel engineer.

Throughout this section and the rest of this chapter, we rely on the ontology model introduced in Chapter 3. While this basic model constitutes the minimal set of modelling primitives, we will generally assume that ontologies will be more richly structured in general, in particular that they use other primitives of OWL and the associated Description Logics (DLs). Throughout this section, we will assume that kernels are always computed with respect to a fixed reference ontology $\mathcal{O} = (\mathcal{I}, \mathcal{C}, \mathcal{P}_0, \mathcal{P}_D, \mathcal{D}, \mathcal{S})$. Of course in data integration scenarios, $\mathcal{O}$ can be any combination of n individual ontologies, i.e. $\mathcal{O} = \mathcal{O}_1 \cup \ldots \cup \mathcal{O}_n$. Hereby $\mathcal{O}_1 \cup \mathcal{O}_2$ is the *global interpretation* of the two ontologies $\mathcal{O}_1$, $\mathcal{O}_2$ which is given by the union of the entities and axioms of the respective ontologies.[2]

7.2.1 Atomic Kernel Functions

This class of kernels consists of those kernel functions which do not rely on any other individuals for computing the kernel values than the two argument individuals.

Identity Kernel

The identity layer kernel can be seen as the most simple kernel function.

Definition 7.1 (Identity Kernel for Individuals). Given two instances $ind_1, ind_2 \in \mathcal{I}$ of some reference ontology $\mathcal{O}$, we define the identity kernel $\kappa_\equiv : \mathcal{I} \times \mathcal{I} \mapsto \{0,1\}$ as:

$$\kappa_\equiv(ind_1, ind_2) = \begin{cases} 1, & \text{if } \mathcal{O} \models ind_1 \equiv ind_2 \\ 0, & \text{otherwise} \end{cases} .$$

As we will see, the purpose of this kernel is mainly to serve as an argument to other (non-atomic) kernel functions to be considered subsequently. Note that the kernel does not compare instances based on their instance name but in terms of their *asserted equality*. Recall from the discussion in Chapter 3 that in the common setting of DLs, where the unique names assumption does not hold, this means that the two different instance names can resolve to one and the same object. The kernel thus corresponds to an implicit feature space in which each instance is mapped to a distinct dimension that corresponds to its counterpart in the model domain.[3]

[2]The global interpretation of multiple ontologies is usually the desired setting. If for specific reasons a *local interpretation* is desired, the framework could be extended accordingly, e.g. in terms of the C-OWL formalism proposed by Bouquet et al. (2003).

[3]Such mappings enhance the independence of inputs and force learning algorithms to rely on more data points when building a statistical model. Refer to Shawe-Taylor and Cristianini (2004) for a discussion of the relation of this to 1-norm and 2-norm formulations of the soft margin Support Vector Machine (SVM).

Class-Based Kernels

The core modelling primitive for organizing instances in ontologies is the use of *classes*. Named classes organize instances along basic properties and are related amongst each other by subsumption relationships. Further, for expressive ontology languages based on DLs, specifically OWL, complex classes can be related to complex descriptions, whereby a description can make use of various constructors and constraints (c.f. Chapter 3). In richly axiomatized ontologies, most of the interesting logic deductions relate to the automated structuring of complex classes on the subsumption hierarchy.

Classes correspond to basic properties and abstractions. The different class(es) that are instantiated by the argument instances thus form the basic building block for their comparison within the kernel framework. Intuitively, this type of similarity is useful in those cases where there is some variation in the classes that are instantiated.

Definition 7.2 (Common Class Kernel)**.** Given two instances $ind_1, ind_2 \in \mathcal{I}$ of some reference ontology $\mathcal{O}$, and a set $\hat{\mathcal{C}} \subseteq \mathcal{C}$ of *reference classes* the *common class kernel* $\kappa_{class} : \mathcal{I} \times \mathcal{I} \mapsto \mathbb{R}$ is given by:

$$\kappa_{class}^{\hat{\mathcal{C}}}(ind_1, ind_2) = \sum_{\mathsf{C} \in \hat{\mathcal{C}}} [\mathcal{O} \models \mathsf{C}(ind_1)][\mathcal{O} \models \mathsf{C}(ind_2)]$$

whereby $[\cdot]$ again denotes an indicator function which evaluates to '1' and '0' if the bracketed expression is true or false, respectively.

The validity of the kernel is readily verified as it can be interpreted by defining the mapping $\phi(\cdot)$ as a mapping into a vector space whose dimensions correspond to the classes $\hat{\mathcal{C}}$ and the vector components correspond to the binary indicators.

Note that the scope of the kernel is defined by the selection of the classes in $\hat{\mathcal{C}}$. In particular, this also comprises the case of a single class. For the case of expressive ontology languages in the spirit of DLs, this can include complex class descriptions such as $\exists \mathsf{prop}_1.\mathsf{class}_1$.

Example 7.2 (Academic Domain (cont.))**.** Recall the situation of Figure 7.1. The common class kernel can compare the individuals *person204* and *person111* based on all classes. As both indicator functions on the right hand side evaluate to '1' for the Person and ResearchStaff classes, the kernel would yield $\kappa_{class}(person204, person111) = 2$.

An alternative paradigm for computing class-based kernel functions is to associated an individual with its *most specific concept (MSC)* and employ a kernel function on the respective classes.

Definition 7.3 (Most Specific Class)**.** Given an instance *ind* $\in \mathcal{I}$ of some reference ontology $\mathcal{O}$, the *most specific class* of *ind* is the class $\mathrm{msc}(ind) = \mathsf{C}$ such that $\mathcal{O} \models \mathsf{C}(ind)$ and $\mathsf{C} \sqsubseteq \mathsf{D}\ \forall \mathsf{D}$ with $\mathcal{O} \models \mathsf{D}(ind)$.

Note however, that for many Description Logics there need not be a single *atomic* class which fulfills the condition of a MSC — in this case, the MSC needs to be formed by the conjunction over several classes (Baader et al., 2003). This setting presupposes the definition of a valid kernel function on class descriptions $\kappa : \mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}$. Two individuals are then assessed via this kernel function on their respective most specific classes, i.e. $\kappa(ind_1, ind_2) = \kappa(\mathrm{msc}(ind_1), \mathrm{msc}(ind_2))$. Such kernel functions could, for example be taxonomic similarity measures as discussed in Chapter 4 as long as positive semi-definiteness can be ensured. This approach has also been considered by Fanizzi and d'Amato (2006) in conjunction with special kernel functions on complex class descriptions.

Data Property Kernel Component

As a next building block for determining the similarity of two individuals we consider the properties the argument individuals participate in. The general structure of this kernel is a sum of kernel evaluations of all compatible pairings of properties.

Definition 7.4 (Data Property Kernel)**.** Given two instances $ind_1, ind_2 \in \mathcal{I}$ of some reference ontology $\mathcal{O}$, and a reference data property $\mathsf{dprop} \in \mathcal{P}_D$ with associated kernel $\kappa(\cdot,\cdot)$, the *data property kernel* $\kappa_{data} : \mathcal{I} \times \mathcal{I} \mapsto \mathbb{R}$ is defined as:

$$\kappa_{data}^{\mathsf{dprop}}(ind_1, ind_2) = \sum_{\{d_i \,|\, \mathcal{O} \models \mathsf{dprop}(ind_1, d_i)\}} \; \sum_{\{d_j \,|\, \mathcal{O} \models \mathsf{dprop}(ind_2, d_j)\}} \kappa(d_i, d_j)\,.$$

Hereby, $\kappa(d_i, d_j)$ is any valid kernel function defined on the data values of $\mathcal{O}$.

Again, the kernel property can be easily verified as the kernel is an instantiation of the crossproduct kernel of Definition 4.7. The kernel makes use of an underlying base kernel that is defined on the respective datatype linked to by the respective data property. For basic datatypes, such as strings or numeric values, a variety of useful kernels have been defined. For String datatypes, we may for example consider dot products of their Vector Space Model (VSM) representation as discussed in Chapter 5 or the String Kernels as introduced by Lodhi et al. (2002). For numerical values, the Gaussian Kernel defined on real numbers is a common choice. Again, we refer to Shawe-Taylor and Cristianini (2004) for information on such kernels. The feature space associated with this kernel is the feature space associated with the respective base kernel, whereby an individual is represented implicitly as the sum of the representation vectors of the data values linked to by the data property under consideration.

Example 7.3 (Academic Domain (cont.))**.** Recall the situation of Figure 7.1. The data property kernel can compare the individuals *publication104* and *publication204* based on the targets of their respective title properties. Depending on the employed base kernel, the occurrence of the word *"ontologies"* in both titles would lead to judging the instances as related to some extent.

7.2.2 Kernel Functions on Relational Embeddings

In a similar spirit as with the datatype property kernel, we define kernel functions on object properties as follows.

Definition 7.5 (Object Property Kernel)**.** Given two instances $ind_1, ind_2 \in \mathcal{I}$ of some reference ontology $\mathcal{O}$, and a reference object property $\mathsf{oprop} \in \mathcal{P}_0$, with associated kernel $\kappa(\cdot,\cdot)$, the *object property kernel* $\kappa_{object} : \mathcal{I} \times \mathcal{I} \mapsto \mathbb{R}$ is defined as:

$$\kappa_{object}^{\mathsf{oprop}}(ind_1, ind_2) = \sum_{\{ind_i \,|\, \mathcal{O} \models \mathsf{oprop}(ind_1, ind_i)\}} \; \sum_{\{ind_j \,|\, \mathcal{O} \models \mathsf{oprop}(ind_2, ind_j)\}} \kappa(ind_i, ind_j)\,.$$

Hereby, $\kappa(ind_i, ind_j)$ is any valid kernel function defined on the individuals of $\mathcal{O}$.

The definition in terms of the crossproduct kernel again makes reference to an underlying base kernel $\kappa(\cdot,\cdot)$, this time defined on *instances* just as the overall kernel itself. For example, the base kernel could be the matching kernel in which case the kernel feature space would boil down to counting the number of common targets for oprop.

If another kernel is used as base kernel, the respective feature space also becomes the feature space of the object property kernel, whereby the argument individuals are represented as the sum of the vector representations of their respective target individuals.

Recall from Remark 3.3 that in the common setting of Description Logics, the unique names assumption does not hold. In such cases, instance retrieval of the type $\{ind_j \,|\, \mathcal{O} \models \mathsf{oprop}(ind, ind_j)\}$ may return multiple references to one and the same object in the interpretation domain. If it is undesired that multiple references enter the summation, the set of retrieved instances needs to be explicitly constrained to *distinct* individuals.

Further note that the formulation of the kernel only considers target individuals. A corresponding kernel can be formulated for comparing instances based on the source individuals which link to the argument individuals. In the practical settings of OWL based ontologies, this behaviour can be simulated by introducing the inverse property and relating it to the original property by means of the inverse statement.

Example 7.4 (Academic Domain (cont.))**.** Recall the situation of Figure 7.1. The object property kernel can compare the individuals *person204* and *person111* based on the targets of their respective publication properties. If the identity kernel is used as base kernel, the result will be based on the joint publication *publication100*. If the data property kernel of the previous example is used as base kernel, the kernel would yield a more differentiated result based on the similarity between all pairs of publication titles of the two authors.

7.2.3 Kernel Aggregation and Modification

We have so far presented isolated building blocks for kernel calculations on individuals, e.g. kernels comparing the class structure or the property extensions of the instances. The power of kernel functions however also stems to a large extent from the capability of combining and modifying elementary kernel functions to yield more expressive feature spaces.

Given base kernel functions $\kappa_1(\cdot,\cdot)$ and $\kappa_2(\cdot,\cdot)$, they can be *combined* using the combination operators discussed in Section 2.4.2, namely multiplication and addition.

- The interpretation of the feature space associated with the *additive* combination of several kernels is that it corresponds to the composite feature space that is made up of all dimensions of the feature spaces of the contributing kernels, i.e. by means of the concatenation of the (implicit) argument vectors.
- The interpretation of the feature space associated with the *multiplicative* combination of two kernels is that it corresponds to the tensor product of the two implicit feature vectors, i.e. with the feature vector where each components corresponds to the product of two independent features of the original feature vectors.

Similarly, basic kernel functions can be *modified*, by using them as argument to known kernel modifiers such as the *cosine normalization modifier*, the *polynomial kernel modifier* or the *Gaussian kernel modifier* with the corresponding interpretations.

Note that, as discussed in Chapter 2, all of these operations retain the kernel property. As the component kernel functions as well as their combination or modification operators are valid, the validity of the overall set of kernel functions that can be constructed within this framework is ensured at all times.

7.3 Implementation Strategies

From the practical perspective of the implementation, it is desirable to optimize the efficiency of the computation of kernel functions. In the following, we discuss the strategies to optimize the computations within the kernel framework.

Caching

First, each component kernel maintains a cache on pairs of argument instances. At the top level kernel, all pairs of argument individuals are usually evaluated only once. Substantial repetitions can occur however for kernel functions that act as base kernels for object property kernels. The effect of caching is more favourable, the more interlinked the target instances are.

Optimizing Class-Based Kernel Functions

Secondly, the computation of class-based kernel functions can be optimized by iterating over the individual classes in $\hat{C}$ by traversing them along their hierarchical order imposed by the subsumption hierarchy $\mathcal{O}$ in a depth-first manner. The optimized strategy then returns from a given path in this hierarchy as soon as one of the argument individuals fails to instantiate the given class. This procedure avoids the unnecessary exploration of classes which, due to their specificity with respect to the class for which the common instantiation has failed first, would not yield any further positive contribution to the kernel anyway.

Optimizing References in Aggregate Kernel Functions

Thirdly, the kernel function infrastructure proposed so far often relies on the embedding of base kernels within other, higher-level kernels. This nesting of kernel functions yields a tree-like structure. In this structure, it is possible to merge those subtrees that correspond to identical kernel chains, resulting in a kernel reference graph that conforms to a directed acyclic graph (DAG). As a result, identical kernels or chains of identical kernels that occur in different nesting contexts are evaluated only once.

Consider Figure 7.2 which illustrates the concept in the context of our earlier example from the academic domain. The aggregated kernel aims at comparing individuals of the person class based on their affiliations, co-authors and the affiliations of the co-authors. The top level kernel additively combines two object property kernels based on the affiliation and author object properties. While the former kernel directly references the identity kernel $\kappa_{\equiv}$ (i.e. it directly compares the affiliations), the latter again references another chain of kernels. In particular, the publication individuals linked to by the author object property are compared by an aggregated kernel that additively combines the identity kernel $\kappa_{\equiv}$ and an object property kernel which again references the affiliation object property with, again, an embedded identity kernel.

Through structuring the mutual references between the component kernels in a DAG structure, the the identity kernel need to be evaluated only once as well as the object property kernel based on the affiliation property. For any subsequent references,

$$\begin{tikzcd}
 & \kappa_{addition} & & \\
\kappa_{scale}\ (0.3) \arrow[ur] & & \kappa_{scale}\ (0.7) \arrow[ul] & \\
 & & \kappa^{\text{author}}_{object} \arrow[u] & \textit{(publications)} \\
 & & \kappa_{addition} \arrow[u] & \\
 & \kappa^{\text{author-of}}_{object} \arrow[ur] & & \textit{(co-authors)} \\
\kappa^{\text{affiliation}}_{object} \arrow[uuuu] \arrow[ur] & & \kappa_{\equiv} \arrow[ll] \arrow[uu] & \textit{(affiliations)}
\end{tikzcd}$$

Figure 7.2 — Example of a Kernel Computation Graph on the SWRC ontology.

the result can be retrieved from the kernel cache.

7.4 Related Work

The work in this chapter has covered the design of kernel functions for individuals described within formal ontologies. In this section, we shortly review related research directions, namely those that address the design of kernels or arbitrary similarity functions for individuals in ontologies or related forms of structured data.

Distance and Similarity Measures for Logic-Based Formalisms

A general framework for similarity measures in ontologies is proposed by Ehrig et al. (2005). The framework is based on distinguishing several layers of similarity, namely the data layer which takes into account syntactic similarities of the involved ontology primitives, the ontology layer which corresponds to measures that use the ontology structure and the context layer which takes into account how ontology entities are used in some external context. Bernstein et al. (2005) and Hefke et al. (2006) provide software frameworks for computing semantic similarity measures according to a wide range of similarity measures. Kiefer et al. (2007b) further provide an infrastructure for integrating the similarity assessment between entities into standard Simple Protocol and RDF Query Language (SPARQL) queries.

The main distinction of these similarity frameworks to the kernel framework presented in this section is that the proposed measures are used as heuristics for various tasks, but not for statistical learning. The most prominent application domain for this kind of similarity measures is the area of ontology alignment (Ehrig, 2007, and references therein) which targets the automated discovery of correspondences within different ontologies. Along another line, Kiefer et al. (2007a) apply their similarity framework for retrieval of semantic business processes descriptions. Other areas that commonly employ ontology based similarity measures are ontology-based information retrieval, or ontology-based case-based reasoning. All these tasks impose only minor formal requirements on the properties of the similarity functions and many of the proposed functions will not conform to valid kernel functions.

Early endeavours for the definition of comprehensive similarity measures within formal knowledge representation for use within *learning tasks* were made by Bisson (1995), however with no practical application in mind. Several variants of similarity and dissimilarity measures have been used in the context of *relational instance based learning*, i.e. Nearest Neighbour-type learning (c.f. Section 2.3.3) on relational knowledge structures (Emde and Wettschereck, 1996; Horváth et al., 2001). While these measures are (mostly) designed to fullfill the requirement of conforming to a proper mathematical distance, they will usually not lead to valid kernel functions. In a similar spirit as the work in this chapter, Maedche and Zacharias (2002) propose a set of ontology-based similarity measures for clustering ontology-based metadata descriptions. However, the restriction to the application of hierarchical agglomerative clustering poses no formal requirement on the form of the employed measures which consequently do not always exhibit kernel characteristics.

In summary, the work on arbitrary distance or similarity functions is widely scattered. However, the application context considered for these similarity functions does not require that the respective similarity functions are positive semi-definite which makes them not generally suited for application in kernel-based learning methods.

Convolution Kernels

Starting with the work of (Haussler, 1999), research on kernel functions for structured data, i.e. for data that is expressed in a paradigm different from the standard vectorial representation, has become a major topic of investigation in the machine learning community (Gärtner, 2003).

Much of the work on kernels for structured data is rooted in the idea of the *convo-*

lution kernel proposed by Haussler (1999). The idea of the kernel is to define for any kind of "structured" objects $x_1, x_2 \in \mathcal{X}$ with tuples $\vec{x_1}, \vec{x_2} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_D$ of "parts" of these objects a relation $R \subseteq (\mathcal{X}_1 \times \ldots \times \mathcal{X}_D) \times \mathcal{X}$ which encodes the decomposition of an object into its "parts" $R^{-1}(x) = \{\vec{x} : (\vec{x}, x) \in R\}$. In this framework, Haussler defined a kernel as a generalized convolution as follows:

$$k(x_1, x_2) = \sum_{\vec{x_1} \in R^{-1}(x_1)} \sum_{\vec{x_2} \in R^{-1}(x_2)} \prod_{d=1}^{D} k_d(x_1^d, x_2^d), \tag{7.1}$$

whereby the k_i correspond to arbitrary kernels defined on $\mathcal{X}_i$. The general structure of the kernel has been highly influential for work on kernel functions for structured data.

The universality of the convolution is also its problematic aspect as it leaves all details on the choice of R in practical settings unspecified. In fact, for specific choices of the relation R, most kernel functions on structured data, including the framework considered in this chapter can be seen as instantiations of the convolution kernel.

Kernel Functions for Logic-based Formalisms

As the first work in the direction of kernel functions for logic-based representations, Gärtner et al. (2003b) have proposed kernel functions on individuals represented as (closed) terms in the typed higher-order logic of Lloyd (2003). The terms of the logic are the terms of the typed λ-calculus while the basic type structure of this knowledge representation formalism consists of function types, product types, and type constructors. Neglecting most of the technicalities, this logic essentially allows the construction of complex types like sets, lists or trees out of other types (e.g. natural numbers). Function types can be used to represent types such as e.g. sets or multisets. Product types can be used to represent types corresponding to tuples of other types. Type constructors can be used to represent types corresponding to structured objects such as lists or trees. The work presents a kernel defined inductively on the terms in the associated logic. The basic term kernel is defined as product and sum of the kernel functions considered for every part of a term thus allowing a proof that the kernel is positive definite on all basic terms. This work constitutes the first principled approach that allows to exploit a logic-based knowledge representation language by means of kernel functions. However, the expressiveness of the proposed knowledge representation formalism is limited as it does deliberately not consider the link structure between individuals, although the authors suggest that the framework could be extended in this direction. The more problematic aspect is that the practical application of the framework for describing existing data items requires a complete translation of the data into the given formalism and is thus unsuited for direct application on Semantic Web data. As the aggregation of kernel functions on standard types is fixed based on the used type constructors, the approach also requires to think a-priori about the implications of the chosen representation on the kernel function, whereas the kernel framework proposed in this chapter allows to flexibly design kernels for the given representation primitives.

Another approach is taken by Frasconi et al. (2004). Within this framework, objects can be described using a restricted knowledge representation language consisting of a simple type hierarchy, attributes, as well as a fixed set mereotopological relations between objects. The proposed kernel, again guides the aggregation of basic kernel evaluations based on the employed topological relations. Similar to the previous case, users are thus required to commit themselves to the whole framework, and relating their modelling decisions to the structure of the kernel i.e. to adapt the data to the kernel, whereas the kernel framework of this chapter aims at adapting the kernel(s) to the given data.

In a very different spirit, Passerini et al. (2006) have proposed kernel functions on PROLOG proof trees. Like in the setting of this thesis, individuals are described as first-order logic objects in the context of global background knowledge. The idea of this kernel function is then to measure the similarity of two individuals by means of the similarity of the proof trees of a special logic program, called the *visitor program*, which probes certain characteristics of the individual that may be of interest for the domain. Despite this innovative approach, the interplay between the interesting characteristics of the data, the visitor program and the traces that can be observed in the proof tree will often be unclear. Despite interesting results, the authors themselves acknowledge that the proof trees are likely to contain unnecessary noise that hurts the overall performance.

On the schema-level, Fanizzi and d'Amato (2006) propose a declarative kernel for *concept descriptions* in the description logic $\mathcal{ALC}$. Structurally, the kernel is based on a representation of class descriptions in normal form. The kernel is then defined inductively: disjunctive descriptions are treated as taking the sum of the cross-similarities between any couple of disjuncts from the argument descriptions while conjunctive descriptions are treated as the product of the similarities between two input descriptions, distinguishing among primitive concepts, those referred in the value restrictions and those referred in the existential restrictions. The similarity between atomic classes is measured in terms of the intersection of their respective extensions.

The obvious difference of this approach to the kernel framework presented in this thesis is that it only allows to compute kernel functions on class descriptions but not on individuals. As a consequence, Fanizzi and d'Amato suggest that the kernels on class descriptions could also be used for describing individuals by means of representing an individual in terms of its most specific concept. In particular, this kernel is used in Fanizzi and d'Amato (2007) to simulate the instance retrieval problem. While this view is certainly interesting in cases of a highly differentiated network of complex class descriptions, it does not provide any means to assess the characteristics of a given individual in terms of its object and data properties as in our framework.

7.5 Summary and Discussion

Bringing together state-of-the-art methods from machine learning and knowledge representation is an important step towards extending the scope of current Semantic Web applications. This chapter has introduced a framework for the design of kernel functions on instance data in ontologies. The family of kernels introduced provides an interface for exploiting modelled domain knowledge within statistical learning algorithms in a principled way.

Open issues in this direction include methodologies for designing adequate kernels for a given learning problems within the proposed framework in a user-friendly way. Among others, this requires to think about intuitive interfaces for kernel design.

As for all learning methodologies, the ideal choice of representation depends on the setting investigated. Obviously, there is no kernel function that performs optimally for all datasets and all types of learning tasks. Instead, the specific instantiation of the proposed framework in a given learning setting will always remain a kernel engineering problem and the appropriateness of the resulting kernels need to be ultimately evaluated and compared in experimental evaluations. Similarly, the success also depends on which background knowledge is represented in the reference ontology. In the next chapter, we investigate the pratical application of the kernel framework for a set of selected learning problems.

Chapter 8

Experiments with Kernel Functions for Instance Data in Ontologies

In this chapter, we apply the framework for kernel function design on instance data in ontologies which was discussed in the previous chapter in practice. We look at different mining tasks on ontology-structured datasets, discuss the design of appropriate kernels and evaluate their performance experimentally. This chapter is structured as follows. We discuss the software implementation of the kernel design framework of the previous chapter in Section 8.1. Section 8.2 discusses a small experiment with the common class kernel in the context of an approximate reasoning task on the GALEN ontology. Section 8.3, reports on the application of the whole kernel framework on a classical Semantic Web dataset, the AIFB portal metadata in conjunction with the SWRC ontology. In Section 8.4, we report on the application of the framework on the CORA bibliographical dataset. We conclude with a short summary and a discussion of the findings in Section 8.5.

8.1 Implementation

As part of this thesis, the kernel framework described in the previous chapter has been implemented in the KAON2SIMILARITY system.[1] The implementation is provided as part of the ontology management and reasoning infrastructure KAON2.[2]

KAON2 is a complete infrastructure for managing OWL-DL and F-Logic ontologies, as well as the so-called *DL-safe* fragment of the Semantic Web Rule Languages (SWRLS). KAON2 provides, amongst others, a complete Application Programming Interface (API) for programmatic management of ontologies, means for processing ontologies in XML and RDF presentation syntax and an inference engine for answering conjunctive queries (expressed using SPARQL syntax). The infrastructure as well as the underlying reasoning algorithms were devised by Motik (2006).

KAON2SIMILARITY provides instantiations of all the kernel functions presented in

[1] `http://kaon2similarity.ontoware.org/`

[2] `http://kaon2.semanticweb.org/`

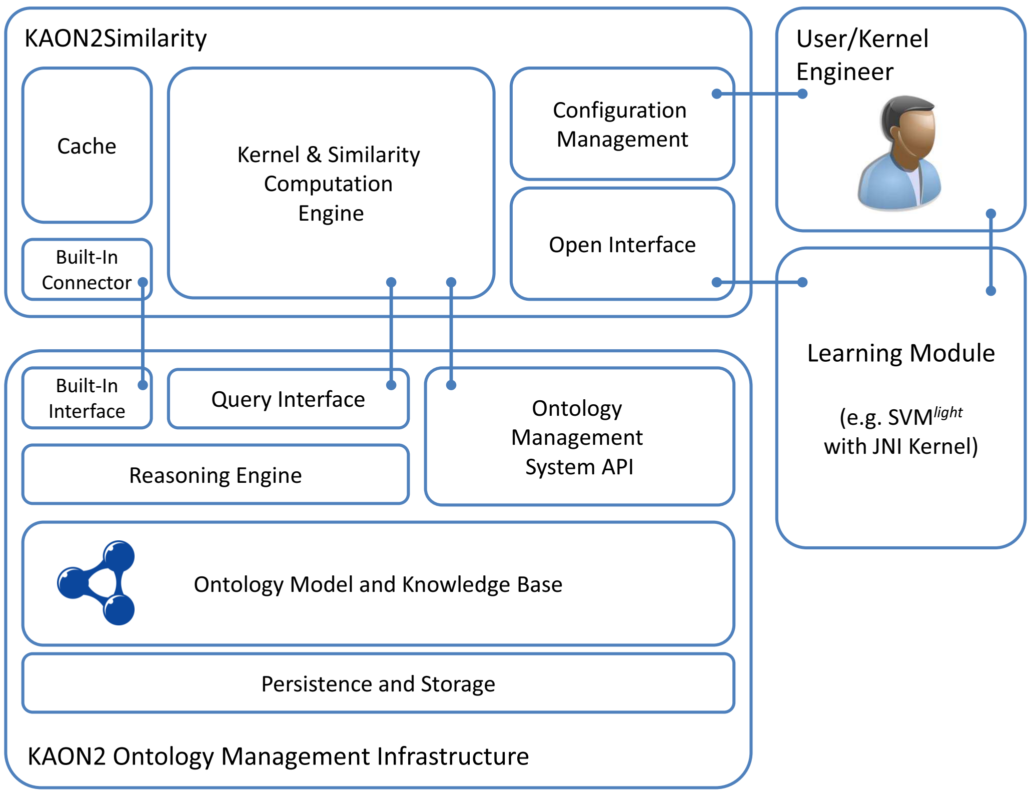


Figure 8.1 — Conceptual Architecture of KAON2SIMILARITY

the previous chapter, together with supporting infrastructure such as kernel modifiers, kernel aggregators and a caching module that can be flexibly plugged together for the purpose at hand. Figure 8.1 illustrates the main components of the system as well as the interaction with KAON2. In the following, we shortly discuss the main components.

The structure of the employed kernel functions can be specified declaratively by the user in terms of a simple XML syntax as exemplified in Figure 8.2. This configuration is processed by the configuration management component and the internal object representation of the kernel is set up. In the example of Figure 8.2, a kernel is specified as the sum of two base kernels, namely a common class kernel and an object property kernel based on the swrc : publication property whereby the matching kernel is implicitly defined as the identity kernel $\kappa_{\equiv}$. Alternatively, the kernels can also be constructed programmatically via the corresponding API of KAON2SIMILARITY.

A given configured kernel can be accessed programmatically via an open interface

```
<org.ontoware.kaon2similarity.aggregators.WeightedAddition>
  <baseSimilarities>
    <org.ontoware.kaon2similarity.individuals.CommonClassSimilarity>
       <inferencing>true</inferencing>
    </org.ontoware.kaon2similarity.individuals.CommonClassSimilarity>
    <org.ontoware.kaon2similarity.individuals.ObjectPropertyCrossProductSim>
       <inferencing>true</inferencing>
       <direction>FROM</direction>
       <oprop class="org.semanticweb.kaon2.ze">
         <b>http://swrc.ontoware.org/ontology#publication</b>
       </oprop>
    </org.ontoware.kaon2similarity.individuals.ObjectPropertyCrossProductSim>
  </baseSimilarities>
  <baseWeights>
    <double>1.0</double>
    <double>1.0</double>
  </baseWeights>
</org.ontoware.kaon2similarity.aggregators.WeightedAddition>
```

Figure 8.2 — Example of XML-based Kernel Specification in KAON2SIMILARITY

by external learning modules. The reference ontology is also specified during startup via this interface. For example, we have already mentioned the generic extension to the SVMLIGHT software by Joachims (1999) which provides an interface to kernel functions that are implemented in JAVA (c.f. Appendix A) which also allows to integrate kernel functions available via KAON2SIMILARITY into SVMLIGHT. This is also the setup adopted for the experiments in this chapter.

During computation of a given kernel, KAON2SIMILARITY interacts with the KAON2 modules, either by means of the ontology management system API of KAON2 or via the corresponding reasoning engine. The internal caching module avoids repeated evaluations of any kernel component.

As another way of interacting with KAON2, a given kernel configuration can also be registered with the KAON2 built-in interface. This interface allows to integrate extra-logical predicate evaluations with KAON2. While this mechanism is not related to the exposure of the kernel functions to external learning modules, it allows to integrate notions of similarity into interactions with KAON2, e.g. for use within SPARQL queries.

8.2 Common Class Kernel for the GALEN Ontology

In this section, we first illustrate the use of machine learning classification using only the common class kernel of Definition 7.2 for approximating reasoning tasks in a manner similar to approximate reasoning techniques (Hitzler and Vrandecic, 2005).

8.2.1 Task and Dataset

For this purpose, experiments were conducted on the OWL-DL version of the GALEN Upper Ontology.[3] The ontology contains atomic 175 classes, together with restrictions arranged in 193 SubClassOf axioms, 51 EquivalentClasses axioms, 127 DisjointClasses axioms and no nominals.

The task of this experiment was to imitate the classification behaviour of the ontology given a semantically weakened ontology. The basic idea of this setup was to investigate whether it is possible to accurately train a classification model that simulates the behaviour of a reasoning engine. As only the TBox of the ontology is available, the ontology was randomly populated with 1000 individuals. We used 7 subclasses of the Self_standing_entity class, namely the Biological_entity, Physical_entity, Complex, Continuant_entity, Discrete_entity, Mass_entity and Occurent_entity classes.[4] Ontology axioms were filtered and only the basic SubClassOf and ClassMember axioms of the basic ontology model of Chapter 3 were retained to obtain a weaker and semantically different ontology $\mathcal{O}_{weak}$. Table 8.1 summarizes the entailment statistics for the 7 classes for the original and for the changed version of the ontology.

8.2.2 Experimental Setup

The overall instance set was split in two groups of 500 instances for training and testing with class labels assigned according to the semantics of $\mathcal{O}_{original}$. SVMs with soft margin parameter $C = 1$ were trained using the common class kernel in conjunction with a cosine normalization modifier only using the information present in $\mathcal{O}_{weak}$. The trained models were then evaluated on the 500 test instances and $\mathcal{O}_{weak}$ and the results were compared with their actual classifications in $\mathcal{O}_{original}$.

8.2.3 Experimental Results

The results of the test runs are reported in Table 8.1. As the results show, the common class kernel is easily able to imitate the reasoning behaviour of the complex

[3]`http://www.cs.man.ac.uk/~rector/ontologies/simple-top-bio/`

[4]The Non_Biological_entity, Organelle and Non_Physical_entity were ignored as they are either complements of existing concepts or did not receive enough positive examples.

Table 8.1 — Statistics and Results of the GALEN Experiment — Number of overall positive Instances and classification results in terms of accuracy (A), precision (P), recall (R) and F_1.

GALEN Target Class	# in $\mathcal{O}_{original}$	# in $\mathcal{O}_{weak}$	A	P	R	F_1
Biological_entity	550	63	97.8	100.0	96.1	98.0
Physical_entity	587	11	97.8	99.3	97.0	98.1
Complex	116	88	99.2	100.0	93.1	96.4
Continuant_entity	600	96	98.2	100.0	97.1	98.5
Discrete_entity	433	78	99.2	100.0	98.2	99.1
Mass_entity	70	16	100.0	100.0	100.0	100.0
Occurrent_entity	95	0	98.0	100.0	75.0	85.7

ontology on a weaker, semantically different, ontology. These consistent positive results suggest that the weaker ontology contains sufficiently crisp patterns for certain Class (sub-) structures that are adequately represented within the common class kernel function. In fact, variations of the experiment revealed that given an alternative parameter setting or using a non-normalized kernel can achieve 100% accuracy for almost all classes. As the setting of this experiment is comparatively simple, it is mainly meant to illustrate the potential for learning logical patterns in a statistical manner.

8.3 Mining the SWRC Ontology

This section reports on experiments based on the SWRC ontology and the metadata available within the Semantic Portal of the Institute AIFB at the University of Karlsruhe. The dataset has been introduced in the context of an earlier publication (Bloehdorn and Sure, 2007) but has in the meantime also been used by Kiefer et al. (2008). The results reported in this section extend the results reported earlier (Bloehdorn and Sure, 2007).

8.3.1 Task and Dataset

The SWRC ontology initially grew out of the activities in the KA2 project and has since then been ported to various knowledge representation languages including OWL (Sure et al., 2005).[5] The ontology generically models key classes relevant for the research domain and the relations between them. The current version of the ontology comprises a total of 53 classes in a taxonomy and 42 object properties, 20 of

[5] `http://ontoware.org/projects/swrc/`

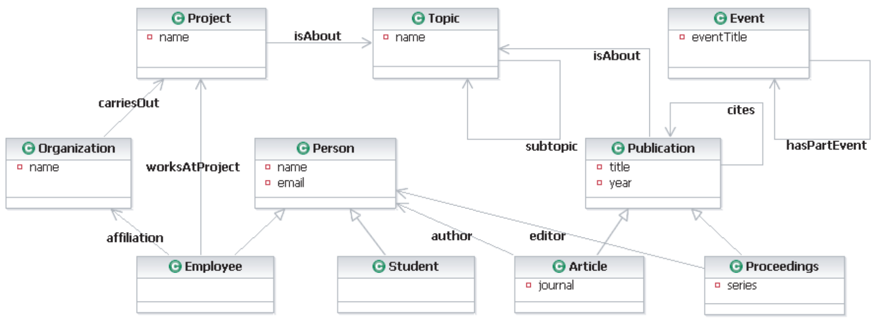


Figure 8.3 — Main classes and properties of the SWRC Ontology.

which are participating in 10 pairs of inverse object properties. All entities are enriched with additional annotation information. The SWRC ontology comprises at total of six top level concepts, namely the Person, Publication, Event, Organization, Topic and Project classes. Figure 8.3 shows a small portion of the SWRC ontology with its main top-level concepts and relations.

The SWRC ontology is used in a number of different settings. In particular, the vocabulary of the ontology is used for providing structured metadata for web portals of research institutions and research projects. These metadata descriptions include the Semantic Portal of the Institute AIFB at the University of Karlsruhe.[6]

The experiments in this section are based on the AIFB metadata from November 2006. The overall dataset comprises a total of $2,547$ instances. $1,058$ of these can be deduced to belong to the Person class, whereby 178 of these have an affiliation to one of the institute's groups (the others correspond to external co-authors). 78 persons thereof are research staff at the timepoint of the snapshot.[7] $1,232$ instances instantiate the Publication class, 146 instances instantiate the ResearchTopic class and 146 instances instantiate the Project class. The instances are connected by a total of $15,883$ object property axioms and participate in a total of $8,705$ datatype properties.

[6]`http://www.aifb.uni-karlsruhe.de/about.html`

[7]The entities corresponding to currently employed research staff are meant to be retrieved using the Employee class. However, the original version of the SWRC contained the OWL axioms ResearchTopic $\sqsubseteq \forall$isWorkedOnBy.AcademicStaff and AcademicStaff $\sqsubseteq$ Employee. The isWorkedOnBy is meant to link from research topics to persons. The specific modelling essentially makes a DL reasoner consider any person with associated research topic an employee, a situation that is not necessarily true. This modelling flaw is to be removed in future versions of the ontology. For the preparation of the datasets described below, the first of the above axioms was removed to identify proper employees only. For the actual experiments, however, the original ontology was used.

8.3.2 Experimental Setup

Given the information in the SWRC ontology, three different classification problems have been devised for experimental evaluation, namely:

person2affiliation (any with affiliation) In this setting, the 178 instances of type Person which have an affiliation with any of the institutes research groups form the instance set. The dataset considers as the four positive target classes the membership in each of the four research groups *"Business Information and Communication Systems"* (73), *"Efficient Algorithms"* (28), *"Knowledge Management"* (60) or *"Complexity Management"* (16).[8]

person2affiliation (employee) This setting corresponds to the setting considered in (Bloehdorn and Sure, 2007). It is equivalent to the setting described above but additionally requires that positive target instances in each of the research groups correspond to currently employed researchers, i.e. *"Business Information and Communication Systems"* (24), *"Efficient Algorithms"* (14), *"Knowledge Management"* (27) or *"Complexity Management"* (13). The remaining instances (i.e. former employees whose affiliation was considered in the previous setting) were considered as negative instances for each of the four target groups.

paper2affiliation In this setting, the 1,232 instances of type Publication form the instance set. The target classes are again the four research groups of the institute as considered above and a publication instance is considered a positive example for a given research group if any of the authors is affiliated with it (i.e. a multi-label setting in case of cross-research group publications).

Note that the information on the affiliations in the AIFB Portal is maintained by the institute's administration and can thus be considered a very clean learning problem while the data about people's research interests, projects or about paper metadata may be noisy or inconsistent because this kind of data is maintained autonomously by the researchers.

8.3.3 Results of the Person2Affiliation Experiments

In the first set of experiments, we investigated the performance of various kernel configurations for the two *person2affiliation* tasks. It is important to note that a small number of the Person individuals with affiliations are, however, not described at all in terms of assertions other than their affiliation and basic data like name and eMail

[8]The *"Man-Machine systems/Usability Engineering"* group is ignored, because it has only one associated person.

Table 8.2 — Kernel configurations for SWRC experiments on the *person2affiliation* tasks.

Short name	Description
proj	Cosine-normalized kernel on the object property project with the identity kernel as base kernel.
pub	Same for the object property publication.
top	Same for the object property workedOnBy^{-1}, i.e. linking from instances to topic instances.
proj+top+pub	Sum of the three kernels *pub*, *proj*, and *top*
class+proj+top+pub	Sum of the three kernels *pub*, *proj*, and *top* and an additional common class kernel with additive weight 0.1.
class+proj+top	Same, but without *pub*.
pub.title.bow	Like *pub*, but with cosine normalized bag–of-words type kernel on the title data property of target publications as base kernel.
class+proj+top+pub.title.bow	like *class+proj+top+pub* but using *pub.title.bow* instead of *pub*.
pub.author	Cosine normalized object property kernel on the publication object property, with an object property kernel on the author property as base kernel.
pub.author.pub	As *pub.author*, but with an additional object property kernel on the publication property as base kernel.
(pub+1)(top+1)*(proj+1)*	Multiplicative combination of the three kernels *pub*, *proj*, and *top*, each with an additional constant of 1.

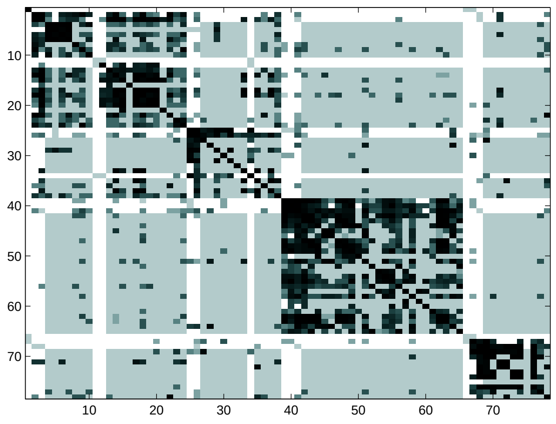
address which does not help in discriminating the research groups. For these individuals it will be generally hard to build adequate models such that the best classification that is theoretically achievable is below 100% accuracy a-priori.

Description of Kernel Configurations

Table 8.2 on page 184 summarizes the investigated kernel configurations. Obviously, none of the kernels used did rely directly or indirectly on the target property (affiliation). The basic kernels *proj*, *pub* and *top* are object property kernels associated with the immediate descriptions available about the person instances, namely the projects they are working on, the publications they have authored or the topics they are working on. The other kernel configurations either combine the elementary kernel functions or refine them by means of a specific choice of the employed base kernels. For example, the configuration *pub.author* corresponds to an object property kernel on the publictation object property that links from authors to their publications which again exploits another object property kernel on the author object property which points to all authors of the respective publication. In this case, the feature space effectively corresponds to all instances that stand in a co-author relationship to the argument instances. As another example, the configuration *class+proj+top+pub* combines a common class kernels and all of the three basic object property kernels mentioned above.

Visual Comparison of Kernel Configurations

Figure 8.4 on page 186 shows heatmap visualizations of the kernel matrices of the instances corresponding to 78 active AIFB researchers based on the *class+proj+top+pub* and the *pub.author.pub* kernel configurations which perform best in the experiments reported on below. Both visualizations clearly mirror the structure of the institute in terms of four independent research groups. On the first sight, the *class+proj+top+pub* configuration (left) seems to generate more compact groups while on the second sight, we note that it does not provide large mutual similarity values for the instances in the second research group. It also appears that this representation induces a bit more noise than the *pub.author.pub* configuration on the right as some of the kernel values accross groups also show high values. These could, for example be due to overlapping topics. The representation on the right does show a more compact representation of the second group but on the other hand seems to split the first group into a small and a larger group. Indeed, the first group corresponds to the *"Business Information and Communication Systems"* research group at AIFB which is run by two group leaders and has two subgroups.



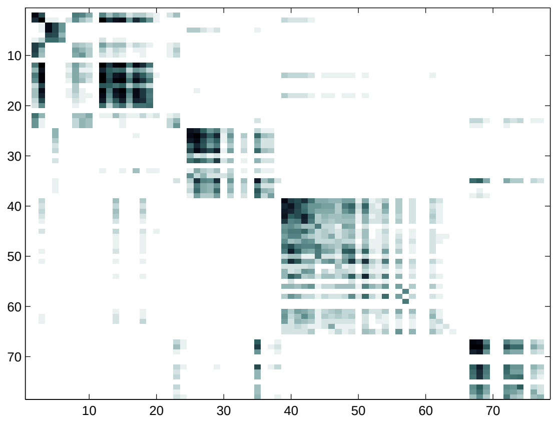


Figure 8.4 — Heatmap visualization of the kernel matrix of the instances corresponding to 78 active AIFB researchers. Visualizations are based on the *class+proj+top+pub* (top) and the *pub.author.pub* (bottom) kernel configurations. Each matrix cell corresponds to the evaluation of the kernel on a particular pair of instances. Darker shadings indicate higher values of the kernel evaluation. Instances are ordered according to their research group affiliations.

Experimental Results

We now turn our attention to the actual classification experiments. For each learning problem and kernel configuration, we performed 4 binary classification experiments, one for each research group using SVMs with soft margin parameters $C = 1$ and $C = 10$. To account for the small number of training instances, performance estimations did not rely on a fixed training/test split of the dataset but were estimated via the leave–one–out (LOO) cross-validation strategy.

Table 8.3 on page 188 summarizes the macro-averaged results of these classification experiments.[9] From the results, we generally note that for all kernel functions, the precision easily reaches levels of above 90% while recall values are generally worse. Further, as a general observation, the kernel configurations based on individual object properties, i.e. *pub*, *proj* and *top* perform rather poorly. In contrast, the combined kernel functions can substantially improve the results. When comparing the *person2affiliation (employee)* and *person2affiliation (all)* settings, we note that the *class+proj+top+pub* configuration performs better better than the *proj+top+pub* on the *person2affiliation (employee)* task, while the opposite is the case for the *person2affiliation (all)* task, suggesting that the additional class information helps to distinguish employees from non-employees while the distribution to the groups is probably due the other kernel components.

In fact, the *class+proj+top+pub* configuration is the best performing configuration for the *person2affiliation (employee)* setting at a F_1 level of 70.8% for C=1. For the *person2affiliation (all)* setting, the best results are achieved for the *pub.author.pub* configuration at F_1 levels of 86.3% and 85.6% for C=10 and C=1, respectively. Recently Kiefer et al. (2008) have also considered the *person2affiliation (all)* setting in the context of classification experiments with statistical relational learning techniques, specifically the relational Bayesian classifiers of Neville et al. (2003). The best result reported there is a F_1 measure of 83.7%, which is thus outperformed by the *pub.author.pub* kernel configuration. However, the experiments described there use a single-label classification setting such that the results are not fully comparable.

Recalling from the reasoning above that some of the Person individuals in the knowledge base do not have any publication, project or topic information associated with them, the results also constitute a pessimistic estimate of the performance in cases where full information would be available.

[9]Here, the choice of the macro-averaging scheme is rooted in practical considerations. In particular, we have exploted the LOO estimation feature of SVMLIGHT for the computation of the class-wise accuracy, precision and recall values as described by Joachims (2000) which were then used directly to compute the macro-averages.

Table 8.3 — Leave-one-out classification results for the *persons2affiliation (employee)* problem on SWRC for different kernels and kernel modifiers. All numbers are percentages.

(a) person2affiliation (employee)

	$C = 1$				$C = 10$			
Configuration	A	P	R	F_1	A	P	R	F_1
pub	92.3	98.1	26.3	38.5	93.4	87.0	46.8	60.5
proj	94.4	95.8	47.6	58.8	94.2	81.5	49.6	60.1
top	91.3	82.2	32.8	44.9	90.9	68.4	41.3	50.0
proj+top+pub	94.4	92.2	50.5	63.0	94.4	88.1	52.3	64.4
class+proj+top+pub	95.5	95.8	58.1	70.8	94.8	90.9	54.2	66.9
class+proj+top	94.5	93.1	53.6	66.9	93.5	86.9	49.8	62.2
pub.title.bow	93.7	79.0	54.4	63.7	93.1	74.9	53.7	62.1
class+proj+top+pub.title.bow	95.5	95.9	57.3	69.5	95.1	89.8	57.1	68.5
pub.author	93.1	75.4	54.6	63.1	93.5	74.1	67.2	69.8
pub.author.pub	93.8	77.9	65.7	70.5	93.4	74.1	61.8	67.2
(pub+1)(top+1)*(proj+1)*	92.8	87.6	41.4	55.2	93.0	87.0	43.3	56.6

(b) person2affiliation (all)

	$C = 1$				$C = 10$			
Configuration	A	P	R	F_1	A	P	R	F_1
pub	83.2	100.0	26.0	39.3	84.1	86.4	58.0	62.6
proj	82.9	95.8	35.1	50.0	82.2	82.1	54.3	58.5
top	84.4	92.6	40.4	55.8	83.4	82.2	56.3	61.8
proj+top+pub	87.8	91.7	60.3	67.8	86.5	90.3	60.8	67.1
class+proj+top+pub	87.4	91.4	59.7	67.4	87.1	88.8	63.1	70.1
class+proj+top	88.6	93.2	59.6	68.6	87.1	88.4	62.5	70.4
pub.title.bow	88.5	82.2	65.8	70.0	88.6	81.0	65.8	70.1
class+proj+top+pub.title.bow	89.5	92.3	64.0	71.6	89.2	89.8	67.0	74.1
pub.author	93.0	93.8	78.5	83.8	93.1	93.6	79.3	84.1
pub.author.pub	93.7	94.0	81.1	85.6	94.2	94.4	81.9	86.3
(pub+1)(top+1)*(proj+1)*	87.9	92.1	61.7	69.2	86.0	86.8	63.6	68.9

8.3.4 Results of the Paper2Affiliation Experiments

In the second set experiments, we investigated the performance of various kernel configurations for the *paper2affiliation* task.

Description of Kernel Configurations

Table 8.4 on page 190 summarizes the kernel configurations investigated for this task. As a kind of baseline, we report results on the *title.bow* kernel, which corresponds to the bow cosine kernel on the title datatype property of the publications. The other kernels combine the normalized common class kernel (again with weight 0.1), with normalized object property kernels on the isAbout (pointing to topics), author and hasProject properties. This time we have also employed different kernel modifiers, namely the plain sum of the component kernels without their individual normalization (*-plain*), and *pg1* and *pg3* kernel modifiers which correspond to the Gaussian modifiers applied to the plain sum of kernel components with bandwidth parameters σ equal to 1 or 3.

Experimental Results

For each learning problem and kernel configuration, we again performed 4 individual binary classification experiments with soft margin parameters $C = 1$ and $C = 10$. Further the performance metrics were again estimated via the Leave-One-Out cross-validation strategy.

Table 8.5 on page 190 summarizes the macro-averaged results of the classification experiments. In summary, the *papers2affiliations* task has achieved virtually optimal results that are stable over the different kernel variants whereby the best F_1 result is achieved for the *class+author+top+title.bow (plain)* configuration and C=10 at a level of 97.5%.[10] These good results can be traced to the fact that the object properties pointing to associated authors have been included in the kernel computation, inherently bearing a strong correspondence to the research groups. As a general observation we note that the use of Gaussian modifiers does, at least for the investigated configurations, not improve upon the kernel results.

[10]Unfortunately, the results of Kiefer et al. (2008) on this task are not comparable because in the reported setting the original classification problem was changed in so far as a given paper was assigned to the affiliation of the majority of the authors.

Table 8.4 — Kernel configurations for SWRC experiments on the *paper2affiliation* tasks.

Short name	Description
title.bow	Cosine-normalized bag–of-words kernel on the title data property
class+author+top	Sum of cosine-normalized common class kernel (with weight 0.1), and object property kernels on isAbout (i.e. linking to topics) and author properties with identity kernel as base kernel.
class+author+top+proj	As *class+author+top* but with an an additional object property kernel on the hasProject property.
class+author+top+title.bow	As *class+author+top* but with an an additional *title.bow* data property kernel.
{*} (plain)	Any kernel of the above without cosine normalization of the component kernels.
{*} (G σ)	Any kernel of the above without cosine normalization of the component kernels but with post-modification by a Gaussian Kernel modifier with bandwidth parameter σ.

Table 8.5 — Leave-one-out classification results for the *papers2affiliation* problem on SWRC for different kernels and kernel modifiers. All numbers are percentages.

	$C = 1$				$C = 10$			
Configuration	A	P	R	F_1	A	P	R	F_1
title.bow	92.5	86.8	47.8	56.0	93.8	91.1	63.1	72.4
class+author+top	99.3	99.8	94.5	97.1	99.4	99.2	95.0	97.0
class+author+top (plain)	99.3	99.7	95.1	97.3	99.4	99.7	95.2	97.4
class+author+top (G 1)	93.2	96.4	57.0	67.8	93.9	96.8	61.2	72.1
class+author+top (G 3)	98.6	99.6	90.1	94.5	99.3	99.8	94.9	97.3
class+author+top+proj	99.3	99.5	94.5	96.9	99.3	98.8	94.9	96.8
class+author+top+proj (plain)	99.2	99.8	94.7	97.1	99.3	99.7	95.0	97.3
class+author+top+proj (G 1)	92.4	95.9	53.1	63.8	93.4	96.3	60.1	70.9
class+author+top+proj (G 3)	98.6	99.6	90.3	94.6	99.2	99.6	94.6	97.0
class+author+top+title.bow	99.3	99.6	94.5	97.0	99.2	99.7	94.5	97.0
class+author+top+title.bow (plain)	99.4	99.7	95.4	97.5	99.4	99.7	95.4	97.5

8.4 Mining the Cora Dataset

In this section, we consider a further small experiment on the CORA dataset compiled by McCallum et al. (2000). The results reported in this section are based on experiments which are also reported in a recent report (Bloehdorn and Sorg, 2008).

8.4.1 Task and Dataset

The dataset we consider is the CORA classification dataset compiled from the example application of an intelligent web portal (McCallum et al., 2000).[11] The dataset consists of about 50,000 scientific publications classified into several thematic categories. For each of the documents, basic bibliographical metadata, i.e. paper abstracts and author information are provided. Furthermore, relational information about the mutual citation among the papers is provided. While the original dataset is not described with respect to a formal Semantic Web-type ontology the dataset shows an interesting relational structure that can readily be translated into RDF or and Semantic Web formalisms.

We shortly review the primitives of the knowledge structure that were considered in the experiments. The dataset consists of instances of type document. These instances are described by the datatype properties title and abstract which link to titles and abstracts of the documents, respectively. Documents are further linked to instances of type Person by means of the object property author and to other documents by means of the cites property with the obvious interpretations, i.e. the relation to the document authors and their citation relationships. For the purpose of our experiments, we introduce a generic citation property and the axioms (i) $\mathsf{citation} \sqsupseteq \mathsf{cites}$ and (ii) $\mathsf{citation} \sqsupseteq \mathsf{cites}^{-1}$, i.e. the new citation property comprises all (undirected) citation relations between any two document instances.

In the experiments, we aimed at comparing the performance of classical VSM representation of the document abstracts versus kernel functions based on the relational information provided by citation and author relations.

8.4.2 Experimental Setup

The experiments are based on a subset of the CORA dataset similar to the subset used by Lu and Getoor (2003). The target categories correspond to 7 subcategories of the category *"Computer Science"* containing about 4,500 publications. Any documents which did not contain any mutual citation information were removed, effectively yielding a number of 3,641 document instances for publication. For the experiments the dataset was randomly split into four parts whereby all 7 target classes were ensured to have the same share of positive examples in all splits. The first three parts

[11] The dataset is available for download from `http://www.cs.umass.edu/~mccallum/data/`.

were used for training and parameter tuning while the fourth part was used for the evaluation.

In the experiments, we considered the following kernel functions. First of all, as a kind of a baseline, the configuration *abstract.bow* refers to a cosine normalized datatype property kernel on the abstract property, using the standard Bag-of-Words (BOW) representation with the inner product as base kernel. Next, the kernel *author.publication* corresponds to an object property kernel on the author property, with an embedded object property kernel on the publication property. This means that the overall kernel counts the number of joint documents that have common authors with the argument instances. We primarily investigate the repeated nesting of this kernel whereby *author.publication*^{n} corresponds to nesting the *author.publication* n times within itself, i.e. it points to all publications that are exactly n steps away in a graph whereby the links constitute common authors between two publications. The overall kernel *author.publication*^{n*} is then assembled as the weighted sum β^1 *author.publication*^{1} $+\beta^2$ *author.publication*^{2} $+ \ldots + \beta^n$ *author.publication*^{n}. Hereby, $\beta < 1$ specifies a decay factor to account for targets further away from the reference individuals. Similarly, the kernel *citation* corresponds to an object property kernel on the citation property, whereby *citation*^{n} and *citation*^{n*} are defined analogously to the previous case. In all cases, the overall kernels were normalized using the cosine normalization modifier.

The procedure taken for the experiments was as follows. For all kernel configurations, preliminary experiments were conducted using different parameters β and n as well different SVM soft margin parameters C on two quarters of the overall dataset and evaluated on a third quarter. After optimization of the parameters with respect to the training error on the evaluation set, a SVM was trained on all three parts and the generalization capability was evaluated on the fourth quarter.

For comparison, we ran also experiments based on the *diffusion kernel* proposed by Kondor and Lafferty (2002). Like the object property kernels, this kernel makes use of the graph structure but aims at capturing its overall structure by means of exponentials of the graph adjacency matrix. The kernel also takes a single parameter, β which was optimized in analogously to the parameters above. Refer to the exposition by Kondor and Lafferty (2002) for more information on this kernel function.

8.4.3 Experimental Results

The micro-averaged results of the experiments are shown in Table 8.6. As we can see, both types of kernels based on the object property relations perform better than the basic VSM representation of the titles. In particular, at a F_1 level of 83.2% the object property kernels on citations perform better in the classification problem than those based on joint authors at a F_1 level of 71.6%. In fact, the graph of mutual citations is quite dense and almost completely connected. In contrast, the common author graph

Table 8.6 — Results of the classification experiment on the CORA publication dataset using individual kernels.

Configuration	A	μP	μR	μF_1
abstract.bow	90.9	90.5	40.8	56.3
citation*	95.5	89.2	78.0	83.2
author.publication*	92.8	82.1	63.5	71.6
citation* (diffusion)	95.3	88.3	77.0	82.3
author.publication* (diffusion)	92.0	84.2	53.8	65.6

has fewer links and consists of many unconnected subgraphs. The citation graph seems therefore to be more appropriate to be used in this classification experiment as input for kernel functions on the respective object properties. The table also includes results based on the diffusion kernel by Kondor and Lafferty (2002) which are, however not superior to the use of the (local) object property kernels.

8.5 Summary and Discussion

In this chapter, we have discussed the application of the kernel framework introduced in Chapter 7 on three real-world datasets. In the following, we summarize the main findings:

- While all three different tasks have addressed different learning targets and used different representations, we have seen that the kernel framework proposed in Chapter 7 allows to naturally define powerful kernel functions.
- As the task of learning from Semantic Web data has not yet been addressed actively, comparative evaluations are difficult due to the lack of standardized evaluation data sets. The experiments reported on this section thus need to be seen mostly as an exploratory approach to evaluating the power of different kernel configurations.
- For the SWRC dataset, the results are however competitive with new results reported in literature recently while for the CORA dataset, the results do outperform the common VSM representation.

Open issues along the work reported in this chapter include the practical application of the kernel framework on further datasets. The datasets in this section, especially the SWRC dataset, at the same time provide an anchor point for experiments with future systems on Semantic Web datasets.

In any case, the results reported in this chapter demonstrate that powerful kernel functions on ontological instance data can be designed flexibly by exploiting the available modelling primitives within the ontologies considered.

Part V

Conclusion

Chapter 9

Conclusion

Machine learning techniques are a successful engineering paradigm for a large class of practical data analysis problems. Ontologies and knowledge structures allow to model a domain of interest in terms of declarative knowledge. While these formalisms have been studied for some time, principled approaches for the exploitation of knowledge structures within machine learning settings are still a major subject of research.

In this thesis, we have recasted the question of the combination of these two fields into the field of kernel methods. Kernel methods are a successful paradigm for incorporating knowledge about the domain of interest into classical machine learning techniques or for interacting with data structures that do not lend themselves naturally to a vector-based representation. The core of this approach is the definition of appropriate kernel functions for a given scenario. Conceptually, kernel functions can be regarded as special purpose similarity functions that implicitly allow an interpretation as dot products of vectors of real numbers in a corresponding feature space. The so-called kernel trick refers to the idea of rewriting standard learning algorithms entirely in dual form such that all information about the geometry of the feature space and the learned models can be expressed in terms of pairwise dot products or, equivalently, in terms of kernel functions. Via the paradigm of kernel functions, this thesis combines techniques from the area of knowledge representation and machine learning in a principled way. In the following, we summarize the work of this thesis and sketch paths for future research activities.

9.1 Summary of Contributions

This thesis started out with a comprehensive introduction to both underlying research areas, namely machine learning with kernel methods and formalisms for knowledge structures. The main contribution of this thesis has been the design of kernel functions that either use knowledge structures as complementary background knowledge or directly operate on instances defined within these knowledge structures themselves. On the theoretical side, we have (i) analyzed existing similarity functions for entities in taxonomic structures in terms of their suitability as kernel functions, (ii) proposed two types of expressive and adequate kernel functions for textual data that employ linguistic background knowledge resources to address problems of variability in natural language, and (iii) proposed a framework for kernel functions on data items that are formally described as instances of an ontological structure. On the practical side, we have (iv) evaluated the kernel functions proposed in (ii) and (iii) in an extensive series of experiments. In the following, we take up on the main contributions of this thesis in closer detail.

Kernel Functions for Entities in Taxonomic Structures

On the basis of a theoretical analysis, Chapter 4 of this thesis has investigated the question whether, or under which conditions, the most prominent and well-motivated similarity functions for entities in taxonomic structures can be given an interpretation as kernel functions. This analysis has been conducted by showing whether the respective similarity functions conform to positive semi-definite functions. The results of this section provide the first comprehensive analysis of taxonomic similarity functions with respect to this question. While the results are both theoretically and practically interesting for themselves, they also lie at the core of the kernel functions discussed in the subsequent parts. In particular, this question is important to decide whether these similarity functions can be used as smoothing parameters for Semantic Smoothing Kernels or Semantic Syntactic Tree Kernels.

Kernel Functions for Semantic Smoothing in Text Mining

Chapter 5 of this thesis has then systematically investigated the concept of Semantic Smoothing Kernels as a technique for minimizing the effects of the variability of natural language in the common Vector Space Model for text mining applications. The analysis has been conducted both in terms of their theoretical interpretation and practical properties. The same chapter of this thesis has then introduced Semantic Syntactic Tree Kernels as a generalization of tree kernel functions which directly combines semantic background knowledge with the analysis of syntactic structure of textual inputs. To the best of the author's knowledge, Semantic Syntactic Tree Kernels constitute the first principled framework for kernel functions that build upon linguistic structure and background knowledge about the semantic dependencies of terms at the same time.

Semantic Smoothing Kernels and Semantic Syntactic Tree Kernels have then been implemented as part of a modular software infrastructure. Subsequently, both methods have been evaluated in an extensive experimental analysis in terms of three sets of text classification experiments. As the analysis showed, Semantic Smoothing Kernels are an effective and powerful paradigm for leveraging the performance of text classifiers in situations where training data is scarce and the underlying patterns are not dominated by a small set of frequently occurring terms. This finding suggests that Semantic Smoothing Kernels can significantly reduce the labelling effort for training data thus allowing for "fast prototyping" of new classifiers. For text mining settings in which more training data is available, the positive effect of Semantic Smoothing Kernels fades bit by bit. Similarly, the experiments on the question classification setting suggest that the smoothing component within Semantic Syntactic Tree Kernels leads to a superior performance compared to ordinary tree kernels. For the dataset used, the results also outperform all results reported in literature so far. As, for both types of kernels, we have not used any proper word sense disambiguation strategy, the reported results also need to be seen as a pessimistic estimate of their potential on fully disambiguated output.

Kernel Functions for Instance Data in Ontologies

Chapter 7 of this thesis has addressed the question how instances which are formally described within ontological knowledge structures can become the subject of machine learning techniques. In particular, the chapter has introduced a framework for designing kernel functions on instance data in ontologies that is on the one hand flexible but on the other hand ensures the validity of the corresponding kernel regardless of parameter choices. Again, on the practical side, these kernel functions for instance data in ontologies have been implemented as software modules as part of the KAON2SIMILARITY API. The application of these kernel functions has been demonstrated in three sets of practical experiments in Chapter 8 of this thesis. The analysis has demonstrated that powerful kernel functions on ontological instance data can be designed flexibly and intuitively by exploiting the available modelling primitives within the ontologies considered. As the task of learning from Semantic Web data has not yet been addressed actively, comparative evaluations are not easy due to the lack of standardized evaluation data sets. For the SWRC dataset, the results are however competitive with new results reported in literature recently. For the CORA dataset, the results show that kernels defined within the proposed framework do outperform the popular Vector Space Model (VSM) representation.

9.2 Outlook

In this section, we investigate some of the paths for future work at the intersection of kernel-based machine learning and formal knowledge representation.

Kernel Functions for Mining Whole Knowledge Structures

While this thesis has investigated means for mining both, instances external to a given knowledge structure and internal to a given knowledge structure, the classification of approaches for machine learning with knowledge structures in Chapter 1 has already pointed to a third alternative, namely the mining of whole knowledge structures, i.e. sets of statements. Practical applications of this direction of work lie in the discovery of mappings between knowledge structures or fragments thereof (Ehrig, 2007; Udrea et al., 2007), in the automated discovery of "interesting" substructures (Ramakrishnan et al., 2005) or in predictive settings, e.g. the prediction of the reasoning complexity of an ontology.

Existing work in the direction of defining kernel functions for knowledge structures as a whole have usually neglected any semantic aspects and focused on the syntactic graph structure. However, in contrast to special purpose graphs like the case of trees investigated in Chapter 5, kernels for arbitrary graphs have proved to be more difficult to design.

Gärtner et al. (2003a); Gärtner (2005) provide an analysis which suggests that computing a complete graph kernel, i.e. a kernel which is capable of taking onto account the full structure of the argument graphs is at least as hard as solving the graph isomorphism problem. The analysis also shows that computing an inner product in a feature space indexed by all possible graphs, where each feature counts the number of isomorphic subgraphs is NP-hard. These results suggest that it will be very unlikely to find sufficiently efficient kernel functions that are based on the full structure of the argument graphs. At the same time, these findings motivate the investigation of kernel functions where the requirements on the structure of the graphs are weakened or where the requirements on the structures to be discovered are weakened. In the same work, Gärtner et al. (2003a) suggest kernels based on walks within the graph, which can be computed from the product graph of the arguments in polynomial time. The suggested kernels on labelled pairs assume that only the distance between the pairs of nodes (of some label) is of interest while those based on contiguous walks in the argument graphs implicitly define a feature space where each dimension corresponds to a particular label sequence and a graph is represented by counting for each of these sequences how many walks in a graph match this sequence. Kashima et al. (2003) propose a similar kernel on transition graphs. Horváth et al. (2004) propose kernel functions for a different group of descriptive features, namely features based on the cyclic patterns within a graph. Although kernels based on cyclic patterns are generally not efficiently computable, several restrictions aid the efficient computation in special cases.

Future work in the direction of mining whole knowledge structures will be based to some extent on existing graph kernels but will also need to take the formal semantics of the argument structures into account.

Formal Knowledge Structures for Feature/Kernel Specification

A different research direction is the use of formal knowledge structures to design kernel functions in the first hand. Along this line, Cumby and Roth (2003a) have proposed to use a simple Description Logic to describe the desired structure of feature spaces for given learning problems in terms of a feature description language. Later, Cumby and Roth (2003b) show that some of the explicit feature encoding can be performed implicitly by defining corresponding kernel functions which are parametrized by the given feature descriptions.

The basic approach of designing knowledge structures for guiding a kernel function in computing their results appears to be a promising line for future research. Challenges on this way include to move from the comparatively simple Description Logic (DL) employed by Cumby and Roth (2003b) to more expressive DLs along the lines of current activities in the context of the Semantic Web research area.

Learning with Indefinite Kernel Functions

Much of the work of this thesis was occupied with the question whether and how the kernel function under investigation can be ensured to be *valid*, i.e. to constitute positive semi-definite functions on the input arguments. In Section 2, we have derived this requirement as a direct consequence of the implicit representation in a Reproducing Kernel Hilbert Space (RKHS). This interpretation provides the basis for the proper analysis of many models and algorithms in terms of statistical learning theory and most kernelized learning algorithms directly build on the notion of such an inner product space, such as e.g. the margin maximization principle for Support Vector Machines (SVMs).

Nevertheless, it seems natural to wonder in how far kernel methods could be used in conjunction with arbitrary symmetric similarity functions, even if not positive semi-definite. Despite the missing interpretation, various studies report results on experiments with indefinite kernel functions (Haasdonk, 2005, and references therein). However, the results of these studies are mixed, indicating performance gains in selected situations but worse results in others.

Haasdonk (2005) provide a theoretical and geometrical framework in terms of pseudo-Euclidean spaces and provide an analysis of the behaviour of SVMs in situations of indefinite kernel functions. The analysis shows that SVMs can produce optimal hyperplane classifiers not by margin maximization, but by minimization of distances between convex hulls in pseudo-Euclidean spaces. The analysis also provides rough criteria, for example the number of negative eigenvalues of the kernel matrix, for checking whether a given indefinite kernel is promising.

Ong et al. (2004) provide a first deeper analysis of the general implications of indefinite kernel functions in terms of the Reproducing Kernel Kreĭn Space (RKKS), a generalized type of functional space which corresponds to the RKHS in selected aspects.

Using indefinite kernels has not only consequences for the geometrical interpretation but also on the practical implications for the numerical optimization problem, as convexity is lost. Specifically, in the case of SVMs, the optimization is usually solved with quadratic programming methods. However, as in the cases no unique global optimum exists, quadratic programming approaches are in most cases not able to find satisfying solutions at all or do not even terminate. Along this line, Mierswa (2006)has proposed a first practical solution to the issue of learning SVMs equipped with potentially indefinite kernel functions. The issue is addressed by approaching the constrained optimization problem of the SVM by means of evolutionary algorithms which produced good initial results.

Whether these results will open up the road towards arbitrary kernel functions that still exhibit the statistical stability is questionable. The careful investigation whether the constraint of positive semi-definiteness can be relaxed in selected cases is, however, a line of interesting research that will also naturally affect the field of kernel functions for knowledge structures.

Learning with Structured Output Spaces

Along a different line, we now turn our attention away from kernel functions in particular and sketch another road for the interaction of machine learning and formal knowledge structures.

Recently, there has been increased interest a field called *Learning with Structured Output Spaces*. In contrast to the classical supervised learning setting, the target variable is not constrained to take the form of a single value but can have a complex structure (Tsochantaridis et al., 2004). Without going into the technical details of this area, it has to be remarked that these learning problems — at their core — also boil down to the one-variable case but take advantage of an elegant formulation of the learning problem as a whole, together with algorithms tailored to the specific setting. While such techniques can on the one hand be used to directly optimize SVM classification with respect to specific performance measures like precision or recall (Joachims, 2005), they can, on the other hand, provide a tool for incorporating knowledge about the output space. Extending existing ideas from classification into hierarchically organized classes (Cai and Hofmann, 2004), it would be interesting to investigate cases where the output classes can be structured by mutual logical constraints, such as in ontologies based on expressive DLs.

9.3 Final Remarks

In summery, this work has provided several elements of a kernel infrastructure for formal knowledge structures. It can be seen as a contribution to the vision of hybrid intelligent systems which jointly build on formal knowledge-based and informal statistical paradigms of artificial intelligence, i.e. systems that combine deductive and inductive reasoning techniques. While the increased availability of data sources on the World Wide Web, the increased availability of formal domain models as part of the envisioned Semantic Web and the constant advance in computing power contribute to this goal, several additional research questions have to be addressed to fully achieve it. Together with several ongoing complementary research activities in the area of kernel functions for structured data, this thesis provide a solid basis for initial practical application as well as for future research.

Part VI

Appendix

Appendix A

Implementation

JNI Adapter for SVMLIGHT and Kernel Library

This software is an extension of the SVMLIGHT software (V6.01) developed by Joachims (1999) available at

`http://svmlight.joachims.org/`

which constitutes one of the most common implementations of the SVM algorithms. The extension, which is available at

`http://www.aifb.uni-karlsruhe.de/WBS/sbl/software/jnikernel/`

provides an interface to kernel functions that are implemented in JAVA by means of the Java Native Interface (JNI) Invocation API. Technically, the custom C/C++p kernel module of SVMLIGHT now becomes a proxy that is tightly coupled to the JAVA class `edu.unika.aifb.jnikernel.KernelManager`. This class delegates all kernel evaluation calls to a user-specified JAVA class that implements the `edu.unika.aifb.kernels.api.Kernel` interface. Thus, SVMLIGHT can directly (i.e. without further modifications) work with any Java class that implements this interface. The interface can be used to implement whatever kernel, as long as it implements the `edu.unika.aifb.kernels.api.Kernel` interface.

Semantic Smoothing Kernel Adapter for SVMLIGHT

For historical and performance reasons, the plain semantic smoothing kernels are also available as a standalone plugin for SVMLIGHT, implemented in C++. This kernel plugin is available at:

`http://www.aifb.uni-karlsruhe.de/WBS/sbl/software/semkernel/`

together with accomanying documentation.

Appendix B

Penn Treebank Tagset

For a full documentation, please refer to the Penn Treebank Manuals (Santorini, 1990; Bies, 1995).

Clause Level Annotations

S	simple declarative clause, i.e. one that is not introduced by a (possible empty) subordinating conjunction or a wh-word and that does not exhibit subject-verb inversion. SBAR - Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ	Direct question introduced by a wh-word or a wh-phrase. Indirect questions and relative clauses should be bracketed as SBAR, not SBARQ. SINV - Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal.
SQ	Inverted yes/no question, or main clause of a wh-question, following the wh-phrase in SBARQ.

Phrase Level Annotations

ADJP	Adjective Phrase.
ADVP	Adverb Phrase.
CONJP	Conjunction Phrase.
FRAG	Fragment.
INTJ	Interjection. Corresponds approximately to the part-of-speech tag UH.
LST	List marker. Includes surrounding punctuation.

NAC	Not a Constituent; used to show the scope of certain prenominal modifiers within an NP.
NP	Noun Phrase.
NX	Used within certain complex NPs to mark the head of the NP. Corresponds very roughly to
N-bar	level but used quite differently. PP - Prepositional Phrase.
PRN	Parenthetical.
PRT	Particle. Category for words that should be tagged RP.
QP	Quantifier Phrase (i.e. complex measure/amount phrase); used within NP.
RRC	Reduced Relative Clause.
UCP	Unlike Coordinated Phrase.
VP	Vereb Phrase.
WHADJP	Wh-adjective Phrase. Adjectival phrase containing a wh-adverb, as in how hot.
WHAVP	Wh-adverb Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing a wh-adverb such as how or why. WHNP - Wh-noun Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing some wh-word, e.g. who, which book, whose daughter, none of which, or how many leopards.
WHPP	Wh-prepositional Phrase. Prepositional phrase containing a wh-noun phrase (such as of which or by whose authority) that either introduces a PP gap or is contained by a WHNP. X - Unknown, uncertain, or unbracketable. X is often used for bracketing typos and in bracketing the...the-constructions.

Word Level Annotations

CC	Coordinating conjunction
CD	Cardinal number

DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun (prolog version PRP-S)
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form

VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun (prolog version WP-S)
WRB	Wh-adverb

List of Tables

List of Figures

List of Acronyms

API	Application Programming Interface
BOW	Bag-of-Words
CD	Conceptual Density
DAG	directed acyclic graph
DC	Dublin Core
DL	Description Logic
ECOC	error correcting output code
EM	expectation maximization
ERM	empirical risk minimization
FOAF	Friend of a Friend
IC	information content
IDF	Inverse Document Frequency
IE	information extraction
ILP	Inductive Logic Programming
IR	information retrieval
JNI	Java Native Interface
kNN	*k*-Nearest Neigbhour
LOO	leave–one–out
LSI	Latent Semantic Indexing
LSK	Latent Semantic Kernel
LSO	Lowest Super Ordinate

MSC	most specific concept
NLP	natural language processing
OL	ontology learning
QC	question classification
QA	question answering
OWL	Web Ontology Language
p.s.d.	positive semi-definite
PLSI	Probabilistic Latent Semantic Indexing
POS	Part–of–Speech
RBF	Radial Basis Function
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RKHS	Reproducing Kernel Hilbert Space
RKKS	Reproducing Kernel Kreĭn Space
SPARQL	Simple Protocol and RDF Query Language
SKOS	Simple Knowledge Organisation Systems
SRM	structural risk minimization
SSK	Semantic Smoothing Kernel
SSTK	Semantic Syntactic Tree Kernel
SVD	singular value decomposition
SVM	Support Vector Machine
SWRC	Semantic Web Research Community
SWRL	Semantic Web Rule Language
TC	text classification
TFIDF	Term Frequency Inverse Document Frequency

TM	text mining
URI	Uniform Resource Identifier
VSM	Vector Space Model
WN	WordNet
WSD	word sense disambiguation
WWW	World Wide Web
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Bibliography

Agirre, E. and Rigau, G. (1996). Word sense disambiguation using conceptual density. In *Proceedings of the 16th Conference on Computational Linguistics (COLING '96), August 5-9, 1996, Copenhagen, Denmark*, pages 16–22. Association for Computational Linguistics, Morristown, NJ, USA.

Aizerman, M. A., Braverman, E. A., and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837.

Allen, J. (1995). *Natural Language Understanding*. Benjamin-Cummings Publishing, Redwood City, CA, USA, second edition.

Allwein, E. L., Schapire, R. E., and Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141.

Apté, C., Damerau, F., and Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, Cambridge, UK.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley-Longman, Harlow, UK.

Baker, L. D. and McCallum, A. K. (1998). Distributional clustering of words for text classification. In Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98), August 24-28 1998, Melbourne, Australia*, pages 96–103. ACM Press, New York, NY, USA.

Basili, R., Cammisa, M., and Moschitti, A. (2005a). A semantic kernel to classify texts with very few training examples. In Bloehdorn, S., Hotho, A., and Buntine, W., editors, *Proceedings of the Workshop on Learning in Web Search at the 22nd International Conference on Machine Learning (ICML 2005), August 7-11, 2005, Bonn, Germany*, pages 10–17. Published online in August 2005 at `http://cosco.hiit.fi/search/learninginsearch05/ICML_W4.pdf`.

Basili, R., Cammisa, M., and Moschitti, A. (2005b). A semantic kernel to exploit linguistic knowledge. In Bandini, S. and Manzoni, S., editors, *AI*IA 2005: Advances in Artificial Intelligence — Proceedings of the 9th Congress of the Italian Association for Artificial Intelligence, September 21-32, 2005, Milan, Italy*, volume 3673 of *Lecture Notes in Computer Science*, pages 290–302. Springer, Berlin–Heidelberg, Germany.

Basili, R., Cammisa, M., and Moschitti, A. (2006). A semantic kernel to classify texts with very few training examples. *Informatica*, 30(2):163–172.

Bekkerman, R., El-Yaniv, R., Tishby, N., and Winter, Y. (2001). On feature distributional clustering for text categorization. In Kraft, D. H., Croft, W. B., Harper, D. J., and Zobel, J., editors, *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01), September 9-13, 2001, New Orleans, Louisiana, USA*, pages 146–153. ACM Press, New York, NY, USA.

Berendt, B., Hotho, A., and Stumme, G. (2002). Towards semantic web mining. In Horrocks, I. and Hendler, J. A., editors, *The Semantic Web — Proceedings of the First International Semantic Web Conference (ISWC 2002), June 9-12, 2002, Sardinia, Italy*, volume 2342 of *Lecture Notes in Computer Science*, pages 264–278. Springer, Berlin–Heidelberg, Germany.

Berners-Lee, T., Fielding, R., and Masinter, L. (2005). Uniform resource identifier (URI): Generic syntax. Internet Engineering Task Force RFC 3986, Internet Society (ISOC). Published online in January 2005 at `http://tools.ietf.org/html/rfc3986`.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):34–43.

Bernstein, A., Kaufmann, E., Kiefer, C., and Bürki, C. (2005). SimPack: A generic Java library for similiarity measures in ontologies. Technical report, Department of Informatics, University of Zurich, Zurich, Switzerland.

Bhatia, R. (2007). *Positive Definite Matrices*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, USA.

Bies, A. (1995). Bracketing guidelines for Treebank II style — Penn Treebank project. Technical report, Department of Linguistics, University of Pennsylvania, Philadelphia, PA, USA.

Bisson, G. (1995). Why and how to define a similarity measure for object based representation systems. In Mars, N. J. I., editor, *Towards Very Large Knowledge Bases*, pages 236–246. IOS Press, Amsterdam, The Netherlands.

Bloehdorn, S., Basili, R., Cammisa, M., and Moschitti, A. (2006a). Designing semantic kernels as implicit superconcept expansions. In Schaaf, M. and Althoff, K.-D., editors, *LWA 2006: Lernen — Wissensentdeckung — Adaptivität, KDML 2006 :12. Workshop der Fachgruppe Knowledge Discovery, Data Mining und Maschinelles Lernen und des Arbeitskreises Knowledge Discovery, October 9-11, 2006, Hildesheim, Germany*, volume 1/2006 of *Hildesheimer Informatik-Berichte*, pages 255–261. Universität Hildesheim, Institut für Informatik, Germany.

Bloehdorn, S., Basili, R., Cammisa, M., and Moschitti, A. (2006b). Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*. IEEE Computer Society, Washington, DC, USA.

Bloehdorn, S., Cimiano, P., and Hotho, A. (2006c). Learning ontologies to improve text clustering and classification. In Spiliopoulou, M., Kruse, R., Nürnberger, A., Borgelt, C., and Gaul, W., editors, *From Data and Information Analysis to Knowledge Engineering: Proceedings of the 29th Annual Conference of the German Classification Society (GfKl 2005), March 9-11, 2005, Magdeburg, Germany*, volume 30 of *Studies in Classification, Data Analysis, and Knowledge Organization*, pages 334–341. Springer, Berlin–Heidelberg, Germany.

Bloehdorn, S., Cimiano, P., Hotho, A., and Staab, S. (2005). An ontology-based framework for text mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 20(1):87–112.

Bloehdorn, S., Haase, P., Sure, Y., and Völker, J. (2006d). Ontology evolution. In Davies, J., Studer, R., and Warren, P., editors, *Semantic Web Technologies — Trends and Research in Ontology-Based Systems*, chapter 4, pages 51–70. John Wiley & Sons, Chichester, West Sussex, UK.

Bloehdorn, S. and Hotho, A. (2004). Text classification by boosting weak learners based on terms and concepts. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, pages 331–334. IEEE Computer Society, Washington, DC, USA.

Bloehdorn, S. and Hotho, A. (2008). Machine learning and ontologies. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, chapter 31. Springer, Berlin–Heidelberg, Germany, second edition. To appear.

Bloehdorn, S. and Moschitti, A. (2007a). Combined syntactic and semantic kernels for text classification. In Amati, G., Carpineto, C., and Romano, G., editors, *Advances in Information Retrieval — Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007), 2-5 April 2007, Rome, Italy*, volume 4425 of *Lecture Notes in Computer Science*, pages 307–318, Berlin–Heidelberg, Germany. Springer, Berlin–Heidelberg, Germany.

Bloehdorn, S. and Moschitti, A. (2007b). Structure and semantics for expressive text kernels. In Silva, M. J., Laender, A. H. F., Baeza-Yates, R. A., McGuinness, D. L., Olstad, B., Olsen, O. H., and Falcao, A. O., editors, *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007), November 6-9, 2007, Lisbon, Portugal*, pages 861–864. ACM Press, New York, NY, USA.

Bloehdorn, S. and Sorg, P. (2008). Effective kernel function design for graph-structured instance data. Technical report, Institute AIFB, University of Karlsruhe (TH), Germany. Under Submission.

Bloehdorn, S. and Sure, Y. (2007). Kernel methods for mining instance data in ontologies. In Aberer, K., Choi, K.-S., Noy, N. F., Allemang, D., Lee, K.-I., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web — Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC 2007 + ASWC 2007), November 11-15, 2007, Busan, Korea*, volume 4825 of *Lecture Notes in Computer Science*, pages 58–71. Springer, Berlin–Heidelberg, Germany.

Borgida, A., Walsh, T., and Hirsh, H. (2005). Towards measuring similarity in description logics. In Horrocks, I., Sattler, U., and Wolter, F., editors, *Proceedings of the 2005 International Workshop on Description Logics (DL2005), July 26-28, 2005, Edinburgh, Scotland, UK*, volume 147 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In Haussler, D., editor, *Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT '92), July 27-29, 1992, Pittsburgh, PA, USA*, pages 144–152. ACM Press, New York, NY, USA.

Bouquet, P., Giunchiglia, F., Harmelen, F., Serafini, L., and Stuckenschmidt, H. (2003). C-OWL: Contextualizing ontologies. In Fensel, D., Sycara, K. P., and Mylopoulos, J., editors, *The Semantic Web — Proceedings of the Second International Semantic Web Conference (ISWC 2003), October 20-23, Sanibel Island, FL, USA*, volume 2870 of *Lecture Notes in Computer Science*, pages 164–179. Springer, Berlin–Heidelberg, Germany.

Bray, T., Hollander, D., Layman, A., and Tobin, R. (2006). Namespaces in XML 1.0. W3C recommendation, W3C. Published online on August 16th, 2006 at `http://www.w3.org/TR/2006/REC-xml-names-20060816`.

Brickley, D. and Guha, R. (2004). RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`.

Brickley, D. and Miller, L. (2007). FOAF vocabulary specification. Technical report, FOAF project. Published online on May 24th, 2007 at `http://xmlns.com/foaf/spec/20070524.html`.

Budanitsky, A. and Hirst, G. (2001). Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources at the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-01), June 2001, Pittsburgh, PA, USA*, pages 29–34. Carnegie Mellon University, Pittsburgh, PA, USA.

Budanitsky, A. and Hirst, G. (2006). Evaluating WordNet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Buitelaar, P., Cimiano, P., and Magnini, B., editors (2005). *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence*. IOS Press, Amsterdam, The Netherlands.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Cai, L. and Hofmann, T. (2003). Text categorization by boosting automatically extracted concepts. In Callan, J., Clarke, C., Cormack, G., Hawking, D., and Smeaton, A., editors, *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03), July 28 - August 1, 2003, Toronto, Canada*, pages 182–189. ACM Press, New York, NY, USA.

Cai, L. and Hofmann, T. (2004). Hierarchical document categorization with support vector machines. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management (CIKM '04), November 8-13, 2004, Washington, D.C., USA*, pages 78–87. ACM Press, New York, NY, USA.

Carlson, A., Cumby, C., Rosen, J., and Roth, D. (1999). The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA.

Chakrabarti, S. (2002). *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman Publishers, San Francisco, CA, USA.

Charles, W. G. (2000). Contextual correlates of meaning. *Applied Psycholinguistics*, 21:505–524.

Charniak, E. (2000). A maximum-entropy-inspired parser. In Wiebe, J., editor, *Proceedings of the First Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2000), April 29 - May 04, 2000, Seattle, Washington*, pages 132–139. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Cimiano, P. (2006). *Ontology Learning and Population from Text — Algorithms, Evaluation and Applications*. Springer, Berlin–Heidelberg, Germany. Originally published as PhD Thesis, 2006, Universität Karlsruhe (TH), Karlsruhe, Germany.

Cimiano, P., Hotho, A., and Staab, S. (2005). Learning concept hierarchies from text corpora using formal concept anaylsis. *Journal of Artificial Intelligence Research*, 24:305–339.

Collins, M. and Duffy, N. (2001). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In Isabelle, P., editor, *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02), July 7-12, 2002, Philadelphia, PA, USA*, pages 263–270. Association for Computational Linguistics, Morristown, NJ, USA.

Collins, M. and Duffy, N. (2002). Convolution kernels for natural language. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14 — Proceedings of the 2001 Neural Information Processing Systems Conference (NIPS 2001), December 3-8, 2001, Vancouver, British Columbia, Canada*, pages 625–632. MIT Press, Cambridge, MA, USA.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21– 27.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, UK.

Cristianini, N., Shawe-Taylor, J., and Lodhi, H. (2002). Latent Semantic Kernels. *Journal of Intelligent Information Systems*, 18(2-3):127–152.

Cruse, D. (1986). *Lexical Semantics*. Cambridge University Press, Cambridge, UK.

Culotta, A. and Sorensen, J. (2004). Dependency tree kernels for relation extraction. In Scott, D., editor, *Proceedings of the 42th Annual Meeting on Association for Computational Linguistics (ACL 2004), July 21-26, 2004, Barcelona, Spain*, pages 423–429. Association for Computational Linguistics, Morristown, NJ.

Cumby, C. M. and Roth, D. (2003a). Learning with feature description logics. In Matwin, S. and Sammut, C., editors, *Proceedings of the 12th International Conference on Inductive Logic Programming (ILP 2002), July 9–11, 2002, Sydney, Australia — Revised Papers*, number 2583 in Lecture Notes in Computer Science, pages 32–47. Springer, Berlin–Heidelberg, Germany.

Cumby, C. M. and Roth, D. (2003b). On kernel methods for relational learning. In Fawcett, T. and Mishra, N., editors, *Proceedings of the 20th International Conference on Machine Learning (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 107–114. AAAI Press, Chicago, IL, USA.

DCMI Usage Board (2006). DCMI metadata terms. DCMI recommendation, Dublin Core Metadata Initiative. Published online on December 18th, 2006 at `http://dublincore.org/documents/2006/12/18/dcmi-terms/`.

de Buenaga Rodriguez, M., Gómez-Hidalgo, J. M., and Diaz-Agudo, B. (1997). Using WordNet to complement training information in text categorization. In Nicolov, N. and Mitkov, R., editors, *Recent Advances in Natural Language Processing II: Selected Papers from the Second International Conference on Recent Advances in Natural Language Processing (RANLP 1997), March 25-27, 1997, Stanford, CA, USA*, Amsterdam Studies in the Theory and History of Linguistic Science, Series IV: Current Issues in Linguistic Theory, pages 353–364. John Benjamins Publishing, Amsterdam, The Netherlands.

Debole, F. and Sebastiani, F. (2004). An analysis of the relative hardness of Reuters-21578 subsets. *Journal of the American Society for Information Science and*, 56(6):584–596.

Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407.

Dietterich, T. G. and Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286.

Ehrig, M. (2007). *Ontology Alignment: Bridging the Semantic Gap.* Number 4 in Semantic Web and Beyond: Computing for Human Experience. Springer, New York, NY, USA. Originally published as PhD Thesis, 2006, Universität Karlsruhe (TH), Karlsruhe, Germany.

Ehrig, M., Haase, P., Stojanovic, N., and Hefke, M. (2005). Similarity for ontologies — a comprehensive framework. In Bartman, D., Rajola, F., Kallinikos, J., Avison, D., Winter, R., Ein-Dor, P., Becker, J., Bodendorf, F., and Weinhardt, C., editors, *Information Systems in a Rapidly Changing Economy: Proceedings of the 13th European Conference on Information Systems (ECIS 2005), May 26-28, 2005, Regensburg, Germany*. Association for Information Systems. Published online at `http://is2.lse.ac.uk/asp/aspecis/`.

Emde, W. and Wettschereck, D. (1996). Relational instance based learning. In Saitta, L., editor, *Proceedings 13th International Conference on Machine Learning (ICML 1996), July 3-6, 1996, Bari, Italy*, pages 122–130. Morgan-Kaufman Publishers, San Francisco, CA, USA.

Fanizzi, N. and d'Amato, C. (2006). A declarative kernel for ALC concept descriptions. In Esposito, F., Ras, Z. W., Malerba, D., and Semeraro, G., editors, *Foundations of Intelligent Systems, 16th International Symposium,*, volume 4203 of *Lecture Notes in Computer Science*, pages 322–331. Springer, Berlin–Heidelberg, Germany.

Fanizzi, N. and d'Amato, C. (2007). Inductive concept retrieval and query answering with semantic knowledge bases through kernel methods. In Apolloni, B., Howlett, R. J., and Jain, L. C., editors, *Proceedings of the 11th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2007), XVII Italian Workshop on Neural Networks,*, number 4692 in Lecture Notes in Computer Science, pages 148–155. Springer, Berlin–Heidelberg, Germany.

Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery: an overview. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in knowledge discovery and data mining*, pages 1–34. AAAI Press, Menlo Park, CA, USA.

Feldman, R. and Dagan, I. (1995). Knowledge discovery in textual databases. In Fayyad, U. M. and Uthurusamy, R., editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), August 20-21, 1995, Montreal, Canada*, pages 112–117. AAAI Press, Chicago, IL, USA.

Feldman, R. and Sanger, J. (2006). *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, Cambridge, MA, USA.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

Frasconi, P., Passerini, A., Muggleton, S., and Lodhi, H. (2004). Declarative kernels. Technical Report RT 2/2004, Dipartimento di Sistemi e Informatica, Universit'a di Firenze, Florence, Italy.

Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. (1987). The vocabulary problem in human-system communication. *Communumications of the ACM*, 30(11):964–971.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58.

Giuglea, A.-M. and Moschitti, A. (2006). Semantic role labeling via FrameNet, VerbNet and PropBank. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL 2006). July 17-21, 2006, Sydney, Australia*. Association for Computational Linguistics, Morristown, NJ, USA.

Gomez-Perez, A. and Corcho, O. (2001). Ontology languages for the semantic web. *IEEE Intelligent Systems*, 17(1):54–60.

Gosset, W. S. (1908). The probable error of a mean. *Biometrika*, 6(1):1–25. Originally published under the pseudonym "Student".

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871.

Gower, J. C. and Legendre, P. (1986). Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(1):5–48.

Green, S. (1999). Building hypertext links by computing semantic similarity. *IEEE Transactions on Knowledge and Data Engineering*, 11:713–730.

Grimm, S., Hitzler, P., and Abecker, A. (2007). Knowledge representation and ontologies. In Studer, R., Grimm, S., and Abecker, A., editors, *Semantic Web Services: Concepts, Technology and Applications*, pages 51–106. Springer, Berlin–Heidelberg, Germany.

Gärtner, T. (2003). A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58.

Gärtner, T. (2005). *Kernels for Structured Data*. PhD thesis, University of Bonn, Germany.

Gärtner, T., Flach, P., and Wrobel, S. (2003a). On graph kernels: Hardness results and efficient alternatives. In Schölkopf, B. and Warmuth, M. K., editors, *Computational Learning Theory and Kernel Machines — Proceedings of the 16th Annual Conference on*

Computational Learning Theory and 7th Kernel Workshop (COLT/Kernel 2003) August 24-27, 2003, Washington, DC, USA, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143. Springer, Berlin–Heidelberg, Germany.

Gärtner, T., Lloyd, J., and Flach, P. (2003b). Kernels for structured data. In Matwin, S. and Sammut, C., editors, *Proceedings of the 12th International Conference on Inductive Logic Programming (ILP 2002), July 9 - 11, 2002, Sydney, Australia — Revised Papers*, number 2583 in Lecture Notes in Computer Science, pages 66–83. Springer, Berlin–Heidelberg, Germany.

Gruber, T. R. (1993). Towards principles for the design of ontologies used for knowledge sharing. In Guarino, N. and Poli, R., editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, Deventer, The Netherlands.

Guarino, N. (1998). Formal ontology and information systems. In Guarino, N., editor, *Proceedings of the First International Conference on Formal Ontologies in Information Systems (FOIS-98), June 6-8, 1998, Trento, Italy*, pages 3–15. IOS Press, Amsterdam, The Netherlands.

Haasdonk, B. (2005). Feature space interpretation of svms with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492.

Haasdonk, B. and Bahlmann, C. (2004). Learning with distance substitution kernels. In Rasmussen, C. E., Bülthoff, H. H., Schölkopf, B., and Giese, M. A., editors, *Pattern Recognition, 26th DAGM Symposium, August 30 - September 1, 2004, Tübingen, Germany, Proceedings*, volume 3175 of *Lecture Notes in Computer Science*, pages 220–227. Springer.

Harris, Z. (1968). *Mathematical Structures of Language*. Wiley, New York, NY, USA.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA.

Hayes, P. (2004). RDF semantics. W3C recommendation, W3C. Published online on February 10th, 2004 at `http://www.w3.org/TR/2003/PR-rdf-mt-20031215/`.

Hayes, P. J. and Weinstein, S. P. (1991). CONSTRUE/TIS: A system for content-based indexing of a database of news stories. In Rappaport, A. T. and Smith, R. G., editors, *Proceedings of the 2nd Conference on Innovative Applications of Artificial Intelligence (IAAI-90), May 1-3, 1990, Washington, DC, USA*, pages 49–64. AAAI Press, Chicago, IL, USA.

Hefke, M., Zacharias, V., Abecker, A., Wang, Q., Biesalski, E., and Breiter, M. (2006). An extendable java framework for instance similarities in ontologies. In Manolopoulos, Y., Filipe, J., Constantopoulos, P., and Cordeiro, J., editors, *Proceedings of the 8th International Conference on Enterprise Information Systems: Databases and Information Systems Integration (ICEIS 2006), May 23-27, 2006, Paphos, Cyprus*, pages 263–269. INSTICC, Setúbal, Portugal.

Hitzler, P., Krötzsch, M., Rudolph, S., and Sure, Y. (2008). *Semantic Web — Grundlagen*. eXamen.press. Springer, Berlin–Heidelberg, Germany. In German.

Hitzler, P. and Vrandecic, D. (2005). Resolution-based approximate reasoning for OWL DL. In Gil, Y., Motta, E., Benjamins, V. R., and Musen, M. A., editors, *Proceedings of the 4th International Semantic Web Conference (ISWC 2005), November 6-10, 2005, Galway, Ireland*, volume 3729 of *Lecture Notes in Computer Science*, pages 383–397. Springer, Berlin–Heidelberg, Germany.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In Gey, F., Hearst, M., and Tong, R., editors, *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99), August 15-19, 1999, Berkeley, CA, USA*, pages 50–57. ACM Press, New York, NY, USA.

Hofmann, T., Scholkopf, B., and Smola, A. J. (2007). Kernel methods in machine learning. Published online at arXiv.org `http://arxiv.org/abs/math/0701907v2`. To appear in Annals of Statistics, 2008.

Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2003). From SHIQ and RDF to OWL: The Making of a Web Ontology Language . *Journal of Web Semantics*, 1(1).

Horváth, T., Gärtner, T., and Wrobel, S. (2004). Cyclic pattern kernels for predictive graph mining. In Kim, W., Kohavi, R., Gehrke, J., and DuMouchel, W., editors, *Proceedings of the 10t ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004), August 22-25, 2004, Seattle, WA, USA*, pages 158–167. ACM Press, New York, NY, USA.

Horváth, T., Wrobel, S., and Bohnebeck, U. (2001). Relational instance-based learning with lists and terms. *Machine Learning*, 43(1):53–80.

Hotho, A. (2004). *Clustern mit Hintergrundwissen*. Number 286 in Dissertationen zur Künstlichen Intelligenz. Akademische Verlagsgesellschaft, Berlin, Germany. In German. Originally published as PhD Thesis, 2004, Universität Karlsruhe (TH), Karlsruhe, Germany.

Hotho, A., Nürnberger, A., and Paaß, G. (2005). A brief survey of text mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 20(1):19–62.

Hotho, A., Staab, S., and Stumme, G. (2003). Wordnet improves text document clustering. In Ding, Y., van Rijsbergen, K., Ounis, I., and Jose, J., editors, *Proceedings of the Semantic Web Workshop of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval (SIGIR 2003), August 1, 2003, Toronto Canada*. Published Online at `http://de.scientificcommons.org/608322`.

Hovy, E., Gerber, L., Hermjakob, U., Lin, C.-Y., and Ravichandran, D. (2001). Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT '01), March 18 - 21, 2001, San Diego, CA, USA*, pages 1–7. Association for Computational Linguistics, Morristown, NJ, USA.

Ide, N. and Véronis, J. (1998). Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, 24(1):1–40.

Ifrim, G. and Weikum, G. (2006). Transductive learning for text classification using explicit knowledge models. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Knowledge Discovery in Databases — Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006), September 18-22, 2006, Berlin, Germany*, volume 4213 of *Lecture Notes in Computer Science*, pages 223–234. Springer, Berlin–Heidelberg, Germany.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

Jing, Y. and Croft, W. (1994). An association thesaurus for information retrieval. In *Proceedings of the 4th International Conference on Computer Assisted Information Retrieval – Recherche d'Informations Assistee par Ordinateur (RIAO 94)*, pages 146–160.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In Nédellec, C. and Rouveirol, C., editors, *Proceedings of the 10th European Conference on Machine Learning (ECML 1998), April 21-23, 1998, Chemnitz, Germany*, pages 137–142. Springer, Berlin–Heidelberg, Germany.

Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA.

Joachims, T. (2000). Estimating the generalization performance of an SVM efficiently. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000), June 29-July 2, 2000, Stanford, CA, USA*, pages 431–438. Morgan-Kaufman Publishers, San Francisco, CA, USA.

Joachims, T. (2001). A statistical learning model of text classification for support vector machines. In Croft, W. B., Harper, D. J., Kraft, D. H., and Zobel, J., editors, *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001), September 9-13, 2001, New Orleans, LA, USA*, pages 128–136. ACM Press, New York, NY, USA.

Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines — Methods, Theory, and Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA. Originally published as PhD Thesis, 2001, Technische Universität Dortmund, Germany.

Joachims, T. (2005). A support vector method for multivariate performance measures. In Raedt, L. D. and Wrobel, S., editors, *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), August 7-11, 2005, Bonn, Germany*, pages 377–384. ACM Press, New York, NY, USA.

Karush, W. (1939). Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, Chicago, IL, USA.

Kashima, H., Tsuda, K., and Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In Fawcett, T. and Mishra, N., editors, *Proceedings of the 20th International Conference on Machine Learning (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 321–328. AAAI Press, Chicago, IL, USA.

Kiefer, C., Bernstein, A., Lee, H. J., Klein, M., and Stocker, M. (2007a). Semantic process retrieval with iSPARQL. In Franconi, E., Kifer, M., and May, W., editors, *The Semantic Web: Research and Applications — Proceedings of the 4th European Semantic Web Conference (ESWC 2007), June 3-7, 2007, Innsbruck, Austria*, volume 4519 of *Lecture Notes in Computer Science*, pages 609–623. Springer, Berlin–Heidelberg, Germany.

Kiefer, C., Bernstein, A., and Locher, A. (2008). Adding data mining support to SPARQL via statistical relational learning methods. In Bechhofer, S., Hauswirth, M., Hoffmann, J., and Koubarakis, M., editors, *The Semantic Web: Research and Applications — Proceedings of the 5th European Semantic Web Conference (ESWC 2008), June 1-5, 2008, Tenerife, Spain*, volume 5021 of *Lecture Notes in Computer Science*, pages 478–492. Springer, Berlin–Heidelberg, Germany.

Kiefer, C., Bernstein, A., and Stocker, M. (2007b). The fundamentals of iSPARQL — a virtual triple approach for similarity-based semantic web tasks. In Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web — Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC 2007 + ASWC 2007), November 11-15, 2007, Busan, Korea*, volume 4825 of *Lecture Notes in Computer Science*, pages 295–308. Springer, Berlin–Heidelberg, Germany.

Kimeldorf, G. S. and Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95.

Klyne, G. and Carroll, J. J. (2004). Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation, W3C. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

Knuth, D. E. (1997). *Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley, Reading, MA, USA, 3rd edition.

Kolmogorov, A. N. and Fomin, S. V. (1970). *Introductory Real Analysis*. Prentice-Hall, Englewood Cliffs, NJ, USA, revised English edition edition.

Kondor, R. I. and Lafferty, J. D. (2002). Diffusion kernels on graphs and other discrete input spaces. In Sammut, C. and Hoffmann, A. G., editors, *Proceedings of the 19th International Conference on Machine Learning (ICML 2002), July 8-12, 2002, Sydney, Australia*, pages 315–322. Morgan-Kaufman Publishers, San Francisco, CA, USA.

Kramer, S., Lavrac, N., and Flach, P. (2001). Propositionalization approaches to relational data mining. In Dzeroski, S. and Lavrac, N., editors, *Relational Data Mining*, pages 262–286. Springer New York Inc., New York, NY, USA.

Krumhansl, C. L. (1978). Concerning the applicability of geometric models to similarity data: The interrelationship between similarity and spatial density. *Psychological Review*, 85(5):445–463.

Kucera, H. and Francis, W. N. (1967). *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI, USA.

Kuhn, H. W. and Tucker, A. W. (1950). Nonlinear programming. In Neyman, J., editor, *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, Berkeley, CA, USA.

Lavrač, N. and Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, NY, USA.

Leopold, E. and Kindermann, J. (2002). Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46(1-3):423–444.

Lewis, D. D. (1991). Evaluating text categorization. In *Proceedings of Speech and Natural Language Workshop, Feb 1991, Asilomar, CA, USA*, pages 312–318. Morgan-Kaufman Publishers, San Francisco, CA, USA.

Li, X. and Roth, D. (2002). Learning question classifiers. In Chen, T.-E. and Liu, Y.-F., editors, *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002), August 24 - September 1, 2002, Taipei, Taiwan*, pages 556–562. Morgan-Kaufman Publishers, San Francisco, CA, USA.

Li, X. and Roth, D. (2006). Learning question classifiers: The role of semantic information. *Natural Language Engineering*, 12(3):229–249.

Li, Y., Bandar, Z. A., and McLean, D. (2003). An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882.

Lin, D. (1998). An information-theoretic definition of similarity. In Shavlik, J. W., editor, *Proceedings of the 15th International Conference on Machine Learning (ICML 1998), July 24-27, 1998, Madison, WI, USA*, pages 296–304. Morgan-Kaufman Publishers, San Francisco, CA, USA.

Lloyd, J. (2003). *Logic for Learning: Learning Comprehensible Theories from Structured Data*. Springer, Berlin–Heidelberg, Germany.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Lu, Q. and Getoor, L. (2003). Link-based classification. In Fawcett, T. and Mishra, N., editors, *Proceedings of the 20th International Conference on Machine Learning (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 496–503. AAAI Press, Chicago, IL, USA.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Le Cam, L. M. and Neyman, J., editors, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability - Vol. 1*, pages 281–297. University of California Press, Berkeley, CA, USA.

Maedche, A. (2002). *Ontology Learning for the Semantic Web*. Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, Norwell, MA, USA. Originally published as PhD Thesis, 2001, Universität Karlsruhe (TH), Karlsruhe, Germany.

Maedche, A. and Staab, S. (2001). Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79.

Maedche, A. and Staab, S. (2002). Measuring similarity between ontologies. In Gómez-Pérez, A. and Benjamins, V. R., editors, *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the*

Semantic Web (EKAW 2002), October 1-4, 2002, Siguenza, Spain, volume 2473 of *Lecture Notes in Computer Science*, pages 251–263. Springer, Berlin–Heidelberg, Germany.

Maedche, A. and Zacharias, V. (2002). Clustering ontology-based metadata in the semantic web. In Elomaa, T., Mannila, H., and Toivonen, H., editors, *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2002), August 19-23, 2002, Helsinki, Finland*, volume 2431 of *Lecture Notes in Computer Science*, pages 383–408. Springer, Berlin–Heidelberg, Germany.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

Manola, F. and Miller, E. (2004). RDF primer. W3C recommendation, W3C. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`.

Marwick, A. D. (2001). Knowledge management technology. *IBM Systems Journal*, 40(4):814–830.

Mavroeidis, D., Tsatsaronis, G., and Vazirgiannis, M. (2005a). Semantic distances for sets of senses and applications in word sense disambiguation. In Sirmakessis, S., editor, *Knowledge Mining — Proceedings of NEMIS 2004 Final Conference, 3rd International Workshop in Text Mining and its Applications, October 25, 2004, Athens, Greece*, volume 185 of *Studies in Fuzziness and Soft Computing*, pages 93–108. Springer, Berlin–Heidelberg, Germany.

Mavroeidis, D., Tsatsaronis, G., Vazirgiannis, M., Theobald, M., and Weikum, G. (2005b). Word sense disambiguation for exploiting hierarchical thesauri in text classification. In Jorge, A., Torgo, L., Brazdil, P., Camacho, R., and Gama, J., editors, *Knowledge Discovery in Databases — Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2005), October 3-7, 2005, Porto, Portugal*, volume 3721 of *Lecture Notes in Computer Science*, pages 181–192. Springer, Berlin–Heidelberg, Germany.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

McGuinness, D. L. and van Harmelen, F. (2004). OWL web ontology language overview. W3C recommendation, W3C. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.

McHale, M. (1998). A comparison of wordnet and roget's taxonomy for measuring semantic similarity. In Harabagiu, S. and Chai, J. Y., editors, *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems, August 16, 1998, Montreal, Canada*. Association for Computational Linguistics, Morristown, NJ, USA.

Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, 209:415–446.

Metzler, D. and Croft, W. B. (2005). Analysis of statistical question classification for fact-based questions. *Information Retrieval*, 8(3):481–504.

Mierswa, I. (2006). Evolutionary learning with kernels: A generic solution for large margin problems. In Cattolico, M., editor, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO '06), July 8-12, 2006, Seattle, Washington, USA*, pages 1553–1560. ACM Press, New York, NY, USA.

Miles, A. and Brickley, D. (2005). SKOS core guide. W3C working draft, W3C. Published online on November 2nd, 2005 at `http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102/`.

Miller, G. A. (1995). WordNet: a Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Miller, G. A., Fellbaum, C., Gross, D., and Miller, K. J. (1990). Introduction to WordNet: an On-Line Lexical Database. *International Journal of Lexicography*, 3(4):235–244.

Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, New York, NY, USA.

Morik, K., Brockhausen, P., and Joachims, T. (1999). Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In Bratko, I. and Dzeroski, S., editors, *Proceedings of the 16th International Conference on Machine Learning (ICML 1999), June 27-30, 1999, Bled, Slovenia*, pages 268–277. Morgan-Kaufman Publishers, San Francisco, CA, USA.

Moschitti, A. (2004). A study on convolution kernels for shallow statistic parsing. In Scott, D., editor, *Proceedings of the 42th Annual Meeting on Association for Computational Linguistics (ACL 2004), July 21-26, 2004, Barcelona, Spain*, pages 335–342. Association for Computational Linguistics, Morristown, NJ.

Moschitti, A. (2006). Efficient convolution kernels for dependency and constituent syntactic trees. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Proceedings of the 17th European Conference on Machine Learning (ECML 2006), September 18-22, 2006, Berlin, Germany*, volume 4212 of *Lecture Notes in Computer Science*, pages 318–329. Springer, Berlin–Heidelberg, Germany.

Moschitti, A. and Basili, R. (2004). Complex linguistic features for text classification: A comprehensive study. In McDonald, S. and Tait, J., editors, *Advances in Information Retrieval — Proceedings of the 26th European Conference on Informatino Retrieval (ECIR 2004), April 5-7, 2004, Sunderland, UK*, volume 2997 of *Lecture Notes in Computer Science*, pages 181–196. Springer, Berlin–Heidelberg, Germany.

Motik, B. (2006). *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe (TH), Karlsruhe, Germany.

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing*, 8(4):295–318.

Muggleton, S. and Raedt, L. D. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679.

Neville, J., Jensen, D., and Gallagher, B. (2003). Simple estimators for relational bayesian classifiers. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), December 19-22, 2003, Melbourne, Florida, USA*, pages 609–612. IEEE Computer Society, Washington, DC, USA.

Novikoff, A. B. (1962). On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, New York, NY, USA. Polytechnic Institute of Brooklyn.

Ogden, C. K. and Richards, I. (1923). *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism*. Routledge & Kegan Paul Ltd., London, UK, 10th edition.

Ong, C. S., Mary, X., Canu, S., and Smola, A. J. (2004). Learning with non-positive kernels. In Brodley, C., editor, *Proceedings of the 21st International Conference on Machine Learning (ICML 2004), July 4-8, 2004, Banff, Alberta, Canada*, page 81. ACM Press, New York, NY, USA.

Park, Y. and Choi, K.-S. (1996). Automatic thesaurus construction using bayesian networks. *Information Processing Management*, 32(5):543–553.

Passerini, A., Frasconi, P., and De Raedt, L. (2006). Kernels on prolog proof trees: Statistical learning in the ILP setting. *Journal of Machine Learning Research*, 7:307–342.

Patel-Schneider, P. F., Hayes, P., and Horrocks, I. (2004). OWL web ontology language semantics and abstract syntax. W3C recommendation, W3C. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-owl-semantics-20040210/`.

Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program*, 14(3):130–137.

Prud'hommeaux, E. and Seaborne, A. (2007). SPARQL query language for RDF. W3C working draft, W3C. Published online on March 26th, 2007 at `http://www.w3.org/TR/2007/WD-rdf-sparql-query-20070326/`.

Quarteroni, S., Moschitti, A., Manandhar, S., and Basili, R. (2007). Advanced structural representations for question classification and answer re-ranking. In Amati, G., Carpineto, C., and Romano, G., editors, *Advances in Information Retrieval — Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007), 2-5 April 2007, Rome, Italy*, volume 4425 of *Lecture Notes in Computer Science*, pages 234–245. Springer, Berlin–Heidelberg, Germany.

Rada, R. and Bicknell, E. (1989). Ranking documents with a thesaurus. *JASIS*, 40(5):304–310.

Ramakrishnan, C., Milnor, W. H., Perry, M., and Sheth, A. P. (2005). Discovering informative connection subgraphs in multi-relational graphs. *ACM SIGKDD Explorations Newsletter*, 7(2):56–63.

Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130.

Rissland, E. L. (2006). AI and similarity. *IEEE Intelligent Systems*, 21(3):39–49.

Rosch, E., Mervis, C. B., Gray, W. D., Johnson, D. M., and Boyes-Braem, P. (1976). Basic objects in natural categories. *Cognitive Psychology*, 8(3):382–439.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, USA, second edition.

Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley Publishing Inc, Boston, MA, USA.

Salton, G. and Lesk, M. E. (1971). Computer evaluation of indexing and text processing. In Salton, G., editor, *The SMART Retrieval System – Experiments in Automatic Document Processing.*, pages 143–180. Prentice-Hall, Englewood Cliffs, NJ, USA.

Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, USA.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Sanderson, M. (1994). Word sense disambiguation and information retrieval. In Croft, W. B. and van Rijsbergen, C. J., editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94), July 3-6, 1994, Dublin, Ireland*, pages 142–151. Springer New York Inc., NY, USA.

Santorini, B. (1990). Part-Of-Speech tagging guidelines for the Penn Treebank project (3rd revision, 2nd printing). Technical report, Department of Linguistics, University of Pennsylvania, Philadelphia, PA, USA.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels*. MIT Press, Cambridge, MA, USA.

Schölkopf, B., Smola, A., and Müller, K.-R. (1996). Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany.

Schölkopf, B., Smola, A. J., Williamson, R. C., and Bartlett, P. L. (2000). New support vector algorithms. *Neural Computation*, 12(5):1207–1245.

Schmid, H. (2000). LoPar: Design and implementation. Bericht des Sonderforschungsbereiches "Sprachtheoretische Grundlagen für die Computerlinguistik" 149, Institute for Computational Linguistics, University of Stuttgart, Stuttgart, Germany.

Schütze, H. and Pedersen, J. (1997). A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management*, 33(3):307–318.

Scott, S. and Matwin, S. (1999). Feature engineering for text classification. In Bratko, I. and Dzeroski, S., editors, *Proceedings of the 16th International Conference on Machine Learning (ICML 1999), June 27-30, 1999, Bled, Slovenia*, pages 379–388. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Shadbolt, N., Berners-Lee, T., and Hall, W. (2006). The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK.

Siolas, G. and d'Alche Buc, F. (2000). Support vector machines based on a semantic kernel for text categorization. In *IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, volume 5, pages 205–209. IEEE Computer Society, Washington, DC, USA.

Staab, S. and Studer, R., editors (2004). *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, Berlin–Heidelberg, Germany.

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197.

Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., and Oberle, D. (2005). The SWRC ontology - Semantic Web for research communities. In Bento, C., Cardoso, A., and Dias, G., editors, *Progress in Artificial Intelligence — Proceedings of the 12th Portuguese Conference on Artificial Intelligence (EPIA 2005), December 5-8, 2005, Covilhã, Portugal*, volume 3803 of *Lecture Notes in Computer Science*, pages 218–231. Springer, Berlin–Heidelberg, Germany.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In Brodley, C. E., editor, *Proceedings of the 21st International Conference on Machine Learning (ICML 2004), July 4-8, 2004, Banff, Alberta, Canada*, Banff, Canada. ACM Press, New York, NY, USA. Paper No. 104.

Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327–352.

Udrea, O., Getoor, L., and Miller, R. (2007). Leveraging data and structure in ontology integration. In Zhou, L., Ling, T. W., and Ooi, B. C., editors, *Proceedings of ACM-SIGMOD 2007 International Conference on Management of Data, June 11-14, 2007, Beijing, China*, pages 449–460. ACM Press, New York, NY, USA.

Ureña, L. A., de Buenaga, M., and Gómez, J. M. (2001). Integrating linguistic resources in TC through WSD. *Computers and the Humanities*, 35(2):215–230.

van Assem, M., Gangemi, A., and Schreiber, G. (2006). RDF/OWL representation of wordnet. Editor's draft, W3C semantic web best practices and deployment working group, W3C. Published online on April 26th, 2006 at `http://www.w3.org/2001/sw/BestPractices/WNET/wn-conversion-20062304/`.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths, London, 2 edition.

Vapnik, V., Golowich, S. E., and Smola, A. J. (1997). Support vector method for function approximation, regression estimation and signal processing. In Mozer, M., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9 — Proceedings of the 1996 Neural Information Processing Systems Conference (NIPS 1996), December 2-5, 1996, Dever, CO, USA*, pages 281–287. MIT Press, Cambridge, MA, USA.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer New York Inc., New York, NY, USA.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley, New York, NY, USA.

Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280.

Vapnik, V. N. and Chervonenkis, A. Y. (1974). *Theory of Pattern Recognition [in Russian]*. Nauka, USSR.

Vishwanathan, S. V. N. and Smola, A. J. (2003). Fast kernels for string and tree matching. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15 — Proceedings of the 2002 Neural Information Processing Systems Conference (NIPS 2002), December 9-14, 2002, Vancouver, British Columbia, Canada*, pages 569–576. MIT Press, Cambridge, MA, USA.

Vishwanathan, S. V. N. and Smola, A. J. (2004). Fast kernels for string and tree matching. In Tsuda, K., Schölkopf, B., and Vert, J., editors, *Kernels and Bioinformatics*. MIT Press, Cambridge, MA, USA.

Voorhees, E. (1994). Query expansion using lexical-semantic relations. In Croft, W. B. and van Rijsbergen, C. J., editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94), July 3-6, 1994, Dublin, Ireland*, pages 61–69. Springer New York Inc., NY, USA.

Wang, B. B., Mckay, R. I., Abbass, H. A., and Barlow, M. (2003). A comparative study for domain ontology guided feature extraction. In Oudshoorn, M. J., editor, *Proceedings of the 26th Australian Computer Science Conference (ACSC-2003) , Adelaide, Australia - Volume 16*, pages 69–78. Australian Computer Society, Darlinghurst, Australia.

Wang, Y.-C., Vandendorpe, J., and Evens, M. (1985). Relational thesauri in information retrieval. *Journal of the American Society for Information Science*, 36(1):15–27.

Widdows, D. (2004). *Geometry and Meaning*. CLSI Publications, Stanford, CA, USA.

Wu, Z. and Palmer, M. S. (1994). Verb semantics and lexical selection. In Pustejovsky, J., editor, *Proceedings of the 32th Annual Meeting on Association for Computational Linguistics (ACL '94), June 27-30, 1994, New Mexico State University, Las Cruces, New Mexico, USA*, pages 133–138. Morgan-Kaufman Publishers, San Francisco, CA, USA.

Yang, Y. (1999). An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, 1(1-2):69–90.

Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In Gey, F., Hearst, M., and Tong, R., editors, *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99), August 15-19, 1999, Berkeley, CA, USA*, pages 42–49. ACM Press, New York, NY, USA.

Zegers, F. (1986). *A General Family of Association Coefficients*. PhD thesis, Rijksuniversiteit te Groningen, Groningen, The Netherlands.

Zelenko, D., Aone, C., and Richardella, A. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

Zhang, D. and Lee, W. S. (2003). Question classification using Support Vector Machines. In Callan, J., Clarke, C., Cormack, G., Hawking, D., and Smeaton, A., editors, *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03), July 28 - August 1, 2003, Toronto, Canada*, pages 26–32. ACM Press, New York, NY, USA.

Zhang, M., Zhou, G., and Aw, A. (2008). Exploring syntactic structured features over parse trees for relation extraction using kernel methods. *Information Processing and Management*, 44(2):687–701.