Ontology Matching by Combining Instance-Based Concept Similarity Measures with Structure

Konstantin Todorov

Submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy in Cognitive Science (FB 8)
at the
University of Osnabrück

October 2009

Advisors:

Prof. Dr. Kai-Uwe Kühnberger (University of Osnabrück)

PD Dr. Peter Geibel (Technical University Berlin)

External Reviewer:

Prof. Dr. Alexander Mehler (University of Bielefeld)

Contents

List of Figures

List of Tables

Acknowledgments

During the years of work on this thesis, I was lucky to have been constantly receiving support from many people in different forms. All the persons and institutions mentioned below have helped me directly by providing me with the scientific advice, organizational expertize and emotional comfort that I have needed.

First and foremost, I would like to thank my supervisors, Kai-Uwe Kühberger and Peter Geibel. I am much obliged to Kai-Uwe Kühberger for introducing me to the exciting area of ontology matching four years ago and for continuously supporting my endeavors in that field, scientifically, organizationally, technically and, last but not least, financially. I am indebted to Peter Geibel for his valuable presence, for his critical comments throughout many discussions, for his moral support and constant encouragement, which helped me gain confidence in my ideas.

I would like to thank all my friends and colleagues from the Doctorate Programme of the Institute of Cognitive Science (IKW) who have contributed to my thesis with discussions and comments, but who have also made my stay in Osnabrück a great time to remember. In that respect, I would like to mention the names of Tonio Wandmacher, Ulas Türkmen, Sasha Alexejenko, Ilaria Serafini, Katya Ovchinnikova, Daniel Weiller, Miriam Kyselo and Mikko Määttä. I thank particularly Markus Eronen, for the music and the pipes.

My gratitude equally goes to Carla Umbach and Peter Bosch. I thank them for the organizational help, for the valuable comments on my work during my presentations at the doctorate colloquium and for supporting financially, through the Doctorate Programme which they are in charge of, my conference and scientific trips. Talking about the financial side, I would like to mention the support of the German Academic Exchange Service who accorded me a one year scholarship at a crucial moment of my work. I am as well thankful to Helmar Gust who offered me a *"HiWi"* position at the IKW.

I thank Alexander Mehler who agreed to write an external review of my thesis.

Many thanks to Nicolas Cantono for helping me with part of the experimental work.

Finally, I would like to express my gratitude and indebtedness to my parents Margarita and Georgi and my brother Ivan for their continued and unconditional support, which I was always able to count on, not only during the last four years.

A special thank you to Tsveta, for all the reasons that I hope she knows.

K. G. T., October 2009, Osnabrück

Abstract

Ontologies describe the semantics of data and provide a uniform framework of understanding between different parties. The main common reference to an ontology definition describes them as knowledge bodies, which bring a formal representation of a shared conceptualization of a domain – the objects, concepts and other entities that are assumed to exist in a certain area of interest together with the relationships holding among them. However, in open and evolving systems with decentralized nature (as, for example, the Semantic Web), it is unlikely for different parties to adopt the same ontology. The problem of ontology matching evolves from the need to align ontologies, which cover the same or similar domains of knowledge. The task is to reducing ontology heterogeneity, which can occur in different forms, not in isolation from one another. *Syntactically* heterogeneous ontologies are expressed in different formal languages. *Terminological* heterogeneity stands for variations in names when referring to the same entities and concepts. *Conceptual* heterogeneity refers to differences in *coverage*, *granularity* or *scope* when modeling the same domain of interest. Finally, *prgamatic* heterogeneity is about mismatches in how entities are interpreted by people in a given context.

The work presented in this thesis is a contribution to the problem of reducing the terminological and conceptual heterogeneity of hierarchical ontologies (defined as ontologies, which contain a hierarchical body), populated with text documents. We make use of both intensional (structural) and extensional (instance-based) aspects of the input ontologies and combine them in order to establish correspondences between their elements. In addition, the proposed procedures yield assertions on the granularity and the extensional richness of one ontology compared to another, which is helpful at assisting a process of ontology merging. Although we put an emphasis on the application of instance-based techniques, we show that combining them with intensional approaches leads to more efficient (both conceptually and computationally) similarity judgments.

The thesis is oriented towards both researchers and practitioners in the domain of ontology matching and knowledge sharing. The proposed solutions can be applied successfully to the problem of matching web-directories and facilitating the exchange of knowledge on the web-scale.

Chapter 1

Introduction

Ontology matching is an active interdisciplinary research field, which encompasses subfields of computer science and artificial intelligence, computational linguistics, psychology and philosophy. This chapter has as a goal to introduce the reader to the main topics discussed throughout the thesis, all grounded at or motivating the ontology matching research, taking a generic stance. We start by discussing motivational issues and illustrating the problem of ontology matching with a small example (Section 1.1). Further, we present various methodological frameworks for building solutions to this problem (Section 1.2). We proceed to state our main hypothesis and a set of originating research questions (Section 1.3) before we outline the major contribution and results of this work (Section 1.4). Finally, we describe the structure of the thesis and suggest an intended audience (Section 1.5).

1.1 Aims and Motivation

The relation between the *objects in the world*, their *mental representations (concepts)* and the *words or language expressions (symbols)* that stand for both is fairly complex. These three entities form what Ogden and Richards have called the *meaning triangle* [117]. The connection and interaction between any two vertices of that triangle give origin to interdisciplinary research involving fields of linguistics, cognitive psychology, philosophy and artificial intelligence (AI). In an attempt to prepare the reader for stating the basic research questions, hypotheses and results of this thesis, we will sketch a possible view on how these entities are related.

By objects in the world we understand a whole collection of things, locations, times, events, states, relations, features, etc. People develop knowledge about the categorical nature of these components through their lives, as argued by Barsalou [8]. One way of defining the relation between concepts and real-world objects is to identify concepts with the categories developed by humans, structuring the objects in the world and shaping their knowledge about them. This conceptualization is not universal and does not hold for all people in the world altogether and in the same time. It is distributed and dispersed among smaller communities, among sub-groups of these communities, or even sometimes among particular individuals. What distinguishes, for example, different human cultures is, to a big degree, the shared agreement about categories, which exists among the members of these cultural groups and the interpretation that they give to these categories [56].

The relationship that holds between linguistic symbols and concepts, on one hand, and linguistic symbols and objects in the world, on the other, is not simpler. The names that people give to the things in the world and the categories that they form differ among communities and individuals, as well. Naturally, this is a common phenomenon among multi-linguistic communities: even if a concept is understood equally by people in France and people in Germany (for example, the concept corresponding to the thing known under the English word *wine*), the linguistic symbols used to refer to this concept and its instances differ across these communities (for example, the French *vin* against the German *Wein*). But even in communities, which agree on the use of a common language this ambiguity can occur, among sub-groups of these communities, due to language-related factors, such as synonymy or polysemy[1], or to community-specific factors, such as importance of a concept for that community, scope, terminology, specificity.

To sum up, we point to the observation of [111] that the nature of the relation between words, concepts and real world entities can be accounted for by relating the *meaning of a word* to a *mental description* (conceptual system), which allows people to exemplify words by *objects in the world*. We emphasize that we would expect the mental representation (concept) that characterizes and includes sets of real world objects and the choice of symbols (words, expressions) that stand for them to differ among different people due to many possible ambiguities stemming from the distributed nature of human concept formation and the set of complex psychological principles, which underlie this process [130].

We will close this section by a small example. We all know from biology classes how the different animal species are organized in a taxonomy. They are divided into groups according to morphological, physiological, evolutionary and other criteria (vertebrates vs. invertebrates; mammals, fish, amphibians, birds, etc.) and the different groups are ordered in a tree rooted in the most general class branching down to the individual species and subspecies.

Below is a taxonomy of the animal kingdom, taken from the Chinese encyclopedia *The Celestial Emporium of Benevolent Knowledge* (as cited by J. L. Borges [16]), according to which

> [...] animals are divided into (a) those that belong to the Emperor, (b) embalmed ones, (c) those that are trained, (d) suckling pigs, (e) mermaids, (f) fabulous ones, (g) stray dogs, (h) those that are included in this classification, (i) those that tremble as if they were mad, (j) innumerable ones, (k) those drawn with a very fine camel's hair brush, (1) others, (m) those that have just broken a flower vase, (n) those that resemble flies from a distance.

Evidently, the latter classification differs significantly from the taxonomies found in biology textbooks, in terms of categorization criteria, granularity and scope, even though it also intends to provide a structural account of what kinds of animals there are in the world.

[1]Synonyms are different words, which label the same entity or concept; polysemy is the phenomenon of a word having multiple meanings.

Ontology Matching

In the paragraphs above we have attempted to shape an understanding about the phenomenon of human category formation, which, in our view, underlies the problems studied in this thesis. We will introduce and motivate our research efforts in the sequel.

An ontology in Artificial Intelligence[2] is most generally viewed as a collection of entities, called concepts, objects in the world (instances) and names, together with a set of relations defined on these entities. Usually, the concepts and the relations that hold among them are referred to as the *intension* of the ontology, whereas the objects in the world are referred to as the *extension* of the ontology.

Despite the difficulties faced in agreeing on a common definition of an ontology, many authors refer to Gruber's 1993 paper as the source of the most general available specification of what an ontology is. There, an ontology is defined as

> a formal representation of a shared conceptualization of a domain – the objects, concepts and other entities that are assumed to exist in a certain area of interest together with the relationships holding among them [61].

Ontologies were introduced to provide semantics to data, to structure the knowledge about things in various domains of life, to provide a uniform framework for understanding between different parties within a single community or across communities. Applications of ontologies can be found in many fields of life and science, such as knowledge sharing and re-use, automatic reasoning and problem solving, natural language processing, multimedia search and retrieval, e-business, to name a few [143]. Ontologies are basic components of the Semantic Web – a vision of the future web where data will be made understandable for both humans *and* computers and ontologies will be the linking element, which will provide the necessary semantics for machines to "understand" information, which on the current web is only understandable by humans [12].

The problem of ontology matching stems from the fact that the machine-mediated and oriented process of ontology generation, acquisition, development, re-use and updating is decentralized and strongly human biased, analogous to the natural process of category formation, discussed above. As a result, many ontologies will be created, covering in a non-explicit manner identical or partially overlapping domains of knowledge. For reasons, part of which are explained in the preceding section, it is unlikely to expect that humans, independently, would create ontologies perfectly corresponding to one another with respect to their structure and granularity, linguistic use and scope, even if these ontologies are intended to describe the knowledge about one common domain. We will call this phenomena *ontology heterogeneity* and ontologies, which have been created to describe the same or similar fields, but whose elements do not explicitly match, will be called *heterogeneous* (following [43] and other authors).

Often in practice the communication between ontologies is needed, in order to enable the various applications, discussed above and to unlock the potential of ontologies to provide the required common framework of understanding.

[2]The term *ontology* in AI is borrowed from philosophy, where it stands for a *systematic account of existence*, or *what there is* in the world (http://plato.stanford.edu/entries/logic-ontology).

For that to take place, ontologies need to be brought to agreement and made compatible by reducing their heterogeneity. This can happen on different levels and to different extents: one could consider a small number of domain-specific ontologies or a bigger set of "inter-disciplinary" ontologies, link them entirely or partially, simply state the correspondences, or merge them (or their overlapping parts) into one global ontology. Any of these tasks amounts to consolidating heterogeneous ontologies by finding explicit correspondences between their elements (concepts, relations, instances, etc.) – a process that will be called *ontology matching*.

Illustration by an example

We will illustrate the problem by the help of a small fictional example. Imagine the following two enterprises: a news agency, collecting and editing news articles, and a bookstore, offering, among books and music, also the daily press. Now assume that each of the two enterprises has an *ontology*, which organizes and structures the knowledge necessary in order to run the company. The news agent's ontology would contain concepts like TITLE OF THE ARTICLE and AUTHOR, as well as the different topics on which news articles are written, such as POLITICS, RELIGION, AUTOS, SPORTS, etc. The bookstore's ontology, on the other hand, will also contain concepts like TITLE and AUTHOR, POLITICS, RELIGION, SPORTS, referring to books and not news articles. Since the book store also sells newspapers, it will also have a concept, say, PRESS, which will be the root of a small subtree of the big bookstore ontology, containing concepts, similar to the concepts of the news agency, i.e. TITLE OF THE NEWSPAPER, EDITOR, TOPIC, etc. The concept TOPIC, on its turn will have subconcepts such as POLITICAL LIFE, RELIGION AND SCIENCE, SPORTS, AUTOS, etc.

Now imagine that the news agency and the bookstore have decided to cooperate and for that ends they need to integrate their ontologies, so that communication and information flow between the two enterprises becomes easy and fast, or in other words, they have to *match* their ontologies one against the other. Assume that the ontologies of the two parties are equally large and specific. However, clearly the news agency's ontology would only cover a small part of the onotlogy of the bookstore – that part related to newspapers and daily press. The task will be to identify which part of the bookstore's ontology corresponds to the news ontology (that could be done by mapping the root node of the news ontology to a node in the bookstore ontology) and further find out what are the precise correspondences between the elements of the two ontologies (or precisely between the elements of the news ontology and that part of the bookstore ontology related to news), i.e. discover that the concept POLITICS in the news ontology corresponds to the concept POLITICAL LIFE in the bookstore's ontology; that the concept RELIGION from the news ontology is a sub-concept of the concept RELIGION AND SCIENCE from the bookstore ontology, and so forth. There are plenty of mismatches that a matching system has to cope with: use of different terminology, different specificity and granularity, different scope, etc.

For a thorough overview of the ontology matching problem and the existing approaches, we refer to the recent book by Euzenat and Shvaiko [43], to the survey of tools and theoretical frameworks by Kalfoglou *et al.* [80], and to the classification of different matching techniques by Rahm *et al.* [125].

1.2 Overcoming Ontology Heterogeneity: Applied Methodologies

Heterogeneity among ontologies can appear on different levels. Euzenat *et al.* [43] proposed one possible typology of the different kinds of semantic heterogeneity, claiming that it distinguishes between the most "obvious" ones dividing them in four main groups: syntactical, terminological, conceptual and semiotic. *Syntactically* heterogeneous ontologies are expressed in different formal languages [3]. *Terminological* heterogeneity, on the other hand, stands for variations in names when referring to the same entities and concepts. *Conceptual* heterogeneity refers to differences in *coverage*, *granularity* or *scope* when modeling the same domain of interest. Finally, *semiotic*[3] heterogeneity is about mismatches in how entities are interpreted by people in a given context and is hard to model computationally.

Note that the presented typology is not universal and usually the different heterogeneity types do not appear in isolation from one another. We refer the interested reader to the study on ontology mismatches of Klein [85], the heterogeneity typology suggested by Sheth *et al.* [139] and the classification of schema diversities by Batini *et al.* [10].

There exist many different theoretical attempts to deal with the ontology mismatch problem and reduce or eliminate the heterogeneity by introducing a semantic similarity measure of some kind and / or a procedure for alignment of the elements of the heterogeneous ontologies. Most roughly, the existing approaches can be divided into four basic groups with respect to the account of similarity that they consider: structural, terminological, extensional and semantic techniques [43]. Notice that the approaches from each group can serve to solve multiple types of heterogeneity. For instance, extensional approaches can help resolve terminological and conceptual differences; terminological approaches can be applied to resolving terminological and syntactical mismatches; etc. We will introduce each of the four groups of approaches by specifying the formal methodology most commonly applied for constructing solutions for each groups.

Structural techniques consider how different entities are related within a single ontology, taking an intensional standpoint by defining a concept in relation to other concepts. It is an old idea in the study of semantic memory that a concept is semantically closer to those concepts, which are nearest to it in a conceptual graph, a semantic network or an ontology (see the original response time experiments by Collins and Quillian [28]). Departing from that assumption and extending it, many authors have proposed efficient ontology matching solutions, based on the intuition that structure is a main container of the semantics of a group of concepts (see, for instance, the work of Noy [114], Bernstein *et al.* [96] and Mitra *et al.* [107]). Graph and lattice theory come in support of the representation, modeling and proof of the concepts of many of the matching systems, which rely on structure in order to account for the semantic similarity of entities.

[3]Following a discussion in personal communication with A. Mehler, we draw the reader's attention to a certain inaccuracy in the notation proposed in [43]; it would be more appropriate to talk about *Pragmatic*, instead of *Semiotic* heterogeneity, since semiotics encompasses semantics and syntactics.

Most of the structural approaches have, however, shown not to be sufficient alone for accomplishing entirely a matching task. Therefore they have been often applied in combination with terminological or instance-based methods. *Terminological matching* consists in applying two basic classes of techniques: first, a set of approaches for string and tree matching [54], [69] in order to find lexical correspondences between terms (e.g. names of concepts and relations), and, second, a group of methods relying on external (lexical) resources, such as dictionaries or thesauri combined with approaches from Natural Language Processing, in order to account for problems related to synonymy and polysemy of terms [158, 144].

Instance-based or *extensional ontology matching* comprises a set of theoretical approaches and tools for measuring the semantic proximity of two ontologies based on their extensions – the instances that populate their concepts. Commonly, concepts are modeled as sets of instances, grounded in the external world. The relatedness of a pair of concepts is an outcome of a properly chosen measure of similarity, based on estimations of the similarity of the instance sets of these concepts [39, 72]. Instance-based approaches rely on the knowledge contained in the extension of the part of the world that we model and therefore naturally machine and statistical learning methods for inferring rules and decisions from a collection of observations over facts in the world are most commonly used for accomplishing the matching task.

Instance-based concept mapping is also central for most of the questions that this thesis attempts to answer. In our contribution to bringing ontologies into mutual agreement, we rely on three main statistical and machine learning concepts. The first one is that of *automatic classification*, or the task of learning a binary decision rule based on a collection of categorized observations in order to be able to infer the class of an unseen individual. We put an emphasis on the use of a fairly famous set of classification techniques known as Support Vector Machines (from hereon, SVMs) [30]. The second central concept that we rely on is that of *variable selection*, which comprises methods for reducing the dimensionality of an input dataset by indicating which are the most or less informative features with regard to the output and with respect to certain criteria. Among standard variable selection techniques [168], we introduce a novel approach based on Support Vector Machines. Third, we apply methods from basic data analysis and *descriptive statistics*, such as Principal Components and Discriminant Analyses [77, 48] for the task of detecting potential concept similarities (also referred to as *concept mappings* in the sequel).

Finally, the group of *semantic-based approaches* comprises a set of methods based on logical deduction. These methods provide mechanisms to justify and verify a set of previously generated mappings – result of using external background knowledge of some kind (formal top-level ontologies [50, 121], domain specific formal ontologies [110] or informal resources, such as WordNet[4]). This set of techniques falls out of the scope of the studies carried out in this thesis; a short introduction will be given for completeness later on.

[4]http://wordnet.princeton.edu/

1.3 Main Hypothesis and Research Questions

We will present in a synthesized manner the initial questions which the study presented in this thesis originates at.

Statement of main hypothesis

The central goal of our research efforts is to provide generic, theoretically sound and practically efficient methods for overcoming semantic discrepancies and reducing heterogeneity among ontologies. Our focus falls on ontologies that have been created with the intention to organize instantional data, such as text documents, with respect to their contents. (Examples of such ontologies are web-directories, such as *Yahoo!*[5], the open directory project[6], and other.) We note that the ontologies to which the approaches developed in this thesis can be applied are not limited to text populated ontologies, but include ontologies for which text document-like instances can be generated (these can be as well general OWL ontologies or multimedia resources containing text annotated images), as well as ontologies populated with multimedia instances, such as images (like, for example the LSCOM ontology[7] or the WordNet/LabelMe ontology[8]), for which a proper feature vector representation can be made available. In view of this remark, throughout the rest of the thesis, the terms "document" and "instance" will be used in an interchangeable manner.

Our central hypothesis can be articulated as follows.

> *The heterogeneity between two instance-populated ontologies, assumed to cover (partially) overlapping domains of knowledge, can be reduced efficiently by the help of extensional approaches to concept mapping, optimized by structural accounts for ontology similarity.*

Local questions

Several main research questions stem from the thesis stated above.

1. *What is an appropriate measure of cross-ontology[9] concepts similarity?*

   We have hypothesized that the similarity of concepts, taken from ontologies, which intend to organize text documents (instances) with respect to a certain set of categories, can be most efficiently measured on extensional basis, that is, by using the documents (instances), which are assigned to these concepts as a basis for deriving similarity criteria.

2. *What out of all possible aspects of similarity should be taken into account, provided a similarity measure is chosen?*

   In contrast to many existing solutions in the domain of instance-based ontology matching, we base our measure of concept similarity not directly on the sets of instances, but on the features, which describe them. In fact,

[5]http://www.yahoo.com
[6]http://www.dmoz.org
[7]http://www.lscom.org/ontology/index.html
[8]http://people.csail.mit.edu/torralba/research/LabelMe/wordnet/test.html
[9]Here and hereafter the term "cross-ontology concepts" is used to denote concepts taken from two (or more) different ontologies.

we propose a set of concept similarity criteria, based on (a small subset of) characteristic features selected separately and independently for every concept. Our endeavours are motivated by the following heuristic:

> Similar concepts are characterized by similar (sub)sets of features; otherwise are dissimilar ones.

3. *How can the suggested concept similarity measure be applied in an optimal way on the entire sets of concepts of the source ontologies?*

   Although powerful in estimating the similarity of classes of instances, extensional approaches alone are not sufficient for building an optimal ontology matching solution. Therefore, we propose methods of applying the suggested instance-based measures by taking into account the structural properties of the input ontologies and the inter-ontology relations of concepts.

1.4 Results and Contribution

The work that forms the basis of this dissertation contributes to the research in the field of ontology matching by providing theoretical and technical solutions on two basic levels in terms of:

- Generic methods for concept alignment through similarity measures based on the ontologies extensions;
- Procedures for overall ontology matching and merging (if required), based on combined structural and instance-based similarity criteria.

We will summarize the results from each of the two main research directions listed above.

Extensional concept mapping

We consider as central the finding that similarity between concepts can be measured on extensional basis by the help of the variables that describe the instances that populate these concepts. We have demonstrated that, as a result of a properly performed variable selection procedure, every concept in an ontology can be represented as a list of variables scored with respect to their discriminative power. By discriminative power we understand the importance of every single variable for the separation of the instances in an ontology into those that belong to the concept in question and those that do not. In fact, a concept can be represented by using only a small subset of the whole set of variables containing those with the highest scores. They are most characteristic for this concept and distinguish it best from the rest of the concepts within the same ontology. We introduce and compare several measures of semantic proximity of two cross-ontology concepts based on the small subsets of characteristic variables corresponding to each of the concepts by using a simple set-theoretic measure, or based on the whole lists of scored variables by using statistical correlation coefficients [151].

Important information concerning the structure of the two source ontologies can be revealed by the help of statistical analysis of their groups of instances.

As an additional contribution of the thesis, we propose an approach to detect potential concept mappings based on basic descriptive statistical methods, such as Principal Components Analysis and Discriminant Analysis [149], which can be used either in combination with the proposed variable selection-based techniques, or, under assumptions to be specified later, self-dependently.

Finally, the thesis contains an independent contribution to the machine learning-related research: a novel technique for variable selection in classification elaborated for Support Vector Machines. We propose and empirically evaluate the qualities of an important parameter of the SVMs – their *VC-dimension* – as an indicator of the importance of a single variable, or a block of variables, for separating the instances in a training dataset in a binary classification task [151].

Combined ontology matching procedures

We have observed that, in spite of the excellent results achieved on small sets of concepts, the suggested instance-based concept similarity measures are likely to perform poorly on a large scale. Clearly, simply taking the two sets of concepts and measuring one-to-one similarities would be a fairly time expensive enterprise with real-life ontologies, containing many concepts. But doing so would be also conceptually wrong, for the structure of the classes in both ontologies would not be taken into account. What is the use of naming that an *ontology* matching process, when it comes down to comparing sets of flat collections of classes?

In order to account for these problems we have suggested two overall procedures for ontology matching, which build on the proposed measures of concept similarity, but also take structure into consideration.

The first suggested procedure consists in optimizing the search of similar pairs of concepts by using the structural relation of the concepts within each ontology and recursively applying the concept similarity measure on sets of homogeneous ontology classes. The procedure accounts for the two critiques mentioned above: it considers the structure of the ontologies, and minimizes the number of concepts, for which a similarity check will be performed. Its basic heuristic can be formulated roughly as: *we do not need to check the similarity of the descendants of dissimilar concepts within the two input hierarchies; we do need to check the similarity of the descendants of similar ones.* The procedure is formulated for the case when the two source ontologies are strictly hierarchical and equally deep in terms of levels, but a method for generalization of the suggested approach on structurally dissimilar, non-hierarchical ontologies is proposed. The output of the procedure is a set of pair-wise concept alignments.

In many applications it might be required that the two input ontologies are merged into one, by integrating their overlapping parts. While attempting to enrich and improve the extensional similarity techniques in order to account for the structure and to reduce the dimension of the problem, we came up with an additional finding, in support of the process of ontology merging. Combining graph representation of ontologies with instance-based techniques, and taking into account the sizes and intersections of the sets of instances of the two source ontologies leads to assertions on the specificity, granularity and instantiation mismatches between them. These assertions can be useful in providing guidelines to an ontology engineer through a process of ontology merging, when ontology merging is required for the needs of the concrete task and application

[148, 150]. As noted, the suggested approach relies on the *intersection* of the sets of instances of the two ontologies (among with structural and extensional ontology similarity measures). An additional contribution of the approach is a more relaxed definition of a set intersection, which accounts for the semantics of the instances contained in these sets (which in our case are text documents written in some natural language) and not simply for identical set members.

As a general remark, we note that structure-related ontology matching techniques are not sufficient and reliable indicators of the semantic similarity of two ontologies, but in combination with extensional approaches they can help reduce time complexity and optimize search among the sets of concepts, yielding improved overall results.

1.5 Outline and Intended Audience

The contents of the thesis is divided in six chapters, including the current, introductory **Chapter 1**.

Chapter 2 reviews basic results in fields of mathematics and machine learning, which are relevant to both the previously existing and the original approaches to ontology matching discussed later on in the thesis. The structural properties of ontologies are commonly, and in our case, accounted for by graph and lattice representations. Therefore, the first section of this chapter expands on basic results and definitions from *graph and lattice theory* (Section 2.1). The rest of the chapter is dedicated to methods, which will be further used for the study of the extensional concept and ontology semantic similarity. These include an introduction to some concepts and methods from statistical and machine learning: the *Support Vector Machines classifier* (Section 2.2), *variable selection* (Section 2.3), and *text categorization* (Section 2.4).

Throughout **Chapter 3**, we discuss various issues related to the interdisciplinary research field of *ontology matching*. These comprise definitions and areas of application of ontologies (Sections 3.1 and 3.2), models for knowledge representation (Section 3.3), motivational and conceptual problems related to the question of where the need of ontology matching stems from (Section 3.4) and, finally, a presentation and classification of several existing ontology matching theoretical endeavors and engineering solutions (Section 3.5). On one hand, the chapter intends to provide the reader with a structured account of that part of the state-of-the-art of the ontology matching research, which is relevant to the approaches discussed throughout the thesis. On the other hand, the chapter aims to open a broader view to the origins of the ontology heterogeneity problem drawing motivations from psychology and cognitive science and prepare the ground, which our own contribution to that field will be built on.

Chapter 4 contains the major part of the theoretical results of the thesis. It begins with definitions and preliminary remarks, which set the basic assumptions of the ontology matching model that we develop (Section 4.1). The chapter proceeds to connect the results from graph and lattice theory discussed in Chapter 2 with the structural properties of the ontologies, defining a measure of structural ontology similarity based on solving a graph isomorphism problem (Section 4.2). As mentioned earlier, instance-based ontology mapping is central to the problematics of this thesis. Therefore, several novel approaches to identify instance-based similarity of inter-ontology concepts, based on descriptive statistics, SVMs and variable selection are elaborated next, in terms of criteria and measures for semantic proximity (Section 4.3). Finally, the two major groups of similarity criteria discussed so far, structural and extensional, are combined together in two independent ontology matching procedures (Section 4.4 and Section 4.5).

The major theoretical findings of the preceding sections are supported empirically in **Chapter 5**. More precisely, we provide a thorough evaluation of the proposed instance-based techniques for measuring cross-ontology concept similarity via descriptive statistics (Section 5.1) and variable selection for SVMs or other approaches (Section 5.2). The overall variable-selection-based ontology matching procedure is evaluated by the help of two ontologies intending to organize news-related documents in tree-structured directories (Section 5.3). The chapter closes with a discussion (Section 5.4).

Finally, **Chapter 6** summarizes, concludes and sketches directions for further research in terms of a set of open questions and possible technical improvements of the proposed solutions.

The thesis is oriented towards both researchers and practitioners in the field of ontology matching and semantic similarity, as well as interested readers from neighboring fields, such as machine learning and text categorization. Applications of the proposed approaches can be found in fields as various as the Semantic Web, web-directory mapping, overcoming the problems of the semantic gap, information retrieval and other. The theoretical results of the work are intended and have been tested on text-related applications. However, the findings are general and formal enough so that all the discussed approaches can be generalized to other kinds of non-textual multimedia data, such as images and videos.

Chapter 2

Background Concepts

The problem of ontology matching can be approached from many different angles. The palette of theoretical and methodological tools which are being applied to the end of accomplishing an ontology matching task of some kind is quite colorful: different research groups work on different aspects of the problem by applying different methodologies which have their roots in various fields of mathematics, logics, computer science, machine learning, relational algebra and other.

In this chapter, we provide the minimal necessary background knowledge in several fields in mathematics and statistical learning, which are important for understanding our approach to ontology matching and for enabling us to discuss other related techniques. We start by reviewing the basics in the theory of graphs and lattices (Section 2.1) which is important in our study of the structural properties of two ontologies in terms of their shared commonalities. Further (Section 2.2), we give an introduction to the Support Vector Machines, which we will apply during the instance-based concept mapping phase, together with several Variable Selection techniques (discussed in Section 2.3). The main intended application field of the ontology matching strategy that we have developed is facilitating text-related categorization problems. For that reason the chapter closes with an overview of text categorization methods (Section 2.4).

We note that the content of the chapter might look slightly disunited, at a first glance: there are multiple sections related to common concepts (e.g. SVMs, variable (feature) selection, Latent Semantic Analysis, kernels for structures). This is due to the fact that many of the fields presented in the chapter are strongly interrelated and aspects of many of these fields are needed in order to present aspects of other fields. For example, SVMs have been introduced independently, but also as a feature selection technique and as a text categorization tool; kernels for structures are discussed with regard to graph matching, but also with regard to text categorization; etc. We have attempted, however, to make the distinctions between the different approaches in the light of their different applications as clear as possible.

2.1 Graphs and Lattices

An ontology provides a highly structured representation of knowledge. Graphs and lattices have been widely applied for embodying the structural properties of ontologies. In our approach to ontology matching, we will rely on a couple of main definitions and result from graph and lattice theory, needed in order to be able to discuss the problem of ontology matching as a problem of identifying similarity between structures. Therefore, we begin this chapter by expanding on the necessary ground knowledge in these fields. Thorough introductions to graph and lattice theory are found, for example, in the books by Valiente [155] and Davey *et al.* [33]. We will broadly follow the notations used by these authors.

2.1.1 Facts about Graphs and Trees

A graph is a structure, which consists of a set of objects called **vertices** and a (possibly empty) set of pairs of vertices, called **edges**, or **arcs**. There are different kinds of graphs with respect to the precise abstract representation that they embody. An (un)directed graph is one whose edges are (un)ordered pairs of vertices. The notion of a graph that will be most useful for our goals, and which is most broadly considered in computer science in general is that of a directed graph. We give the following definition.

Definition 1 Graph and subgraph. *A graph $G = (V, E)$ consists of a finite set V of vertices and a finite set $E \subseteq V \times V$ of edges. The number of vertices n of G is defined as its* **order***, $n = |V|$. The* **size***, denoted by m, is the number of edges, $m = |E|$. An edge $e = (v, w)$ is said to be* **incident** *with vertices v and w, where v is the* **source** *and w is the* **target** *of the edge e, and vertices v and w are said to be* **adjacent***. A graph $H = (W, S)$ is called a* **subgraph** *of G if $W \subseteq V$ and $S \subseteq E$.*

The **degree** of a vertex v in a graph G is the number of edges whose target or source is the vertex v. Vertices and edges can have associated names or numbers in which case we say that a graph is *labeled.* Let L_V and L_E denote the finite sets of vertex and edge labels, respectively. The members of the label sets can be natural or rational numbers (known as weights), as well as strings of characters. Formally speaking, a **labeled graph** is a 4-tuple $G(V, E, h_V, h_E)$, where $h_V : V \to L_V$ is a function assigning labels to the vertices and $h_E : E \to L_E$ assigns labels to the edges.

By ordering the vertices and edges of a graph in sequences we define walks, trails and paths. Formally, an alternating sequence of vertices and edges

$$[v_i, e_{i+1}, v_{i+1}, e_{i+2}, ..., v_{j-1}, e_j, v_j],$$

such that $e_k = (v_{k-1}, v_k)$ for $k = i + 1, ..., j$, is called a **walk** from vertex v_i to vertex v_j. A walk with no repeated arcs is called a **trail**, and a trail with no repeated vertices, except (possibly) the initial and final ones is called a **path**. An integer, associated to the walk, trail, or path of a graph, corresponding to the number of edges in the sequence, is called **length**. A graph G is called **connected** if for every pair of vertices $v, w \in E$, there exists a walk from v to w. A path with coinciding start and end vertices is called a **cycle** and a graph containing cycles is said to be **cyclic**. Reversely, a graph not containing cycles is called **acyclic**.

A class of graphs, commonly used in multiple applications and computational models, which will be of particular interest for our study is the class of (directed rooted) trees. A part of the theoretical study of this thesis is done by modeling hierarchical ontologies as such trees, therefore we will discuss some of their properties in more detail in the sequel.

Definition 2 Directed rooted tree. *A directed tree is a connected acyclic graph $G(V, E)$, where $V(G)$ denotes the set of vertices of G and $E(G)$ denotes the set of ordered pairs of vertices of G, called edges. A tree is called a rooted tree if there exists a vertex designated as "root" and all edges have an orientation – either towards or away from the root.*

Let $\texttt{root}(G)$ denote the root node of a tree G and $\texttt{leaves}(G)$ – the set of its leaf nodes. A tree, which is contained within another tree is called a *subtree* of the latter and is defined as it follows. Let G=(V,E) be a tree and let `children`(v) denote the set of children of a node v and `parent`(v) denote the parent of a node v, $\forall v \in V$, $v \neq \texttt{root}(G)$.

- A tree (W,S) is said to be a **subtree** of the tree G if $W \subseteq V$ and $S \subseteq E$.
- A subtree (W,S) is a **top-down subtree** of G if $\texttt{parent}(v) \in W$, $\forall v \in W : v \neq \texttt{root}(G)$, where `parent` is the parent function of the sub-tree.
- A tree (W,S) is a **bottom-up subtree** of G if $\texttt{children}(v) \subseteq W$, $\forall v \in W : v \notin \texttt{leaves}(G)$, where `children` is the children function of the subtree.

One distinguishes between *ordered* and *unordered* trees depending on whether the order of the children of each node is taken into account or not. Ordered trees are defined as trees for which the relative order of the successors of a node is fixed. The order property of the nodes within a tree is not relevant to the ontology structural property that they shall embody. For example, in the taxonomy describing the morphological types of mammals, it is not important whether the subclasses "flying mammals", "water mammals" and "tree mammals" will appear in that, or in a different order. What is important is the relation of each of the sub-classes to its super-ordinate category "mammals" and its sub-ordinate categories. For that reason, in what follows, by trees we will understand unordered trees.

Finally, let us introduce one more important and commonly used class of graphs, the class of directed acyclic graphs, often abbreviated as DAG. DAGs are more general than trees, but share certain commonalities with them. Formally, they are defined as directed graphs with no directed cycles (i.e. no path starts and ends in the same node). DAGs are similar to directed trees, because their nodes are connected by edges in one single direction; dissimilarly to trees, a DAG node can have more than one parent (see Figure 2.1 for a graphical disambiguation). In section 4.2.2 we will show under what conditions it is possible to generalize trees to DAGs and thus generalize the presented approach to less particular structures than trees, allowing relations other than subsumtion, as well as multiple inheritance.

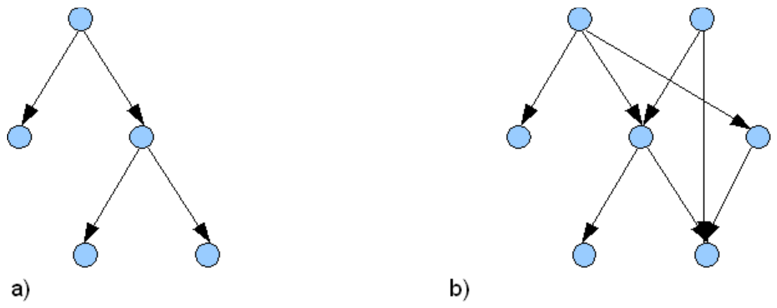


Figure 2.1: a) A directed rooted tree; b) A directed acyclic graph

We will proceed to discuss isomorphism of graphs and trees, which will be the basis of a measure of the structural similarity of two ontologies, to be introduced later on in the thesis.

2.1.2 Graph and Tree Isomorphism

In the ensuing chapters, we will be interested in measuring the similarity between ontologies expressed as trees or general graphs, which is reduced to the well known problem of graph matching. A core notion needed for defining graph similarity is that of graph *isomorphism* – a structure preserving mapping from one graph to another (Figure 2.2).

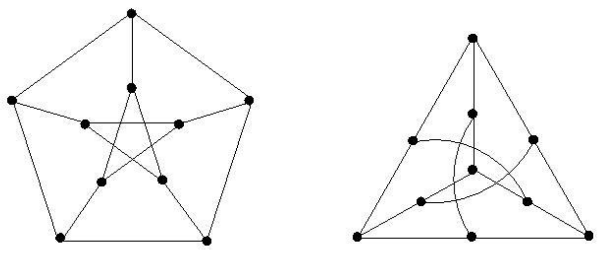


Figure 2.2: Isomorphic graphs: an example.

Definition 3 Graph Isomorphism. *A bijective function $\mu : V_1 \to V_2$ is a **graph isomorphism** from graph $G_1(V_1, E_1)$ to a graph $G_2(V_2, E_2)$ if for any $v_1, v_2 \in V_1$*

$$\langle v_1, v_2 \rangle \in E_1 \Leftrightarrow \langle \mu(v_1), \mu(v_2) \rangle \in E_2.$$

We have previously defined a labeled graph. In many cases, one needs to take into account the nodes and edges names in order to establish their matching. A possible way of defining labeled graph isomorphism is presented in the following definition.

Definition 4 Labeled Graph Isomorphism. *A bijective function $\mu : V_1 \to V_2$ is an **isomorphism** from the labeled graph $G_1(V_1, E_1, h_{V_1}, h_{E_1})$ to a labeled graph $G_2(V_2, E_2, h_{V_2}, h_{E_2})$ if for any $v_1, v_2 \in V_1$, $\langle v_1, v_2 \rangle \in E_1$ it holds:*

$$\langle \mu(v_1), \mu(v_2) \rangle \in E_2 \tag{2.1}$$
$$h_{V_1}(v_1) = h_{V_1}(\mu(v_1)) \tag{2.2}$$
$$h_{E_1}(\langle v_1, v_2 \rangle) = h_{E_1}(\mu(\langle v_1, v_2 \rangle)). \tag{2.3}$$

Throughout our study, we will not consider isomorphism of labeled graphs, for its definition is too restrictive in the ontology matching domain, where it will often not be the case that mapped nodes have identical labels. In fact, the matching approaches presented later do not rely on information related to the entities names at all. The label matching of both nodes and edges is left for a separate linguistic-based study. All definitions which follow will consider the unlabeled case.

Definition 5 Subgraph isomorphism. *An injective function $\mu : V_1 \to V_2$ is a subgraph isomorphism from G_1 to G_2 if there exists a subgraph $S \subseteq G_2$ so that μ is a graph isomorphism from G_1 to S.*

Finding isomorphism properties of two graphs or parts of them leads to identifying *common subgraphs*. Among the set of common subgraphs of two graphs G_1 and G_2, the *maximal common subgraphs*, denoted by $mcs(G_1, G_2)$ are the ones with the maximal order. They will be of particular importance for the measure of structural ontology similarity, to be introduced in Chapter 4.

Definition 6 Maximal common subgraph. *Let G, G_1 and G_2 be graphs. G is a **common subgraph** of G_1 and G_2 if it exists a subgraph isomorphism from G to G_1 and from G to G_2. A common subgraph is **maximal** if it exists no other subgraph isomorphism from G to G_1 and G_2 that has more nodes than G.*

A maximal common subgraph is then the maximal graph isomorphic to subgraphs of both G_1 and G_2 and need not be unique for two given trees or general graphs.

We now move the focus from general graphs to trees, since they are the basic structural representational body for hierarchical ontologies (Section 4.2). Determining whether two trees are isomorphic is a problem, which lies in the core of comparing hierarchical structures.

In conformity with definition 3, two trees are said to be isomorphic if there exists a bijective mapping between the sets of nodes of both trees, preserving their structure. In fact, the root of one tree is mapped to the root of the other and any two adjacent nodes in one tree are mapped to adjacent nodes in the other. In order to identify whether two (unordered) trees are isomorphic, a broadly applied method assigns a unique *isomorphism code* to the root nodes of each tree. The necessary and sufficient condition that two trees are isomorphic is given by the identification of coinciding isomorphism codes. The isomorphism code of the root of a tree $G(V, E)$ of order n is defined as the sequence of n integers in the range $1, ..., n$

$$code[root[G]] = [size[root[G]]], code[w_1], ..., code[w_k],$$

where the nodes $w_1, ..., w_k$ are the children of the root of G. The **isomorphism code of a tree** is the isomorphism code of its root node.

Note that the given definition suggests that every non-root node of G is assigned an isomorphism code which participates in the formation of the isomorphism code of its parent. A simple tree isomorphism criterion results from comparing the isomorphism codes of two trees.

Theorem 7 *Two trees G_1 and G_2 are isomorphic if and only if $code[G_1] = code[G_2]$.*

Identifying Tree Isomorphism

According to theorem 7, in order to find out whether two trees are isomorphic or not we only need to calculate and compare the isomorphism codes of their roots. One algorithm for computing isomorphism codes of the nodes of two trees consists in performing a postorder traversal[1] of the two trees. In that way, all nodes are assigned an isomorphism code, which is needed in order to compute the isomorphism bijection $M \subseteq V \times V$ (cf. appendix of [155]). The algorithm for identifying an isomorphism of two trees consists in first checking whether they are of the same order and then comparing their isomorphism codes.

Identifying Subtree Isomorphism

The problem of subtree isomorphism identification, or determining whether a tree is isomorphic to a subtree of another tree, is somewhat more complex than the particular case just discussed. A standard algorithm for general subgraph isomorphism was introduced by Ullman in the 70s [154]. It is based on a brute-force tree-search enumeration procedure and attains efficiency by inferentially eliminating successor nodes in the tree search. In this section, we will present an algorithm for finding sub-graph isomorphism in the particular case of trees.

As we have observed in the previous section, there can be bottom-up or top-down sub-trees of a tree. One distinguishes between bottom-up and top-down isomorphism with respect to whether a tree is isomorphic to a bottom-up or a top-down subtree of another tree. We will focus on top-down subtree isomorphism because it will be of interest for the structural ontology matching procedure to be introduced in Section 4.2.

Definition 8 *A tree $G_1 = (V_1, E_1)$ is isomorphic to a top-down subtree of another tree $G_2 = (V_2, E_2)$ if there is an injection $M \subseteq V_1 \times V_2$ with the following properties:*

(1) $(root[G_1], root[G_2]) \in M$,

(2) $(parent[v], parent[w]) \in M$ for all nodes $v \in V_1$ and $w \in V_2$ such that $v \neq root[G_1]$, $w \neq root[G_2]$ and $(v,w) \in M$.

The injection M is called a top-down subtree isomorphism of G_1 to G_2.

In order to present the intuition behind the actual algorithm of determining subtree isomorphism we will informally introduce *bipartite graphs*.

A graph G is defined to be a bipartite graph (or a bigraph) if the set of its vertices can be partitioned into two disjoint sets, V' and V'', in such a manner that for every edge $e = (v, w) \in E$ one of both holds: $\{v \in V'$ and $w \in V''\}$

[1] A traversal of a tree or a graph refers to an algorithm, which visits (exactly once following a predefined system) and examines (and possibly changes) each node of the tree or graph. Different kinds of traversals are discussed in, for instance, [155].

or $\{v \in V''$ and $w \in V'\}$. A *matching* in a bipartite graph is defined as a set of edges E' within the graph with the property that no two edges in E' share a common vertex, i.e. $e' = (v', w') \in E'$ and $e'' = (v'', w'') \in E'$ if and only if $v' \neq v''$ and $w' \neq w''$. A *maximal matching* is a matching that is no longer one after adding one more edge and a *maximum matching* is the matching with the largest number of edges possible. The identification of a set of edges within a graph, which is a matching of maximal cardinality is defined as a *maximum bipartite matching problem.*

The algorithm for identifying the existence of a top-down subtree isomorphism from a tree G_1 into a tree G_2 and computing it is based on solving the following maximum bipartite matching problem. Assume that we want to verify whether or not a node v of G_1 can be mapped to a node w in G_2, where the mapping is understood as in definition 8. Let p and q be integers and let $v_1, ..., v_p$ and $w_1, ..., w_q$ be the children of nodes v and w, respectively. We construct a bipartite graph $G_{v,w} = (\{v_1, ..., v_p\}, \{w_1, ..., w_q\}, E_{v,w})$ on $p+q$ vertices with the property that an edge $(v_i, w_j) \in E_{v,w}$ if and only if the node v_i can be mapped to the node w_j (i and j are integers in the ranges $(1, ..., p)$ and $(1, ..., q)$, respectively). If the bipartite graph $G_{v,w}$ has a maximum bipartite matching with p edges, the node v can be mapped into the node w. Recursively solving the problem for the nodes of G_1 determines whether or not there is a top-down subtree isomorphism of G_1 into G_2.

The algorithm for the construction of an actual isomorphism is based on the following result.

Proposition 9 *Let $G_1 = (V_1, E_1)$ be a tree isomorphic to a top-down subtree of a tree $G_2 = (V_2, E_2)$. Let $B \subseteq V_1 \times V_2$ be the set of solutions to all the maximum bipartite matching problems solved during the top-down subtree isomorphism procedure on G_1 and G_2, described above. Then, there exists a unique top-down subtree isomorphism M such that $M \subseteq B$.*

Maximal and Maximum Common Subtree Isomorphisms

We proceed to define the central notions of a maximal and maximum common subtree of a pair of trees. Again, due to the fact that subtrees can be top-down or bottom-up, there is also a distinction between a top-down and a bottom-up maximal and maximum common subtree. With a similar motivation as in the previous section, again the definitions and results to be presented in the sequel only hold for top-down common subtrees.

Definition 10 *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two trees. A* **top-down common subtree** *of G_1 and G_2 is a triple (X_1, X_2, M), where $X_1 = (W_1, S_1)$ is a top-down subtree of G_1, $X_2 = (W_2, S_2)$ is a top-down subtree of G_2, and $M \subseteq W_1 \times W_2$ is a tree isomorphism of X_1 to X_2. A* **maximal top-down common subtree** *of G_1 to G_2 is a top-down common subtree (X_1, X_2, M) such that there is no top-down common subtree (X_1', X_2', M') of G_1 to G_2 such that X_1 is a top-down subtree of X_1' and X_2 is a top-down subtree of X_2'. The tree (X_1, X_2, M) is* **maximum** *if there is no top-down common subtree (X_1', X_2', M') of G_1 to G_2 such that the size of X_1' is larger than the size of X_1.*

Lemma 11 *A maximum common top-down subtree of a tree G_1 to a tree G_2 is also a maximal top-down common subtree of G_1 to G_2.*

We will use the abbreviation *mcs* to stand for a maximal common subtree.

The algorithm for constructing a top-down *mcs* isomorphism of a tree G_1 to a tree G_2 consists in constructing *mcs* isomorphisms of each of the subtrees of G_1 rooted at a non-root node v into each of the subtrees of G_2 rooted at a non-root node w. The resulting recursive algorithm is based on applying the divide-and-conquer technique discussed in [155].

Similarly to the previous section, we will first describe a procedure for verifying whether or not two nodes are to be mapped by a *mcs* isomorphism. The procedure is again based on the construction of a bipartite graph, and solves, this time, a *maximum weight matching problem.* A maximum weight matching problem applies for weighted bipartite graphs and consists in finding a matching whose edges' weights sum up to a maximal value.

Let the node v of G_1 and the node w of G_2 be the two potential mapping candidates with respective children $v_1, ..., v_p$ and $w_1, ..., w_q$ (p and q are integers). We construct a bipartite graph $G_{v,w} = (\{v_1, ..., v_p\}, \{w_1, ..., w_q\}, E_{v,w})$ on $p+q$ vertices with the property that an edge $(v_i, w_j) \in E_{v,w}$ if and only if the *mcs* of the subtree of G_1 rooted at v_i and the subtree of G_2 rooted at w_j is non-empty. Let, in addition, that nonzero size be assigned as a weight to the edge (v_i, w_j). The size of the *mcs* of the subtree of G_1 rooted at v and the subtree of G_2 rooted at w equals to the weight of a maximum weight bipartite matching in $G_{v,w}$ plus one.

Lemma 12 *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two trees. Let $B \subseteq V_1 \times V_2$ be the set of solutions to all maximum weight bipartite matching problems solved during the procedure described above. Then, there is a unique top-down mcs isomorphism $M \in V_1 \times V_2$ with the property $M \subseteq B$.*

The problem of finding a *mcs* for general graphs is NP-complete but in the case of trees it can be solved in polynomial time [52]. Procedures are discussed in [38]. Detecting maximum and maximal subgraphs in general in terms of theoretical and algorithmic approaches is discussed in [93, 101].

2.1.3 Graph Similarity Measures and Distances

In the following, we will review some of the most commonly applied metrics and measures of graph similarity in the context of the problem of graph matching. Before doing so, we introduce definitions of a distance function and similarity measure which will be needed throughout the rest of the thesis and therefore are not only relevant to the contents of the current chapter.

Metric Spaces and Similarity Functions

Measuring graph similarity comes down to defining and applying an appropriate formal quantification of the shared commonality of two given graphs. Intuitively, similarity between two objects can be measured in terms of their distance in a certain (metric) space or in terms of shared common features. We will start by defining and distinguishing between a distance function and a similarity (dissimilarity) measure, which are core notions for the graph similarity problem.

Definition 13 *Let X be a set and let $x, y, z \in X$. A* **metric** *or a* **distance function**[2] *on the set X is a function $f : X \times X \to \mathbb{R}$ with the following properties:*

$$f(x,y) \geq 0, f(x,y) = 0 \Leftrightarrow x = y \quad (positive\ definiteness) \tag{2.4}$$
$$f(x,y) = f(y,x) \quad (symmetry) \tag{2.5}$$
$$f(x,z) \leq f(x,y) + f(y,z) \quad (triangle\ inequality) \tag{2.6}$$

A pair of a set and a metric (X, f) defines a **metric space**.

The notion of similarity or dissimilarity is more liberal than that of a metric. Dissimilarity is intuitively related to the distance between two entities, whereas similarity is the exact inverse of dissimilarity. To assess these concepts usually a measure of dissimilarity is defined which relaxes one or more of the conditions for a distance function. We will introduce the most commonly adopted definitions of similarity and dissimilarity between two entities [37].

Definition 14 Dissimilarity. *Let X be a set and let $x, y \in X$. A dissimilarity function on the set X is defined as a mapping $\delta : X \times X \to \mathbb{R}$ with the following properties:*

$$\delta(x,x) = 0 \quad (minimality) \tag{2.7}$$
$$\delta(x,y) \geq 0 \quad (positiveness) \tag{2.8}$$
$$\delta(x,y) = \delta(y,x) \quad (symmetry) \tag{2.9}$$

Definition 15 Similarity. *Let X be a set and let $x, y, z \in X$. A similarity function on the set X is defined as a mapping $\sigma : X \times X \to \mathbb{R}$ with the following properties:*

$$\sigma(x,x) \geq \sigma(y,z) \quad (minimality) \tag{2.10}$$
$$\sigma(x,y) \geq 0 \quad (positiveness) \tag{2.11}$$
$$\sigma(x,y) = \sigma(y,x) \quad (symmetry) \tag{2.12}$$

The problem of *graph matching* refers to determining the similarity of two graphs. In a general statement of the problem, it is reduced to finding an isomorphism mapping between two graphs. Following the classification of graph matching problems given in [11], we distinguish between *exact* and *inexact* matching. Exact matching is defined as the graph matching problem when there exists an isomorphism from one graph to another or from a subgraph of a graph to (a subgraph of) another graph. The term inexact matching is used to denote a class of matching problems for which it is not possible to find an isomorphism between the two input graphs. It consists in finding the *best* possible matching between the vertices of two graphs, rather than the *exact* node-to-node correspondence.

From a complexity viewpoint, we can distinguish between graph matching problems which are of *undefined complexity* (the general category of exact graph matching problems (finding a graph isomorphism) with exceptions for some

[2]In computer science, the notion of s distance is sometimes considered as weaker and thus not synonymous to a metric, in contrast to the pure mathematical sense of this notion. Example is the edit distance, to be discussed below, which is not a metric in the general case. However, throughout the thesis by metric and distance function we will refer to the same concept.

particular kinds of graphs (e.g. isomorphism of planar graphs [70])), problems which have been proven to be *NP-complete* (the whole class of exact sub-graph matching problems (finding sub-graph isomorphism) [52]), and, finally, problems of *polynomial complexity* (tree matching problems [116], [122], [160]).

Finally, the algorithms available for solving graph matching problems can be classified into *optimal* and *suboptimal*. As we have already observed, the problem of finding a maximal common subgraph, which underlies most graph matching algorithms, is NP-complete. A couple of state-of-the-art graph matching algorithms and approaches designed for various application fields [133], [137], [152] provide an *optimal* solution in exponential time and space which makes them computationally intractable. On the other hand, *suboptimal* or *approximative* methods are able to find a solution in polynomial time, but give no guarantee that the solution found is not due to a local minimum trap [5], [32].

For a more advanced discussion of the overall problem of graph matching in terms of theoretical foundations, algorithms and applications, we refer to [11] and [19]. In the next sub-sections, we will define and discuss several of the most commonly used graph distances and similarity measures which underlie and are independent of the graph matching algorithm to choose.

Edit Distance

Edit distance is one of the most frequently used measures of graph similarity, broadly applied for trees. The main idea behind it is that a tree can be transformed into another tree by applying a set of operations, such as deletion or insertion of vertices. Then, the distance between two trees is defined as the shortest sequence of such operations, called *edit* operations, that transforms one tree into another. However, since one edit operation might be more costly than another, often the *costs* of the operations are taken into account in addition to their number [19].

Definition 16 *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two trees. Define the following set of* **elementary edit operations** *on G_1 and G_2:*

1. **deletion** *of a leaf node $v \in V_1$ from G_1, denoted (v, λ);*
2. **insertion** *of a node $w \notin V_2$ as a new leaf into G_2, denoted (λ, w);*
3. **substitution** *of a node $w \in V_2$ for a node $v \in V_1$, denoted (v, w).*

Deletion and insertion of non-root nodes imply deletion and insertion of incident edges.

Definition 17 *Let $R_T \subseteq (V_1 \cup \{\lambda\}) \times (V_2 \cup \{\lambda\})$, where V_1 and V_2 are the vertex sets of two trees G_1 and G_2, be a relation. R_T is called a* **transformation** *from G_1 to G_2 if it has the following properties:*

1. $\{v \in V_1 | (v, w) \in R_T, w \in V_2 \cup \{\lambda\}\} = V_1$;
2. $\{w \in V_2 | (v, w) \in R_T, v \in V_1 \cup \{\lambda\}\} = V_2$;
3. $(v_1, w), (v_2, w) \in R_T \Rightarrow v_1 = v_2, \forall v_1, v_2 \in V_1 \cup \{\lambda\}$ *and* $w \in V_2$;
4. $(v, w_1), (v, w_2) \in R_T \Rightarrow w_1 = w_2, \forall w_1, w_2 \in V_2 \cup \{\lambda\}$ *and* $v \in V_1$.

We note that not any arbitrary sequence of elementary edit operations yields a meaningful transformation. For example, deletion and insertion are only allowed on leaf nodes, therefore they can only appear in a bottom-up order (according to a proper tree traversal). The set of "allowed" transformations on two trees is called **valid transformations** [155]. The **cost** of an elementary edit operation on trees $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a function $\gamma : V_1 \cup \{\lambda\} \times V_2 \cup \{\lambda\} \longrightarrow \mathbb{R}$ with the properties that γ is nonnegative definite, symmetric and fulfills the triangle inequality. The defined cost function is a *metric* and the space $(\{\Gamma\}, \gamma)$, with $\{\Gamma\}$ the set of trees, is a metric space. Consequently, the **cost of a transformation** $R_T \subseteq (V_1 \cup \{\lambda\}) \times (V_2 \cup \{\lambda\})$ of a tree $G_1 = (V_1, E_1)$ to a tree $G_2 = (V_2, E_2)$ is defined as $\gamma(R_T) = \sum_{(v,w) \in R_T} \gamma(v, w)$.

Finally, the edit distance between two trees is given as follows.

Definition 18 Edit distance. *Let $V_R^{1,2}$ be the set of valid transformations of a tree $G_1 = (V_1, E_1)$ to a tree $G_2 = (V_2, E_2)$. The edit distance between two trees G_1 and G_2 is defined as $\delta_{edit}(G_1, G_2) = min_{R_T \in V_R^{1,2}}\{\gamma(R_T)\}$.*

In the following subsection, we present a different approach to measuring distances between graphs, based on identifying identical sub-structures.

Graph Distance Based on the Maximal Common Subgraph

We will describe Bunke's graph distance, based on the maximal common subgraph of two graphs [20]. Bunke *et al.* observe that when defining similarity measures it is often desirable that they are metrics. Usually edit distance measures are metrics. Still, sometimes the metric properties are too restrictive with respect to the domain of application. An advantage of Bunke's distance in comparison to edit distances is that it does not depend on the cost of the underlying edit operations, which are often computationally problematic.

Definition 19 Bunke's graph distance. *Let $|G|$ denote the number of vertices in a graph G. The distance between two non-empty graphs G_1 and G_2 is defined as*

$$d(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{max(|G_1|, |G_2|)},$$

where mcs stands for maximal common subgraph and $|.|$ denotes graph cardinality.

Theorem 20 *The distance $d(G_1, G_2)$ introduced in definition 19 is a metric*[3].

Bunke's distance takes into account the ratio of the number of vertices in the maximal common subgraph of two graphs and the number of vertices in the larger of both. Note that this is not abusive and theorem 20 holds because the definition of maximal common subgraph itself (and the definition of common sub-graph in general) underlies the isomorphism property. An important implication of the defined metric is that the pairwise distance of a class of graphs to fixed graph might be quite small, still the distance between any two graphs not including the fixed one might be quite big. This can be problematic in terms of defining equivalence classes (or classes of similar graphs) by using Bunke's distance - a problem discussed in the concluding chapter of the thesis.

[3]A proof of the theorem is found in [20].

Relation between edit distance and maximum common subgraph isomorphism

A main result from Bunke discussed in [18] states that the problem of edit-distance-based graph matching and maximum common subgraph similarity are related in the following manner

$$d_{edit}(G_1, G_2) = |G_1| + |G_2| - 2|mcs(G_1, G_2)|. \tag{2.13}$$

This simple result shows that any algorithm for maximum common subgraph determination can be used for graph edit distance and the other way round.

However, we note that the result above should hold for edit distances defined on edit operations with equal costs. Otherwise, the formulation is not entirely accurate, for the cost function (defined previously) does not play a role in the expression (2.13).

Kernel Approaches

Kernel functions are widely used in machine learning in support of various classification and regression tasks. As a main body of the non-linear Support Vector Machines, kernels are discussed in more detail in Section 2.2. They are widely used for text categorization tasks, as well. In Section 2.4, we will expand on this particular application of kernels by referring to much of the contents of the current section.

The general intuition about kernel functions is that they measure the similarity between two data examples (not in isolation, but in the context of a given distribution). Geometrically one can think of a dot product of vectors which provides an example of a kernel.

But how can kernels be of use in measuring similarities on discrete structures, such as graphs and trees? In his survey paper, Gärtner introduces the so called group of syntax-driven kernels for structured data [53] – kernels which handle data that does not consist of mere flat attribute-value pairs, but has structure. We will start by defining the string kernel (introduced by Lohdi, Christianini and Shawe-Taylor in [94]) which underlies a commonly applied tree kernel-based similarity measure we will discuss afterwards. To these ends, we introduce the minimal necessary notation.

Let Σ be a finite set of *characters*, called an *alphabet.* A finite sequence $x = x_1x_2...x_m$ of characters from Σ is called a *string*, the empty string is denoted by ϵ. Let Σ^* denote the set of all non-empty strings defined over Σ. The number of occurrences of a string y in a string x is denoted as $num_y(x)$. A string kernel counts the occurrences of a string s in two input strings x and x' and is defined as

$$k(x, x') = \sum_{s \in \Sigma^*} num_s(x) num_s(x') w_s,$$

where w_s is a weight which is fixed with respect to given data or *a priori.*

In a conceptually similar manner Collins and Duffy [28] defined a tree kernel. Let G_1 and G_2 be two trees and let a subtree be defined as a connected subgraph of a tree such that either all or none of the children of a vertex is in the subgraph. Let $h_i(G)$ be the number of times the i-th subtree occurs in a tree G (some enumeration of the possible subtrees taken into consideration). The tree kernel is defined as

$$k(G_1, G_2) = \sum_i h_i(G_1) h_i(G_2).$$

To generalize, we will introduce a method by Haussler [69] for constructing kernels on strings, trees and graphs which can be applied iteratively to build a kernel on an infinite set from kernels. The family of kernels is known as convolution kernels, for the resulting kernel on given objects is the product of the kernels defined on their parts. Let $x \in X$ and $x' \in X$ be two objects which can be represented as vectors $\mathbf{x}, \mathbf{x}' \in X_1 \times X_2 \times ... \times X_D$, where X, X_i, $1 \leq i \leq D$ are non-empty, separable metric spaces. Let R be the relation $(X_1 \times X_2 \times ... \times X_D) \times X$ and let its decomposition be defined as $R^{-1}(x) = \{\mathbf{x} : R(\mathbf{x}, x)\}$. The convolution kernel is defined in the following manner:

$$k(x, x') = \sum_{\mathbf{x} \in R^{-1}(x), \mathbf{x}' \in R^{-1}(x')} \prod_{i=1}^{D} k_i(x_i, x'_i).$$

The graph matching problem with tree kernels has been addressed in research papers by various authors, we review some of the contributions. Basic definitions, algorithmic complexity proves and evaluations are provided by Smola [159] who also proves that the proposed tree kernel algorithm runs in linear time, thus avoiding dynamic programming with quadratic time complexity. Neuhaus and Bunke [113] propose a convolution graph kernel function, which is based on the graph edit distance and counts on evaluating on each step whether or not the edit paths in two input graphs are valid. The claim is that the method of the authors is more accurate than standard edit distance based methods and their kernel variants. Using Schur-Hadamard inner product in an SVM setting, combined with neural networks to avoid NP-completeness has been proposed and empirically tested by Geibel *et al.* [73]. Geibel, Kühnberger and co-workers suggested using variants of tree kernels for matching XML documents based on their DOM[4] trees [54, 55]. The authors propose to use the underlying structure of documents as an indicator of their "type" or "genre" in a document classification task; see, for more details, various publications by Mehler and co-workers, e.g. [103].

2.1.4 Basic Concepts in Lattice Theory

The mathematical theory of lattices has shown to be a useful model to represent concepts, their structure and relations [51]. A hierarchy of concepts is most intuitively represented by a directed rooted tree (defined in the previous section). However, the nodes of a tree are related by adjacency; it is only by intuition that the adjacency relation is considered to represent the subsumption of concepts within a hierarchy. Lattices, on the other hand, formalize ordered entities where order is a relation formally more similar to subsumption than adjacency.

Ordered Sets

Order is a property of a set of comparable entities. The theory of order formalizes this notion in terms of binary relations between elements of *ordered sets.*

[4]DOM stands for Document Object Model and is a structured representation of XML documents, defining their logical structure and ways of accessing and manipulating them.

We will start our introduction to order and lattice theory by defining a central concept – that of a partial order.

Definition 21 Partial Order. *Let P be a set and let x,y, $z \in P$. An order or a partial order on P is defined as a relation $\leq$, which has the following properties:*

$$x \leq x \qquad (reflexivity), \tag{2.14}$$
$$x \leq y \text{ and } y \leq x \text{ imply } x = y \quad (antisymmetry), \tag{2.15}$$
$$x \leq y \text{ and } y \leq z \text{ imply } x \leq z \qquad (transitivity). \tag{2.16}$$

A pair $(P, \leq)$ of a set and a partial order is called a **partially ordered set** (or a **poset** for short). Similarly to graphs, the notion of isomorphism is introduced for ordered sets, as well. Two ordered sets P and Q are said to be **isomorphic** (or "essentially the same") if there exists a bijective map ϕ, called **order-isomorphism** from P onto Q such that $\forall x, y \in P : x \leq y$ in P if and only if $\phi(x) \leq \phi(y)$ in Q.

A central property of ordered sets is the so called *duality principle*. It states that if a statement Φ is true for all ordered sets, so is the dual statement Φ^{δ}. A dual statement is acquired from a statement by substituting the partial order $\leq$ with $\geq$. For every ordered set P we can construct its dual P^{δ} by defining $x \leq x'$ to hold in P^{δ} if and only if $x' \leq x$ holds in P, for all $x, x' \in P$. The term *dual* and its derivatives will be used in the sequel with respect to the introduced duality principle.

An ordered set can be equipped with two special elements – a bottom and a top element. A **bottom element** of an ordered set P is an element $\bot \in P$ with the property that $\bot \leq x$, $\forall x \in P$. Dually, P's **top element** is an element $\top$ such that for all $x \in P$ it is true that $x \leq \top$.

Lattices and Complete Lattices

Introducing upper and lower bounds of elements of ordered sets gives rise to the definitions of a lattice, a complete lattice and a semi-lattice.

Let P be an ordered set and let $S \subseteq P$. An **upper bound** of S is defined as an element x of P such that for all $s \in S$, $s \leq x$ holds. Dually, a **lower bound** of S is an element $y \in P$ with the property that $t \geq y$, $\forall t \in S$. The set of all upper bounds of S is denoted by S^u and the set of all lower bounds of S is denoted by S^l. If S^u has a least element x, then x is called the **least upper bound** (or **supremum**, denoted sup$S$) of S, which is equivalent to the following two statements:

(1) x is an upper bound of S,
(2) $x \leq x'$ for all upper bounds x' of S.

In a dual manner, the **greatest lower bound** (or the **infimum**, denoted inf$S$) of S is defined as the greatest element of S^l (if it exists).

Now taking the whole set P we will make the following observations:

(1) If P has a top element, then sup$P = \top$. When P has no top element sup$P = \emptyset$.

(2) If P has a bottom element, then inf$P = \bot$. When P has no bottom element inf$P = \emptyset$.

As a matter of notation, the concepts of least upper and greatest lower bound are equally expressed in the following manner. The notation $x \vee x'$ stands for sup$\{x, x'\}$ and the notation $x \wedge x'$ stands for inf$\{x, x'\}$, read, respectively, as "*x join x′*" and "*x meet x′*". Similarly, we alternate the notations sup$S$ and $\bigvee S$, read *the join of S* and inf$S$ and $\bigwedge S$, read *the meet of S*.

We are now ready to give the definition of a lattice and a complete lattice.

Definition 22 Lattice. *Let P be a non-empty ordered set. If $x \vee x'$ and $x \wedge x'$ exist, $\forall x, x' \in P$, then P is called a lattice.*

Definition 23 Complete lattice. *Let P be a non-empty ordered set. If $\bigvee S$ and $\bigwedge S$ exist, $\forall S \subseteq P$, then P is called a complete lattice.*

Semi-lattices

We have defined a lattice as a partially ordered set closed under both binary operations supremum and infimum. Relaxing this condition by demanding that a partially ordered set is closed under only one of the two binary operations gives rise to the notion of **semi-lattice**.

Definition 24 Semi-lattice. *Let P be a set partially ordered by the binary relation $\leq$. Then $(P, \leq)$ is a* **join** *semi-lattice if for any two elements $x, x' \in P$ the least upper bound $sup(x, x')$ exists in P. Respectively, a* **meet** *semi-lattice is defined when for any two elements $x, x' \in P$ the greatest lower bound $inf(x, x)$ exists in P.*

Note that a semi-lattice can be trivially transformed into a lattice by adding a top element for a meet lattice and a bottom element for a join lattice.

2.2 Introduction to Support Vector Machines

The current section aims to acquaint the reader with the basic ideas of binary classifications with support vector machines. Before we proceed to that, we start with some words about machine learning, in general.

2.2.1 Computers and Learning

Machine learning finds an important place in solving a big variety of real life problems. It evolved from the rather philosophical question of whether and how could a machine be trained to *learn* from its "experience" and from the practical need for a machine to be able to do so. Naturally, answering this question and motivating further research in the area, was highly furthered by the development of electronic computers with high capacities and computational speed which helped to establish machine learning as a standard, nevertheless developing branch of Computer Science (CS) and Artificial Intelligence (AI).

One distinguishes between several different kinds of (machine) learning and we will briefly explain them.

Supervised Learning is about learning from a data set containing example pairs of inputs and outputs, referred to as *training data* or a *training set.* The basic assumption is that the set of input-output pairs contains an input-output relationship which the learning algorithm is to uncover and apply as an input to output mapping function on unseen data. **Unsupervised Learning** is applied in the case when no outputs are initially available and the learning algorithm acquires some knowledge of how and under which dependencies the data have been generated. In **Semi-supervised Learning** (also and less commonly referred to as **Query Learning**), the machine learner has the ability of learning various tasks by querying about the output corresponding to a given input. **Reinforcement Learning**, discussed and applied broadly in AI, develops decision mechanisms of taking an action out of a set of actions in order to move from one state to a predefined goal state, the decision being a function of the current state [30].

Depending on the output, the learning problem is characterized as *binary classification* in the case when the outputs are binary values, *multiple classification* when the outputs take their values from a number of finite classes (or categories), and *regression* – in the case when outputs have real values.

In the next section, we will overview the Support Vector Machines, a prominent classification method, which will play a central role in our ontology matching approaches, providing a technique for selecting characteristic variables.

2.2.2 Overview of Support Vector Machines

The Support Vector Machines are supervised learning classification techniques introduced in the mid 1990s by Vladimir Vapnik and coworkers. The method has proved to perform very well in practice for both binary and multiple classification as well as regression tasks on large data sets in diverse application areas. The mathematical machinery behind the SVMs reposes on theoretical grounds in four different mathematical fields – Optimization Theory, Reproducing Kernel Hilbert Spaces [4], Linear Classification [132], and Vapink and Chervonenkis Generalization Theory [157]. For reasons of space, we cannot give a detailed account of all aspects. Instead, we will provide enough knowledge about SVMs in order to understand the ideas behind SVM-based variable selection approaches developed in the past decade, as well as to be able to introduce our method. For thorough introductions to SVMs, we refer to the books of Cristianini *et al.* [30] and Vapnik [157], as well as the papers of Massart *et al.* [14] (discussing the statistical performance of SVMs), and Burges [21] (providing a brief but exhaustive overview of the method). We will broadly follow the notations used by Vapnik, Cristianini and Burges.

Generalization Properties

A supervised machine learning task usually takes place in two main phases: first, a *training phase*, when the machine learner is given data to infer a classification, regression or, generally, a decision rule from and, secondly, a *test phase*, when the learner is confronted with unseen data examples to apply (hopefully correctly) the decision rule on.

The ability to generalize on unseen data, after having learned the training data is essential and central to every learning machine. The following quotation from Burges gives an insight on that [21]:

> 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 botanists lazy brother, who declares that if its green, its a tree.

The conditions under which a learner has sufficiently good generalization properties so that it does not behave like the botanist with photographic memory, but is also sufficiently precise in its judgments, so that it does not behave like the botanist's lazy brother have to be studied and defined.

Let us consider the following binary classification layout. Assume we have l observations $\mathbf{x}_i \in \mathbb{R}^n$ and their associated trusted "truth" $y_i \in \{-1, 1\}$. Data are assumed to be i.i.d. (independent and identically distributed), drawn from an unknown probability distribution $P(\mathbf{x}, y)$. The goal of binary classification is to "learn" the mapping $\mathbf{x}_i \to y_i$ which is consistent with the given examples. Let $\{f(\mathbf{x}, \alpha)\}$ be a set of such possible mappings, where α denotes a set of parameters. Such a mapping is called a classifier and it is deterministic – for a certain choice of $\mathbf{x}$ and α it will always give the same output f.

Since the data are drawn from a distribution P, the natural measure of error would be the probability that a random instance is misclassified. The **actual risk**, or the expectation of the test error for such a learning machine is

$$R(\alpha) = \int \frac{1}{2}|y - f(\mathbf{x}, \alpha)|dP(\mathbf{x}, y). \tag{2.17}$$

The quantity $1/2|y - f(\mathbf{x}, \alpha)|$ is called *loss*. Based on a finite number of observations, we calculate the **empirical risk**

$$R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(\mathbf{x}_i, \alpha)|, \tag{2.18}$$

which is a fixed number for a given training set $\{\mathbf{x}_i, y_i\}$ and a certain choice of parameters α.

For losses taking values 0 or 1, with probability $1-\eta$, $0 \leq \eta \leq 1$, the following bound on the actual risk holds:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h\log(\frac{2l}{h}) + 1 - \log(\frac{\eta}{4})}{l}}, \tag{2.19}$$

where h is a nonnegative integer which will play a core role in our variable selection procedure, called the *VC dimension*, measuring the capacity of the classifier[5]. We note that the presented risk bound does not depend on $P(\mathbf{x}, y)$ and it can be easily computed provided the knowledge of h. We will briefly introduce this parameter.

Let us consider the set of functions $\{f(\mathbf{x}, \alpha)\}$ with $f(\mathbf{x}, \alpha) \in \{-1, 1\}, \forall \mathbf{x}, \alpha$. In a binary classification task there are 2^l possible ways of labeling a set of

[5] *VC* stands for the initials of Vapnik and Cherovonenkis who introduced the theory in the 1980s.

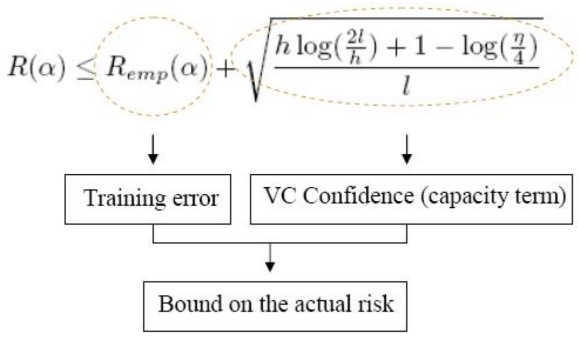

$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h\log(\frac{2l}{h}) + 1 - \log(\frac{\eta}{4})}{l}}$
Training error
VC Confidence (capacity term)
Bound on the actual risk

Figure 2.3: Structural risk minimization I.

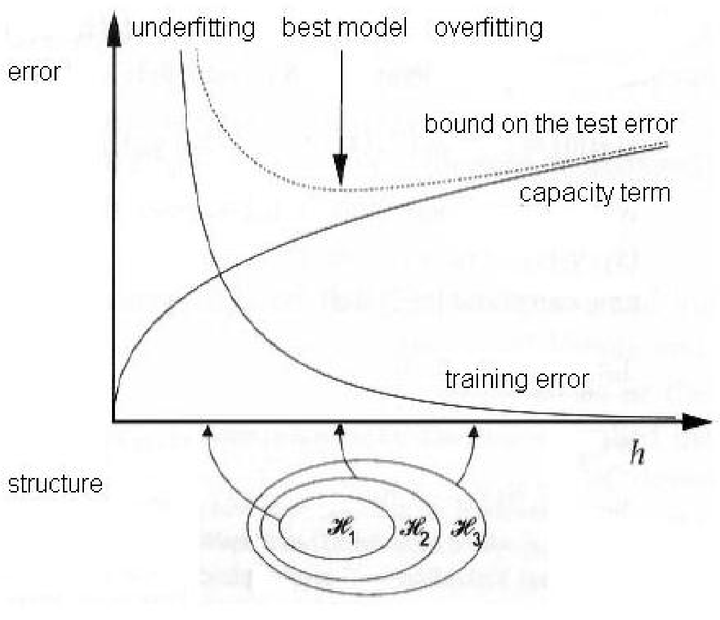


Figure 2.4: Structural risk minimization II.

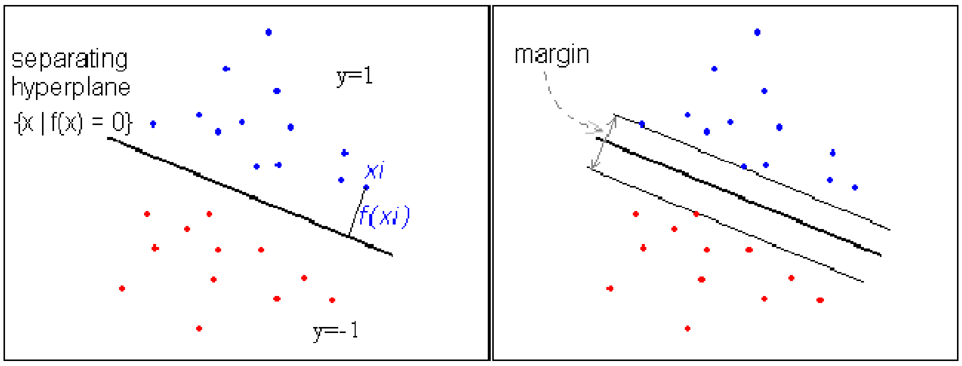


Figure 2.5: A separating hyperplane and a maximal margin in $\mathbb{R}^2$.

l points. If for each labeling there can be found a member of $\{f(\alpha)\}$ which correctly assigns these labels, we say that the given set of points is *shattered* by the given set of functions. The VC dimension is a property of such a family of functions, which is defined as the maximum number of training points that can be shattered by that family.

The bound (2.19) gives an insight on one very important aspect of statistical learning. The term $\sqrt{\frac{hlog(\frac{2l}{h})+1-log(\frac{\eta}{4})}{l}}$, called *VC confidence* is "responsible" for the *capacity* of the hypothesis space (including hypotheses with the ability to learn unseen data without an error). The other right-hand quantity of (2.19) – the empirical risk, measures the *accuracy* attained on the particular training set $\{\mathbf{x}_i, y_i\}$. What is sought for is a function which minimizes the bound on the actual risk and thus provides a good balance between capacity and accuracy – a problem known in the literature under different names such as bias variance tradeoff, overfitting or capacity control. Structural risk minimization, sketched in Figure 2.3 and Figure 2.4, provides a theory to do so. The set of functions is divided into nested subsets ordered by VC-dimension. The one which assures the lowest bound on the risk functional is picked out.

Linear Classification

We come back to binary classification with support vector machines. SVMs are based on a family of linear functions $\{f(\mathbf{x}, \alpha)\}$ mapping elements from the input space to a binary output, as introduced so far with α being the parameters of the linear function $f(\mathbf{x})$. The classification decision is according to the sign of the linear function at the point to be mapped. Geometrically, it can be thought of as a hyperplane separating the space of the inputs in two halves in a way that the margin between the two classes is maximized (shown in Figure 2.5).

More formally, let us consider the input space $X \subseteq \mathbb{R}^n$ and the output domain $Y = \{-1, 1\}$ with a training set $S = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_l, y_l)) \in (X, Y)^l$. SVM is a linear real function $f : X \rightarrow \mathbb{R}$ with

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b,$$

where $\alpha = (\mathbf{w}, b) \in \mathbb{R}^n \times \mathbb{R}$ and $\langle \cdot, \cdot \rangle$ denotes the inner product in a vector space (a dot product in an Euclidean space). The separating hyperplane in the input space X is defined by the set $\{\mathbf{x}|f(\mathbf{x}) = 0\}$. The decision rule assigns an input vector $\mathbf{x}$ positive if and only if $f(\mathbf{x}) \geq 0$ and negative – otherwise. (The inclusion of 0 in the first case and not in the second is conventional.)

We are looking for the best decision function $f(\mathbf{x})$ which separates the input space in a way that the distance between the positive and negative examples closest to the hyperplane is maximized. The parameters of the desired function are found by solving the following quadratic optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \frac{1}{2}\|\mathbf{w}\|^2$$

under the linear constraints

$$\forall i = 1, ..., n,\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) + b) \geq 1.$$

For computational reasons the problem will be solved in a Lagrangian setting. This is useful for two main reasons: first, the constraints will be replaced by constraints on the Lagrangian multipliers which are easier to handle and, second, the training data points will appear only in the form of dot products which, we shall see, is useful for generalizing to the nonlinearly separable case.

After that argumentation, the original problem is transformed to minimizing the Lagrangian functional

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i(\langle \mathbf{x_i}, \mathbf{w} \rangle + b) + \sum_{i=1}^{l} \alpha_i \tag{2.20}$$

with respect to $\mathbf{w}$ and b, where α_i are non-negative Lagrangian multipliers $(i = 1, ..., l)$. However, minimizing 2.20 is equal to solving the dual Lagrangian problem – maximize

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l} \alpha_i \alpha_j y_i y_j(\langle \mathbf{x_i}, \mathbf{x_j} \rangle) \tag{2.21}$$

with respect to the Lagrangian multipliers. This follows directly from setting the derivatives of L_P with respect to $\mathbf{w}$ and b to zero, yielding

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x_i}$$

and

$$\sum_i \alpha_i y_i = 0.$$

A solution $\mathbf{w}$, b of the initial optimization problem must satisfy the following necessary and sufficient conditions (known as Karush-Kuhn-Tucker (KKT) conditions):

$$\begin{aligned}
\frac{\partial}{\partial w_v} L_P &= 0,\ v = 1, ..., n && (2.22)\\
\frac{\partial}{\partial b} L_P &= 0 && (2.23)\\
y_i(\langle \mathbf{x_i}, \mathbf{w} \rangle + b) &\geq 0,\ i = 1, ..., l && (2.24)\\
\alpha_i &\geq 0,\ \forall i && (2.25)\\
\alpha_i(y_i(\langle \mathbf{x_i}, \mathbf{w} \rangle + b) - 1) &= 0,\ \forall i. && (2.26)
\end{aligned}$$

A bound on the VC-dimension

The VC-dimension, discussed in the beginning, is going to play an important role in our SVMs-based variable selection procedure, to be presented in Chapter 4. In general, it is difficult to compute the VC-dimension directly, but we can compute an upper bound for it, depending on the weight vector $\mathbf{w}$ and on properties of the data.

Consider hyperplanes of the kind $\{\mathbf{x}|f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}\rangle + b = 0\}$. The following lemma (Vapnik [157]) provides an upper bound on the VC-dimension.

Lemma 25 *If all example vectors $\mathbf{x_i}$ are contained in a ball of radius R and for all exmples it holds*

$$|\langle \mathbf{w}, \mathbf{x}\rangle + b| \geq 1,$$

then the set of hypotheses $\{f(\mathbf{x})\}$ has a VC-dimension h bounded by

$$h \leq \min(R^2\|\mathbf{w}\|^2, l) + 1.$$

SVMs with Kernels

Figure 2.6 summarizes the steps undertaken towards finding a solution in the linearly separabel case. The power of SVMs is that even if data are not linearly separable in the input space, they can be mapped into a (possibly higher dimensional) space, called *feature space* where a linear boundary between both classes can be found (cf. Figure 2.7 for an illustration). The mapping is done by the help of the transformation

$$\phi : X \longrightarrow F = \{\phi(\mathbf{x})|\mathbf{x} \in X\} \subseteq \mathbb{R}^N,$$

where N is a positive integer or infinity. The set of linear hypotheses in the feature space is constructed in the same way as in the original input space,

$$f(x) = \langle \mathbf{w} \cdot \phi(\mathbf{x})\rangle + b = \sum_{i=1}^{N} \mathbf{w}_i \phi_i(\mathbf{x}) + b. \tag{2.27}$$

We come to the final step which makes SVM so easy to apply. As we have seen, in Lagrangian theory, (2.27) has the dual representation

$$\sum_{i=1}^{l} \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x})\rangle + b. \tag{2.28}$$

In the latter formulation, the nonlinear transformations of the original data ϕ appear only in the form of an inner product which can be replaced by a kernel function with the property

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z})\rangle.$$

In result, we do not need the explicit knowledge of the (possibly nonlinear) transformation ϕ, nor the resulting (possibly higher dimensional) feature space. In the input space we will use a kernel function which acts as a dot product in the feature space. The formal algorithm of finding the decision function is similar to the one from the linear case with the difference that the dot products are

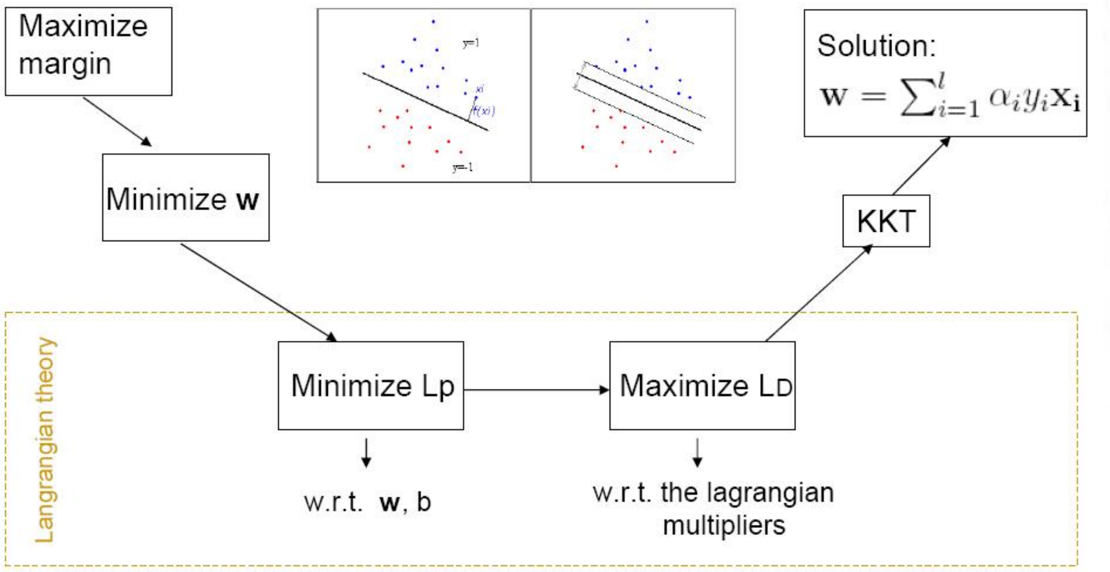


Figure 2.6: SVMs: The linearly separable case.

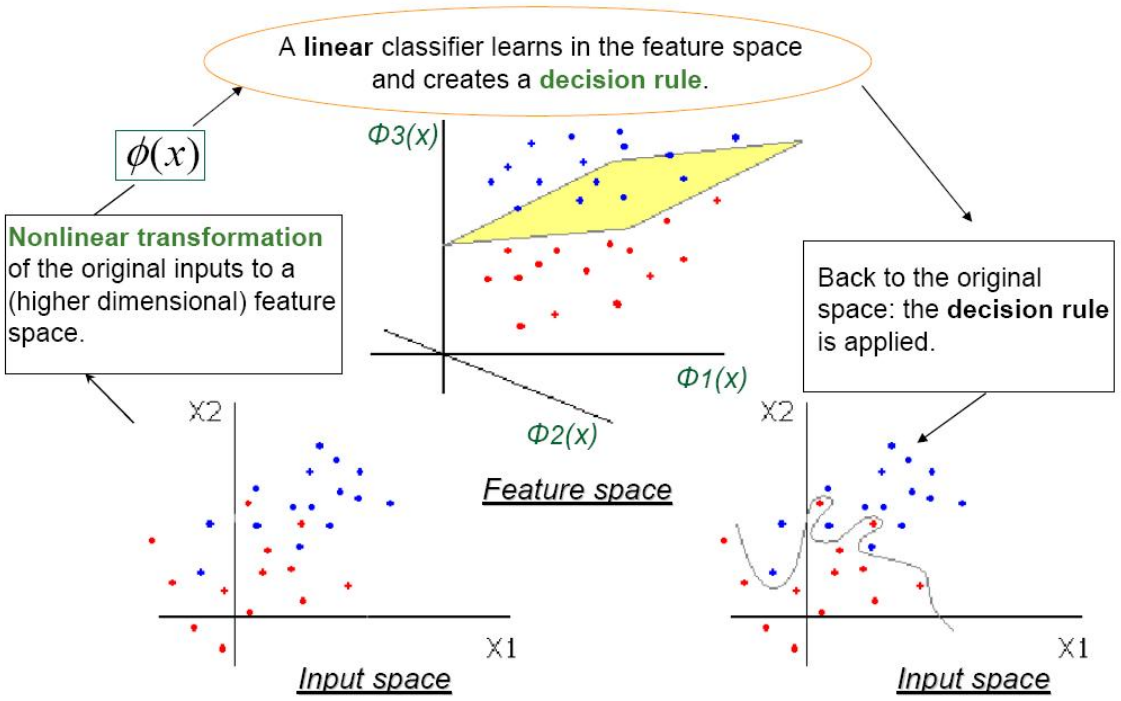

A **linear** classifier learns in the feature space and creates a **decision rule**.
$\phi(x)$
Nonlinear transformation of the original inputs to a (higher dimensional) feature space.
Φ3(x)
Φ1(x)
Φ2(x)
Feature space
Back to the original space: the **decision rule** is applied.
X2
X1
Input space
X2
X1
Input space

Figure 2.7: SVMs: The linearly non-separable case.

replaced by that kernel function. The expressiveness of the learning machines is increased a lot by the introduction of kernels – the underlying linearity of the classifiers remains intact which preserves their computational tractability.

The necessary and sufficient conditions for a function, defined in the original space to be an inner product in the feature space are given by Mercer's theorem [21], [30]. The simple condition of Mercer's theorem is that the kernel must be a positive semi-definite function.

There are many plausible choices of a kernel function which have already proved to be quite efficient in practical applications. Among many popular choices, we will outline the polynomial and the radial basis function kernels given by:

$$K(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^p \qquad (polynomial) \tag{2.29}$$

$$K(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x}-\mathbf{y}\|^2} \qquad (radial\ basis\ function) \tag{2.30}$$

Note that each kernel function depends on one or a set of parameters. In the examples above these are the degree d or the parameter γ. Picking up the right kernel and then setting its parameters appropriately is a difficult task (see research on model and parameter selection, e.g. [30], [9]), usually done manually by using heuristics or methods like cross-validation [30].

Finally, kernels which do not operate on vector spaces and therefore do not fulfill Mercer's condition have also been introduced to deal with data which are not representable in a vector space (or its representation is costly and inefficient), like strings and graphs. Such examples, known as kernels for structures, are considered in Sections 2.1.3 and 2.4 and the sources cited there.

2.3 Variable Selection

Variable selection is a basic problem in a number of real life statistical analysis, classification and regression tasks. Applying a variable selection procedure to a learning task has many benefits, such as enhancing descriptiveness (it can improve the visualization of the data and the understanding of hidden relations), achieving better generalization and / or gaining computational power. The task of variable selection is to evaluate with respect to some predefined criteria the importance of a variable (or a block of variables) for the outcome of a given machine learning task, should that be prediction, classification or a simpler data analysis problem. The result of a variable selection procedure can be a list of the input variables ordered by *importance* or *informativeness*. On the one hand, this helps to reduce the dimension of the input space by eliminating uninformative, noisy or redundant variables. In that way we ensure better computational efficiency and improve generalization. On the other hand, in various domains of application, where the learning tasks are of supervised nature, such as process control, forecasting, text categorization or gene selection it is important to find out more about the input – output relations that hold in a dataset by pointing out the input variables, which most strongly affect the response. In a classification task, it provides mechanisms for ranking the variables with respect to their discriminative power. In our study, we are interested in classification tasks for text categorization and the focus falls on the latter application of variable selection.

We make a distinction between the terms "variable" and "feature", both used (sometimes confusingly) to denote similar but distinct entities in statistical and machine learning terminology. The term "variable" is most commonly used for the original input variables, while the term "feature" stands for variables constructed from the input "variables". Both terms are often used in an substitutable manner; a disambiguation is, however, needed when the features are implicitly computed from the variables like, for instance, in kernel-based methods.

One of the contributions of this dissertation is the definition and application of a novel variable selection criterion, based on Support Vector Machines, which will be introduced in Section 4.3.3. For an overview of standard variable selection applications and existing theoretical approaches we refer to the study of Guyon and Elisseeff [63]. Variable subset selection methods for text-learning have been discussed and evaluated in [109]. A part of the next section of this thesis is also dedicated to selection techniques for text categorization, since they are of particular importance for some of the main results. SVM-based methods, which are directly relevant to our approach, are discussed separately in Section 2.3.2.

2.3.1 Standard Variable Selection Techniques

According to the typology of variable selection techniques brought out by Guyon and Elisseeff [63], one distinguishes between *variable ranking*, *variable subset selection* and *space dimensionality reduction by feature construction*. We outline the main characteristics of each of these methods.

Variable ranking evaluates the contribution of single variables by assigning to each the value of a scoring function. The scoring function is calculated from the data. It can be based on correlation criteria (estimating the correlation between a variable and the output), or information theoretic criteria (such as an estimation of the mutual information between a variable and an output). Equally, the predictive power of an individual variable can be evaluated by using as a criterion the performance of some classifier built with this variable only.

However, ranking variables individually leads to neglecting the informative power of variables taken in groups. Therefore the second important group of techniques unites ***variable subset selection*** methods. The selection of groups of variables can be done in three different manners, where from one can further distinguish between three types of subset selection techniques: *wrappers*, *filters* and *embedded methods*. The difference between the three classes of methods is in their relation to the machine learning process. Wrappers use the machine learner as a "tool" to score groups of variables according to their predictive power [86]. For that reason they are not specific to a given learning machine. Filters, on the other hand operate on a dataset before introducing the machine learner – the variable subset selection is a pre-processing mechanism. Finally, embedded methods act simultaneously with the learning machine by adjusting the scores of the subsets in the time of training. Figure 2.8 sketches the temporal relation of the three methods with the training phase in a machine learning task.

One of the main goals of variable selection is to improve efficiency by reducing dimensionality and hence complexity. The third important class of selection techniques are based on ***space dimensionality reduction*** by constructing

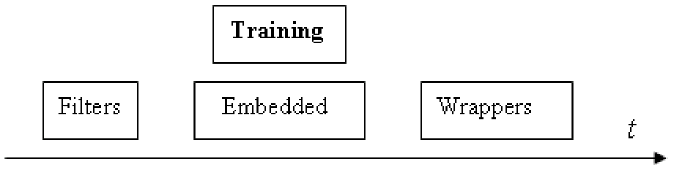


Figure 2.8: Selection techniques in relation to training in time.

features (principal axes) out of the input variables and projecting the data onto the newly constructed feature space. The most pertinent variables are those which contribute most to the construction of the new principle axes.

Clustering is one of the most prominent feature construction techniques. The main idea is to find clusters of similar variables among the set of original input variables and replace each cluster by a feature which is the cluster's centroid.

Singular value decomposition (SVD) is another feature construction method originating from factor analysis and matrix factorization. It is a fact that in many learning problems where hundreds of thousand of variables are available, a much smaller number of them are linearly independent. SVD-based techniques make use of that fact in order to construct new features as linear combinations of the input variables by performing SVD on the matrix containing as rows the observations[6] and as columns – the input variables.

Latent Semantic Analysis, discussed in Section 2.4 of this chapter is based on that technique. Other related approaches are two of the most common descriptive statistics methods – the Principle Component Analysis and the Discriminant Analysis, which on their turn have been dedicated some more space to later on, in Section 4.3.2. Both PCA and DA project the input data on principle axes which are constructed by linear combinations of the input variables. Naturally, different linear combinations, i.e. assigning different weights to the input variables, corresponds to constructing different axes. In our approach, what we are interested in both PCA and DA analyses is the way that classes are represented and separated in a projection over one or two principle axes. For that reason, the variables which contribute at most for the construction of these axes are those that are most important for the separation of the instances projected over these axes. These variables are also said to best discriminate between the classes.

2.3.2 Variable Selection for SVMs

The SVMs have many attractive sides – their performance does not depend on the distribution of the data safe that they are i.i.d., it does not demand a linear input-output relation and they are computationally advantaged over competing methods. The decision function is calculated only on a small number of input points (the support vectors). At least theoretically, the generalization properties of SVMs do not dependent on the size of the input space which makes variable selection little prominent for learning with SVMs. However, the listed properties turn them into a good candidate for a variable selection tool to be used self-dependently. In addition to that, some authors have shown that even though

[6]The term "observations" is common for denoting the examples (or instances) in a dataset.

theoretically unnecessary, variable selection improves SVM learning in practice [64, 126].

SVM-based variable selection has been studied in the past couple of years, although, to the best of our knowledge, with surprisingly little intensity. We briefly review the most important related endeavors.

Guyon *et al.* (2000) proposed the SVM-RFE (standing for Recursive Feature Elimination) algorithm [64] for selecting genes which are relevant for cancer classification. The algorithm selects r out of l initial variables with $r < l$ based on a backward sequential selection. The removal criterion for a given variable is minimizing the variation of the weight vector $\| \mathbf{w} \|^2$, i.e. its sensitivity with respect to a variable.

A method based on finding the variables which minimize bounds on the *leave-one-out* error for classification was introduced by Weston *et al.* (2000) [163]. The *leave-one-out* is a bound on the actual risk, alternative to the one introduced in Section 2.2, based on the support vectors. To get an estimate of the bound, for all training points the following procedure is applied: a training point is removed, SVMs are re-trained over the data without the removed point and tested on it. The expectation of error over all training sets with one point left out gives us the desired bound. The bound is intuitive and quite elegant and formally it looks as follows:

$$E[R_{l-1}] \leq \frac{E[N_{SV}]}{l}, \tag{2.31}$$

where R_{l-1} is the risk of a machine trained over $l-1$ observations, $E[R_{l-1}]$ is the expectation of that risk for all possible choices of sets of size $l-1$, N_{SV} denotes the number of support vectors over a set of $l-1$ observations (its expectation is taken over all possible such sets). Weston *et al.* tested their criterion on toy data, artificially generated for the experiments and on real-life data for face detection, pedestrian recognition and cancer morphology recognition. The performance of the selection method has been evaluated by comparing the generalization error in a classification task when using the selected features only and when using the whole set of features which includes noise by construction (in the toy data), or by nature (in the real-life problems).

Rakotomamonjy *et al.* (2004) suggest a variable selection procedure for SVMs based on the sensitivity of the margin according to a variable. The guiding heuristics of their approach is:

> A variable which is little informative and thus little important for the decision function, is a variable to which the margin $2/\| \mathbf{w} \|$ is little sensitive [126, 127].

To demonstrate the viability of their selection method, the authors replicated the experiments on toy data from Weston *et al.* and carried out additional experiments for pedestrian recognition, outperforming the results reported by Weston and colleagues in [163].

Finally, we will cite one last contribution by Bi *et al.* (2003), who developed the VS-SSVM variable selection method for regression tasks applied to molecules bio-activity prediction problems [13]. The suggested variable selection method is a wrapper technique: it uses the performance of the learner with respect to a group of selected variables as a criterion of their importance. Bi *et al.* point out that, although successful in induction tasks, wrapper methods can be computationally very expensive on a large number of variables. To reduce computational costs, the variable selection is performed by the help of linear SVMs as a preprocessing step to induction tasks with non-linear learning machines.

In Section 4.3.4, we will present our own contribution to the variable selection research for SVMs, based on variations of the VC dimension of the classifiers. The current section will close by an introduction to *text categorization* – a field of machine learning, in which a central part of the solutions proposed in this thesis is grounded.

2.4 Text Categorization

Text categorization is a machine learning classification task which takes as input data a collection of text documents written in some natural language. This could be the set of all web-pages in a web directory, the emails that a certain person has received during the last year, etc. Assume that all these text documents are grouped together in classes of some kind – the categories of the web directory, "spam" vs. "not spam" in the email example. The task of text categorization consists in assigning the correct class label to an unseen text document, given a dataset consisting of correctly classified documents. Automatic filtering of a large amount of text documents is useful for presenting to the user only the ones which are of importance to her or the ones which correspond to a given query.

Instance-based approaches to modeling and comparing ontologies often rely on text categorization techniques. An ontological concept is referred to by a word, a set of words or a set of documents and can be viewed as a category unifying entities which are represented as or appear in text documents. The methods for instance-based concept alignment developed in this thesis (cf. Section 4.3) apply different text classification techniques, as well, and therefore, in the following paragraphs, we will describe several ground notions and approaches from this field of machine learning and natural language processing. Many of the topics discussed below are covered thoroughly in the book by Manning and Schütze [98].

2.4.1 Representing Text Documents as Feature Vectors

In order to be able to perform classification or other machine learning tasks on text documents, we need to render them in a form, which is a suitable input of a computer program and allows this program to run efficiently in terms of time and space complexity.

In the current section, we will introduce a standard approach to represent a text document as a vector in an n-dimensional feature space, based on the frequency of the appearances of *word stems* within the document and known as the *TF/IDF* vector model. A *word stem* is a form of a word which results from removing the information relevant to flection and case from the word [123] (a processes known in computing as *lemmatisation*). That method renders words which may look intuitively the same for humans because they share a morphological root, but appear different to computers, into one single "word"

– for example, the words “plays”, “played” and “playing” are represented by a single word stem “play”.

In the following sections, we will not distinguish between the terms “word”, “word stem” and “term”.

In order to additionally simplify the task of representing human-produced text in a numerical format, often a certain kind of pretreatment of the input text data is required. This most commonly consists in removing so called *stop-words* (topic-neutral words which appear too often in the text corpus), such as definite and indefinite articles, conjunctions and propositions (“or”, “but”, “as”, “the”, “a”, etc.), as well as words which appear too few times in a single text or in the whole corpus. The collection of terms of the corpus which are retained and which will be used to describe a single document will be called *a vocabulary.*

The representational standard *TF/IDF*, a short for Term Frequency / Inversed Document Frequency, consists in coding a text document by an $n-$dimensional feature vector, where the value of each feature is relative to the frequency of the appearance of a word within the text document, scaled by the number of documents in which the word appears within the whole document data base. The number n of words to be considered depends on the particular application and on the user’s choice – in the most specific and expensive case n equals the number of all words encountered in a training corpus.

We will give a formal definition of the *TF/IDF* model starting by introducing some notation. Let the number of times a word w_i appears in a text document $\mathbf{d}$ be denoted by $TF(w_i, \mathbf{d})$. Clearly, a document $\mathbf{d}$ containing n words can be coded by an n-dimensional *TF*-vector by calculating the values of the $TF(w_i, \mathbf{d})$ features for all $i = 1, ..., n$. However, the performance of algorithms using this representation is improved a lot by scaling each of the dimensions of the vector by the so called inversed document frequency, or *IDF* for short. Let the number of documents in which a word w_i occur within the whole text document dataset be $DF(w_i)$. The inverse document frequency of a word, given that m is the number of documents in the training set, is calculated by

$$IDF(w_i) = log(\frac{m}{DF(w_i)}). \tag{2.32}$$

Finally, the *TF/IDF* feature vector corresponding to a document $\mathbf{d}$ on n words looks like that:

$$\mathbf{d} = (TF(w_1, d)IDF(w_1), TF(w_2, d)IDF(w_2), ..., TF(w_n, d)IDF(w_n)).$$

Clearly, due to the fact that each document can contain a different number of words and in order to be able to project all the vectors onto one single vector space where distances can be measured, a normalization of the length of the document vectors is needed, before proceeding to the classification task. Most of the vectors will remain sparse since most of the documents in a training dataset will not contain all the words from the vocabulary.

In the rest of the chapter and of the thesis, we will predominantly consider the *TF/IDF* document representation model, except where stated otherwise. We mention, for completeness, that this is not the only possible text representation. Simpler presentations include collecting the raw counts of the term occurrences in the documents. More complex methods, introduced in [106], consist in defining an attribute for each word position in a text document whose value is the actual natural language word which appears at this particular position. This method of text representation is commonly applied in Bayesian learning text classification, discussed below.

2.4.2 Dimensionality Reduction

As already mentioned above, since words are used as features, the document vectors might become of a prohibitively large dimension, reaching hundreds of thousands of terms. This can lead to a situation in which the essential information is becoming hard to extract and the computational efficiency of the applied learning algorithms is reduced. As it has been observed by Goller [57], dimensionality reduction is also useful in order to avoid overfitting, even for Support Vector Machines.

Text categorization is an inductive learning task and all of the feature selection techniques discussed from a general viewpoint in Section 2.3 can be applied. In the current section, we will outline a few (automatic) techniques for dimensionality reduction applied *particularly* for learning and classifying text. A thorough evaluation of some of the selection techniques on large categorization problems is presented by Mladenic [109] and Yang [167, 168].

One of the most commonly applied selection methods is **information gain** [106]. The method suggests an information theoretic approach to evaluate the goodness of a term with respect to a set of categories. It is based on measuring the number of bits of information obtained for a given category before and after the removal of a certain term.

Another frequently used method tightly related to information gain is **mutual information** in which one considers the co-occurrences of a term and a category [26]. We will present a variation of it, known as the **point-wise mutual information**. Let t be a term and c – a category and let A be the number of times t and c co-occur, B – the number of times t occurs alone, C – the number of times c occurs alone and m – the total number of documents in the dataset. The mutual information criterion is estimated by

$$I(t,c) = log \frac{A \times m}{(A+C) \times (A+B)}. \qquad (2.33)$$

A related approach uses the χ^2 **- statistics** by testing the lack of independence between t and c [166]. The criterion is estimated by

$$\chi^2(t,c) = \frac{m \times (AN - CB)^2}{(A+C) \times (B+N) \times (A+B) \times (C+N)},$$

where N is the number of times neither t, nor c occurs. Based on the χ^2 - statistics, Sebastiani and co-workers proposed the **GSS coefficient** given simply by the difference $AN - CB$ [136].

The criteria discussed so far are similar in that they look into the information about the term-category associations. A second group of methods consists of criteria which are not task-specific in that sense.

The number of documents in which a term occurs is clearly another straightforward criterion for the term's importance. **Document frequency thresholding** is a simple feature selection technique, which is based on the document frequency for each term in the training dataset. Terms with an infrequent occurrence are considered as unimportant for the regrouping of documents into categories.

The importance of a single term can also be evaluated by the so called **term strength** criterion [164]. The quantity that this technique measures is the likeliness of a term to appear in documents that are assumed to be closely related. The relatedness of documents is measured by a certain similarity measure (usually the cosine value, discussed in the next section).

Based on an experimental evaluation of the feature selecting techniques for text categorization tasks discussed above, Yang and Pederson [168] reported that the Information Gain and the Chi-square methods prove to be most efficient. Sebastiani [136] indicates GSS as the most efficient of the proposed techniques. However, the efficiency of one method or another remains a question of a particular application and used data. In our study, we apply feature selection for different ends than the ones suggested by the authors of [168, 136] – instead of dimensionality reduction, we aim at evaluating the *predictiveness* of a small set of features with respect to a class. We will see in the empirical evaluation of our results (Section 5) that for that goals different selection techniques show to be more appropriate.

Feature selection in text-learning can be also based on descriptive statistical methods, such as Principal Components Analysis and Discriminant Analysis (discussed in Section 4.3 and in [77, 48]) and the related Latent Semantic Analysis (LSA). These methods differ from the ones cited above in that they construct a set of new features (fewer than the original number of features) by regrouping together semantically similar terms and forming optimal dimensions. We will describe LSA in some more detail below.

Latent Semantic Analysis is a technique of major importance in natural language processing (NLP), introduced by Landauer and co-workers in the late 1980s [91]. It is Landauer's claim that LSA is also a cognitively plausible model of the way humans representation of meaning reflects the words they read and hear. Leaving aside the discussion of the cognitive aspects of the method, we focus on the fact that LSA is broadly applied in text categorization as well as in general information retrieval, finding relations between terms, cross language retrieval and other NLP tasks.

An old idea in NLP, and one we discussed in details above, is that documents can be represented as vectors in a space, which is constructed by assigning an orthogonal direction to every term. This leads to the creation of spaces of hundreds of thousands of dimensions. LSA evolves from the intuition that such a high dimensionality is useless, based on the heuristics that there are many hidden (latent) relations between the terms. For example, the terms *car* and *automobile* are likely to appear in similar documents, just as *guitar* and *piano*, etc. For that reason, documents as vectors in a high dimensional space where each dimension is a single term will not "occupy" all possible regions of this space, but will be grouped in small subspaces of it. Based on that finding, LSA constructs a lower dimensional concept space by combining related terms in a single dimension.

LSA starts by representing a text corpora as a matrix in which, standardly, each row stands for a term and each column – for a text document. Usually, documents are coded as *TF/IDF* vectors in which every single element of the matrix contains a number relevant to the frequency with which a term appears in a document. Let the term-document matrix be A. In mathematical terms, the discussion about term relatedness from the previous paragraph can be summed up to the following statement about the matrix A: *the rank of A is much lower than* $\min(n, m)$. (Remember that the rank of a matrix is the number of linearly independent rows or columns and is at most equal to the smaller of the numbers of rows and columns [58].)

As a second step, LSA performs a singular value decomposition (SVD) on A. SVD, discussed also in [58], represents a rectangular matrix as a product of three matrices: one describing the original row entities as vectors of derived orthogonal factor values, another describing the original column entities in the same way, and a third one which is a diagonal matrix containing scaling values. The multiplication of the three matrices results in the original rectangular matrix:

$$A_{n\times m} = U_{n\times r}\Sigma_{r\times r}V^T_{r\times m}, \tag{2.34}$$

where r is the rank of A ($r \leq \min(n, m)$) and U and V are column orthogonal matrices ($U^TU = V^TV = \mathbf{I}$, where $\mathbf{I}$ stands for the identity matrix).

By using this decomposition, terms and documents are translated into a common concept space of dimension r lower than n (the original number of terms). Every row i of the matrix U can be viewed as a refined representation of a term i and every row j of V can be viewed as a refined representation of a document j, which results from the orthogonal factoring of the rows and columns of A. Since terms and documents can be equally viewed as vectors living in an r-dimensional space, the document-document, term-term and term-document similarities can be evaluated by the help of a distance metric or a similarity measure of the user's choice – usually the cosine between vectors (cf. next sub-section).

In order to relieve computational complexity, only the k ($k << r$, $k \approx 300$) largest values on the diagonal of Σ are taken, together with their corresponding vectors from U and V. Thus, A is approximated by

$$A_{n\times m} = U_{n\times k}\Sigma_{k\times k}V^T_{k\times m}. \tag{2.35}$$

Finally, LSA as a feature selecting technique is based on deciding on the goodness of a feature with respect to a certain class of documents by measuring their distance or similarity to that class in the LSA space.

2.4.3 Measuring Document Similarity

There are different plausible choices of a distance metric or a similarity measure defined on a set of documents coded as *TF/IDF* vectors. The particular choice is task and data dependent and in most of the cases, once the document vectors are computed, it is easy to try out different distance measures and pick out the most appropriate one.

A review of several commonly used distances and measures is found in [87]. Among them, the most applied ones are the **Cosine** similarity measure and the **Euclidean** distance defined for two documents $\mathbf{d_1} = (d^1_1, ..., d^1_n)$ and $\mathbf{d_2} = (d^2_1, ..., d^2_n)$ living in an n-dimensional space by the well-known formulas:

- **Euclidean distance**:

$$d_{euc}(\mathbf{d_1}, \mathbf{d_2}) = \sqrt{\sum_{i=1}^{n} (d_i^1 - d_i^2)^2}$$

- **Cosine similarity measure**:

$$d_{cos}(\mathbf{d_1}, \mathbf{d_2}) = \frac{\sum_{i=1}^{n} d_i^1 \cdot d_i^2}{\sqrt{\sum_{i=1}^{n} (d_i^1)^2 \cdot \sum_{i=1}^{n} (d_i^2)2}}$$

We note that **kernels** can be also used to evaluate the similarity of documents based on the general intuition that they are measures of similarity between two data examples. The simple dot product of vectors provides an example of a kernel and can be used as a similarity measure. As we shall see in next section, a special kind of kernel for structures is applied as a measure of the closeness in meaning of two documents.

2.4.4 An Overview of Text Categorization Approaches

Let us again recall the setting of the classification problem that we are to solve and the notations that we have decided to use. Our training data consists of m documents coded as n-dimensional vectors, $\mathbf{d_i} \in \mathbb{R}^n$, $i = 1, ..., m$. Additionally, let there be K categories $c_1, ..., c_K$. Each of the documents in our training set is assigned an output value $y_i \in \{c_1, ..., c_K\}$, according to its class label. The number of documents in a category c_i is denoted by m_{c_i}. Finally, let $\mathbf{d} = (d_1, ..., d_n)$ be an unseen document vector that we want to classify in one of the predefined categories.

Text categorization approaches use machine and statistical learning techniques to train an automatic classifier on a dataset of the described type with adequate generalization properties which will allow for the correct label assignment of unseen documents [57, 167]. We review some of the most prominent learning techniques that have been successfully applied so far to the task of categorizing text documents.

Naive-Bayes Learner

Bayes theorem for conditional probabilities is often of help for solving text categorization problems [106]. The classification technique provides an estimation of the probability of a class, given the feature vector of a new document by using the training data. The conditional class probability is given by

$$P(c_i|\mathbf{d}) = \frac{P(c_i)P(\mathbf{d}|c_i)}{P(\mathbf{d})}.$$

The learner has the part "naive" in its name, because, in order to estimate the quantity above, it makes the assumption that words are conditionally independent. This assumption is too strong and generally wrong. It is a fortunate fact that the Naive-Bayes classifier performs very well in practice, nevertheless. Conditional independence assumed leads to the following representation:

$$P(c_i|\mathbf{d}) = P(c_i) \prod_{j=1}^{n} P(d_j|c_i). \tag{2.36}$$

Note that $P(\mathbf{d})$ has been removed from the formula because it is a constant with respect to the categories. Estimates of the two probabilities left to calculate, based on the training set, are given in the following way:

$$\hat{P}(C = c_i) = \frac{m_{c_i}}{m} \tag{2.37}$$

$$\hat{P}(d_j|c_i) = \frac{1 + n_{ij}}{n + \sum_{k=1}^{n} n_{ki}}, \tag{2.38}$$

where n_{ij} denotes the number of times a word j occurs in documents from the category c_i [1].

Rocchio Algorithm

A largely applied classification method, proposed by Rocchio [128], consists in first building a prototype vector for each category c_i, $i = 1, ..., K$ by taking the average of all training documents belonging to that category. In order to assign a class to an unseen document $\mathbf{d}$, the similarity of each prototype vector and $\mathbf{d}$ is measured (commonly using the cosine measure). Then $\mathbf{d}$ is assigned the class whose prototype vector is found to be closest to $\mathbf{d}$.

Nearest Neighbors Learner

Nearest-neighbors methods for classification tasks produce a prediction of the class-value of a new instance based on the most commonly encountered class-value in the set of training points closest to the new instance in the input space. For continuous valued target functions, the k nearest neighbors prediction is calculated by

$$\hat{f}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i,$$

where y_i corresponds to the value of the function in the i-th training instance and $N_k(x)$ is the set containing the k closest to x training points [68, 106]. k-nearest neighbor classifiers have shown very good performance in text categorization tasks.

For a binary classification task, we will give a simple classification rule [98]. Take a proximity of a vector $\mathbf{d}$ in the input space containing its k-nearest neighbors and let k_1 be the number of positive and k_2 the number of negative examples among them. The instance $\mathbf{d}$ is assigned positive if the estimation of the conditional probability of membership

$$P(+1|\mathbf{d}) = \frac{k_1}{k}$$

is greater than the estimation of the conditional probability of membership

$$P(-1|\mathbf{d}) = \frac{k_2}{k};$$

the document is assigned negative otherwise.

Generalization to multiple classification is done trivially by solving a series of binary-class problems.

Decision Trees

The training sample can be used in order to construct a decision tree against which an unseen document is matched in order to find out its class-label. There are several decision trees algorithms which have been applied for text categorization and we will describe one of the most popular ones, CART [2].

CART consists in constructing a binary decision tree by splitting the vectors at each node with respect to one single vector component. In that the algorithm attempts to keep the collection of vectors in each resulting sub-category as less diverse as possible. The vector component which splits the documents best according to that criterion is the one which keeps the node-category diversity low. Commonly, *entropy* is used as a measure of diversity[7], given by the well known formula

$$\sum_{i=1}^{K} p(c_i|t)\log p(c_i|t). \tag{2.39}$$

In the quantity above, $p(c_i|t)$ is the probability of the class c_j given the decision tree node t and is estimated by

$$\hat{p}(c_i|t) = m_i(t)/m(t), \tag{2.40}$$

where $m_i(t)$ is the number of documents of class c_j in t and $m(t)$ is the cardinality of t. The decrease of diversity serves as a measure of the degree of importance of a single component, i.e. one maximizes the quantity $diversity$(before split)$-$[$diversity$(left child)$+diversity$(right child)].

The CART procedure runs until each document is grouped in a category of its own (the data is overfitted) and only then a pruning procedure removes those branches of the tree which have the least additional predictive power for the documents found on their leaves.

Making up for this drawback, CHAID – another commonly used decision tree technique – contains a criterion to stop growing the tree before overfitting of the data occurs [82].

Latent Semantic Indexing (LSI)

LSA has been discussed in the previous section as a feature selection technique. In its application in *information retrieval*, LSA, known as Latent Semantic Indexing (LSI) treats every query q as a small document which is projected onto the k-dimensional LSA space by the help of the transformation

$$\hat{q} = \Sigma_{k\times k}^{-1} U_{k\times n}^{T} q_n. \tag{2.41}$$

The query's relevance to a document or a group of documents is evaluated again by the help of a distance metric or a similarity measure.

Despite of using only $k < r < n$ orthogonal directions in order to describe the data, LSA and LSI suffer a considerable computational cost. In order to cope with this drawback, Cristianini and co-workers [31] have introduced a kernel-defined version of the method.

[7]Note that the method shares similarities with the previously introduced feature selection techniques *Information gain* and *Mutual information*, for it is also based on uncertainty reduction.

Topic Hierarchies

It is a commonly encountered text categorization problem to have to train an automatic classifier on a training set where the class labels are related in a hierarchical structure connected by subsumption. An example of such an intended organization of documents is every web-directory (such as Yahoo![8], the Open Directory Project[9], and other). The documents label assignments follow the inheritance rules of the hierarchy: every document that belongs to a category c_i also belongs to a category c_j if c_j is the parent of c_i.

The task of the classification is to assign to an unseen document a path form the tree-hierarchy which starts at the root (because all documents are trivially assigned to the root category) and ends with the node which is the most-distant to the root but still relevant to the document node. For that purpose, a set of search strategies can be applied, among the most popular ones being *greedy search* and *best first search* (discussed and compared in [24]).

Support Vector Machines and Text Categorization

The application of the Support Vector Machines classifier (see Section 2.2 for an overview) to text categorization tasks compared to other inductive learning methods is discussed broadly by Joachims in a 1998 paper [74]. By using the Reuters 21578 dataset[10], Joachims evaluated the performance of SVMs (with polynomial and radial-base kernels) compared to four state-of-the-art techniques – Naive-Bayes, Rocchio, Decision trees and k-Nearest Neighbors. The method proved to perform better than those techniques, substantially and significantly. Thanks to the good generalization properties of the method on high dimensional datasets, the feature selection task can be omitted, making the procedure less complex and time costly. Dumais *et al.* [40] preformed similar evaluation, which confirmed the results by Joachims reporting that the method performs excellently on large categorization problems, outperforming most of its contenders.

There are several main reasons why SVMs have been so successful. Most of the text-mining[11] problems are set in a space of a very large dimensionality, reaching more than 10,000 features, and SVMs has proved to be able to deal with high dimensions. The vectors of documents are usually sparse – this is again not a problem for an SVM algorithm to perform properly. Finally, Joachims argues that most of the text learning problems are linearly separable, and the original setting of the SVM classification consists in finding an optimal hyperplane between separable classes of observations. We note that this argument needs an additional remark. By linear separability should be understood that data are separable by the help with either hard or soft margin, because even if data appear to be linearly separable, the underlying distribution is usually not; should that be the case, applying the hard margin classifier might lead to overfitting.

[8]http://www.yahoo.com

[9]http://www.dmoz.org/

[10]Reuters 21578 is a publicly available dataset containing articles from Reuters news agency, compiled by David Lewis in 1987, http://www.research.att.com/lewis.

[11] *Text mining* is a term which denotes the domain of machine learning dealing with natural language texts (representation, pretreatment, classification, etc.).

Kernels Accounting for Semantic Proximity

We will close this section with a note on categorization methods based on kernels. Lohdi *et al.* [95] have argued that the construction of a feature vector out of a text document can be just as costly and as complex as the categorization task itself. In addition, by using term frequency features all information related to word order is lost. In Section 2.1.3, we have introduced and discussed kernel methods on structures, such as graphs and trees. We have pointed out that these methods are applicable on text documents as well, as long as a structural representation of text can be made available [103] and are suggested to perform efficiently for text categorization tasks not allowing for the drawbacks of (explicit) feature extraction methods [78].

We will stress on one problematic side of this group of approaches – documents are considered as sequences of symbols without using domain knowledge. A feature space is constructed by the set of all non-contiguous substrings of k symbols. By applying a string kernel the similarity of two documents is given by a simple heuristics: *The more substrings two documents have in common, the more similar they are considered to be [95].* In fact, the feature vectors of the documents are only implicitly represented by using string alignment techniques.

In the feature extraction model documents are real valued vectors living in a certain vector space of some dimensionality. Documents using distinct, although semantically similar terms are considered unrelated. For that reason, this model has been additionally criticized for its inability to exploit semantic similarity of terms. Kernel approaches can account for that drawback, as discussed by Kondola *et al.* in [81]. The similarity of two documents $\mathbf{d_i}$ and $\mathbf{d_j}$ is estimated by using a kernel evaluating their dot product by taking into account the semantic similarity of the *terms* which compose them in the following standard manner:

$$k(\mathbf{d_i}, \mathbf{d_j}) = \mathbf{d_i} P \mathbf{d_j'}, \tag{2.42}$$

where $.'$ denotes the transpose of a vector. What makes (2.42) different from a standard scalar product is that P is a *semantic proximity matrix*, which fulfills Mercer's conditions (see, e.g. [30] for a definition) defined in a special way to account for semantic proximity of the terms that define the vectors $\mathbf{d_i}$ and $\mathbf{d_j}$. In fact, every entry $P_{ab}(= P_{ba})$ of the matrix should give the strength of the relation of the terms a and b.

The authors propose two methods of (implicitly) computing the matrix P. The first method suggests that if we view documents as bags of words, we can equally view terms as bags of documents, in which the proximity of two terms will be calculated on the basis of the correlation of their document vectors. The second method is based on representing the set of all words in a given corpus as a graph, where each node stands for a word and the edges between nodes stand for words co-occurrences [47]. Semantically closer terms will be less distant in the co-occurrence graph.

To sum up, we note that text categorization is a fast developing and important subfield of machine learning, furthered by and furthering results in neighboring subfields of statistical inference, data mining and NLP. Because of the generality of the presented approaches, most of the results discussed in the current section are valid and can be applied, subject to minor modifications, on more general (categorical) multimedia data, such as videos or images.

2.5 Summary: A Framework for Ontology Matching

In the sections above, we have tried to review as concisely and in the same time as exhaustively as possible several standard subfields of mathematics, machine learning and statistical inference as a background for a wide range of ontology matching approaches which will be discussed throughout the following chapters.

We started by presenting basic results and definitions from graph and lattice theory (Section 2.1). As we will see in the ensuing chapters, graphs are helpful for visualizing and embodying the structural properties of an ontology. Although relations of order are not explicitly defined for graphs, intuitively a tree can be thought of as an ordered set of nodes, where the adjacency relation of nodes is, by convention, substituted by a partial order. We introduced lattices and semi-lattices, which are defined as sets equipped with a partial order relation. In fact, formally, a tree can be viewed as a particular semi-lattice. This gives rise and enables replying to a central question: how can an ontology matching environment, defined on hierarchical structures (tree-like ontologies) be generalized for ontologies containing other than subsumptional relations. We will discuss this question in more detail in Chapter 4 of the thesis. Additionally, a common mathematical model of ontological knowledge representation is given by the so called Formal Concept Analysis (FCA), which now has become a standard field of lattice theory. Although not directly related to our approach, FCA has been presented later in Chapter 3 of the thesis, together with a bridging approach for translating an ontology into a concept lattice.

The structure of a set of concepts, or knowing how a concept is related to other concepts, can tell us a lot about the semantics of the entities that they model. However, structure alone cannot account for the semantics of a given concept or what could its extension be. On simply structural bases, we cannot judge whether or not two concepts taken in isolation are similar or dissimilar and to what degree. Therefore, the main emphasis of the thesis will fall on instance-based approaches to concept similarity. In the current chapter, we gave an introduction to Support Vector Machines and discussed issues related to their generalization properties (Section 2.2), we introduced main concepts from variable and feature selection (Section 2.3), and, finally, we revised approaches to automatic text categorization (Section 2.4). These techniques operate entirely on real-world *instance* data and will be our basic tools for measuring the semantic closeness of concepts as a part of an overall procedure for extensional (instance-based) ontology matching.

As different as some of the fields described in this chapter may appear at a first glance, all the discussed results will be necessary in order to introduce a general approach to overcoming semantic discrepancies by combining these results into a single framework for ontology matching. In addition to that, the presented overview sections provide a broad enough ground for discussing a wide variety of other related approaches and tools for ontology matching from theoretical, practical, conceptual and motivational viewpoints. This is where the main focus of the following chapter of the thesis falls.

Chapter 3

Ontology Matching

The present chapter discusses ontology matching by combining two main perspectives. On one hand, we approach the problem from a broader point of view, speculating on possible origins of that problem by taking a cognitive science stance. On the other hand, we present in a structured manner that part of the state-of-the-art of the research in ontology matching, which is relevant to our contribution to the field.

We start by discussing the question of why it is difficult to define an ontology and try to formulate a definition, which covers most of the common conventions of what an ontology is and fits best the matching approach that we propose (Section 3.1). Further, we discuss and classify possible application scenarios (Section 3.2) and present two standard ways of modeling ontologies by using *description logics* and *formal concept analysis* (Section 3.3). They are not directly relevant to our approach, but are building blocks for many approaches related to ours. Section 3.4 contains a discussion of main motivational and conceptual issues concerning the problems of ontology matching and semantic heterogeneity, trying to sketch their evolution and origins from psychological and broader computational points of view. Finally, a classification of different existing ontology matching approaches is given in Section 3.5, before we conclude the chapter (Section 3.6).

3.1 Defining an Ontology

Ontologies in Artificial Intelligence have been introduced to describe the semantics of data in order to provide a uniform framework of understanding between different parties. Ironically enough, despite this intention, there exists little agreement on a common definition of an ontology among different authors, many of which have proposed their own formal definitions, each taking into account different aspects of the acquisition, modeling and intended application of ontologies. The main common reference to an ontology definition was provided by Gruber back in 1993, describing ontologies as knowledge bodies which bring a formal representation of a shared conceptualization of a domain – the objects, concepts and other entities that are assumed to exist in a certain area of interest together with the relationships holding among them. Gruber wrote:

An ontology is an explicit specification of a conceptualization. The

> term is borrowed from philosophy, where an Ontology is a systematic account of Existence - the study of *what there is.* For AI systems, what "exists" is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge [61].

Gruber's definition suggests that an ontology possesses loosely a set of concepts and a set of relations between these concepts. The core-bodies of ontologies are taxonomies – hierarchical structures that organize concepts by a subsumption (`is_a`) relation. Web directories, such as *Yahoo!* or the *Open Directory Project* are examples of taxonomies which classify items in a given domain of interest. In the sequel, we will speak of hierarchical ontologies and we will define precisely what we mean by that in Chapter 4, Section 4.1.1, but let us nevertheless discuss what could a formal definition of an ontology in a broader sense look like.

Despite the relative liberality among the members of the scientific community concerning the question *"what is an ontology?"*, we will provide one formal definition, which is general enough to satisfy many existing understandings of that question, including ours.

Alexander Mädche and Steffen Staab suggested a set of characteristics that an ontology should or could possess which they called *ontology primitives* [97]. The set of ontology primitives consists of:

1. a set of lexical entries $\mathcal{L}$ for concepts and relations;
2. a set of concepts $\mathcal{C}$;
3. a taxonomy of concepts with multiple inheritance (heterarchy) $\mathcal{H}_C$;
4. a set of non-taxonomic relations $\mathcal{R}$ described by their domain and range restrictions;
5. a hierarchy (or heterarchy) on the relations $\mathcal{R}$, $\mathcal{H}_R$;
6. mappings $\mathcal{F}$ and $\mathcal{G}$ that relate concepts and relations with their lexical entries, respectively;
7. a set of axioms $\mathcal{A}$ that describe additional constraints on the ontology and allow to make implicit facts explicit.

We will make several comments on the list of primitives above, in order to clarify its components.

- The set $\mathcal{L}$ is understood as the set of direct lexical references to the concepts and relations in question, e.g. "School" for the concept SCHOOL, "Parent" for the relation `parent`.
- The taxonomy is defined by a partial order on the set of concepts.

- The set $\mathcal{R}$ is left without a precise definition of what kind of relations it might contain. Some examples are the partonimic relation (`part_of`), as well as non-standard relations defined by the ontology engineer (e.g. `parent`, `employed_by`, `graduated_at`, etc.).
- The set $\mathcal{A}$ includes axioms, which do not follow directly from the defined relations and concepts, but are important for modeling the respective domain. They can come from background knowledge sources like dictionaries, thesauri or top-level ontologies.

Based on the proposed list of primitives (but not considering all of its ingredients), Stumme and Mädche gave the following definition of an ontology [144]:

Definition 26 *A (core) ontology is a tuple* $\mathcal{O} := (\mathcal{C}, \texttt{is\_a}, \mathcal{R}, \sigma)$*, where* $\mathcal{C}$ *is a set whose elements are called concepts,* `is_a` *is a partial order on* $\mathcal{C}$*,* $\mathcal{R}$ *is a set whose elements are called relation names (or relations for short), and* $\sigma : \mathcal{R} \to \mathbb{N}$ *is a function which assigns to each relation name its arity.*

An extended version of this definition is found in [79].

We note that the definition above does not prevent from the possibility each of the defined sets to be the empty set, for example the set $\mathcal{R}$, or even the set $\mathcal{C}$. And if an ontology with no relations between its concepts is trivial but still, at least in theory possible, an ontology which consists only of relations but no concepts does not have neither practical nor theoretical meaning, for relations are *defined* on a set of concepts. For that reason, we add to definition 26 one necessary condition for $\mathcal{O}$ to be an ontology: $\mathcal{C} \neq \emptyset$. In fact, adding or removing elements from the definition above moves the ontology that it defines on the axis of expressiveness. An ontology which contains only a set of concepts is the least expressive one. We gain more expressiveness by adding the relation `is_a` and further relations defined on the set of concepts.

In the next sub-section, we will consider various ontology application scenarios.

3.2 Ontology Applications

"Ontology applications" is the title of the last and most voluminous part of the *Handbook of Ontologies*, edited by Stefan Staab and Rudi Studer [143]. Its eleven chapters go into different aspects of application of ontologies in multiple real life scenarios. Since this is one of the few available endeavors in classifying ontology application fields to date, we will describe it briefly in the sequel. We will further argue that the application fields can be organized in a more consistent manner.

Staab and Studer have conventionally classified the ontology application fields into two big families – *Knowledge Management* and *Interoperability and Integration (of Enterprise Applications)* (Figure 3.1).

The first class of applications aims at answering the question how ontologies can be of help in support of the identification, creation, representation and distribution of knowledge. This includes the use of ontologies to support the corporate memory of a *a virtual business partnership* using flexible, but well

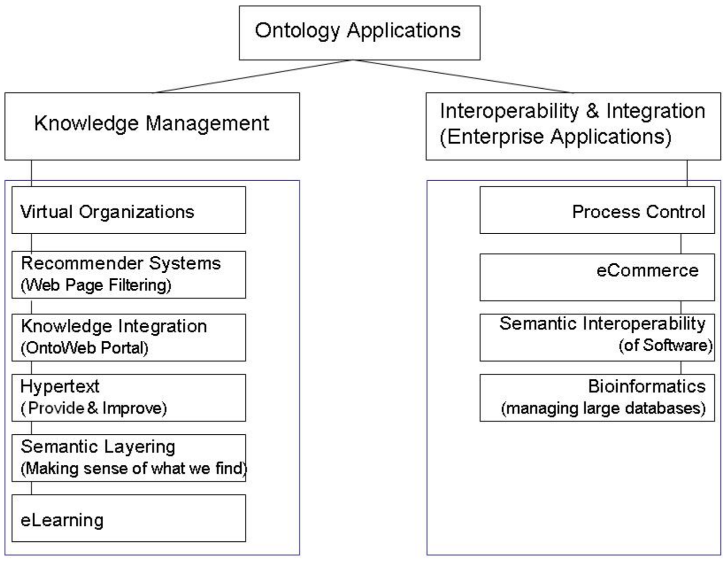

Ontology Applications
Knowledge Management
Interoperability & Integration (Enterprise Applications)
Virtual Organizations
Recommender Systems (Web Page Filtering)
Knowledge Integration (OntoWeb Portal)
Hypertext (Provide & Improve)
Semantic Layering (Making sense of what we find)
eLearning
Process Control
eCommerce
Semantic Interoperability (of Software)
Bioinformatics (managing large databases)

Figure 3.1: Ontology applications: the classification of Staab and Studer.

understood for both humans and machines document-based data structures. Ontologies are main bodies in the Semantic Web [12], therefore researchers and practitioners have put efforts into developing different Semantic Web improvement scenarios. In that context, ontologies are applied in various topics, such as *Recommender Systems* (Web Page Filtering), *Knowledge Integration* (the OntoWeb Portal project[1]), provision and improvement of *hypertext*, *Semantic Layering* (or "making sense of what we find"'). Finally, using ontologies to support *eLearning* finds ultimately place in this general class of applications.

The second class of applications is centered around the role of ontologies for providing interoperability and integration of enterprise applications. Discussed are applications in the fields of *Process Control* (within a company or between multiple partner companies), *semantic interoperability of software* (how to enable the cooperation of two software applications that were initially not developed for this purpose) and *eCommerce*. Finally, ontologies are able to manage large data bases; this has found place in Staab and Studer's classification in the context of *bio-informatics* – a broad contemporary ontology application field.

We would like to comment on the structure and the completeness of the proposed classification.

- Provided the big variety of real life areas where ontologies play a role, splitting the various application fields in only two classes is not granular enough.

- Moreover, the reader is left with the impression that most applications of both classes at the end have to do with knowledge management for the purposes of the *eBusiness*, which is a false suggestion.

[1]http://www.ontoweb.org/

- Virtual Organization, Knowledge Integration and Semantic interoperability in their essence tackle with the same problem and are driven by the same motivation of providing a mutual framework for semantic homogeneity and aim at similar application domains.
- Semantic Layering, Hypertext and Recommender Systems can also be grouped together because, as observed above, they are basically Semantic Web driven applications.
- Interoperability and Integration of data may be viewed as a sub-domain of Knowledge management.
- Some important application fields have been left out.
  - Ontologies in support of *problem solving* is a prominent application domain. Problem solving methods provide reusable reasoning components by specifying the way in which new facts can be inferred from existing facts on the basis of some set of logical axioms complementing an ontology. In that sense they could be classified in the Knowledge Management part of ontology application tree or they can form a class of their own – ontologies as inference systems.
  - *Planning* in Artificial Intelligence deals with building action strategies to be realized by intelligent agents. Applying ontologies in planning for providing semantics to the sequences of actions is a growing research topic.
  - Ontologies play an important role in *Natural Language Processing.* Esteval and co-workers [42] discuss the possibility of coupling ontologies with the lexicon used in a natural language component of a system for facilitating presenting and retrieving of information.

Our own contribution to a classification of the ontology application domains builds on the work of Mizogouchi [108] and Staab and Studer by using parts of the typology presented by Mizogouchi in order to classify the application domains discussed by Staab and Studer and some more fields that we find necessary to include. The new classification tree is represented in Figure 3.2.

We have split the applications in three main blocks, containing intersecting application instances.

- **Ontologies providing a common vocabulary**

  This is the most intuitive and straightforward type of ontology application: having a common vocabulary is the first step towards the systematization and the sharing of knowledge of a given domain.

- **Ontologies in support of information access**

  Ontologies provide vocabulary for annotation of web resources and enable agents to use hierarchy and class relations in order to interpret this vocabulary. This is a step towards making information access more intelligent and using the enormous information sources of the World Wide Web.

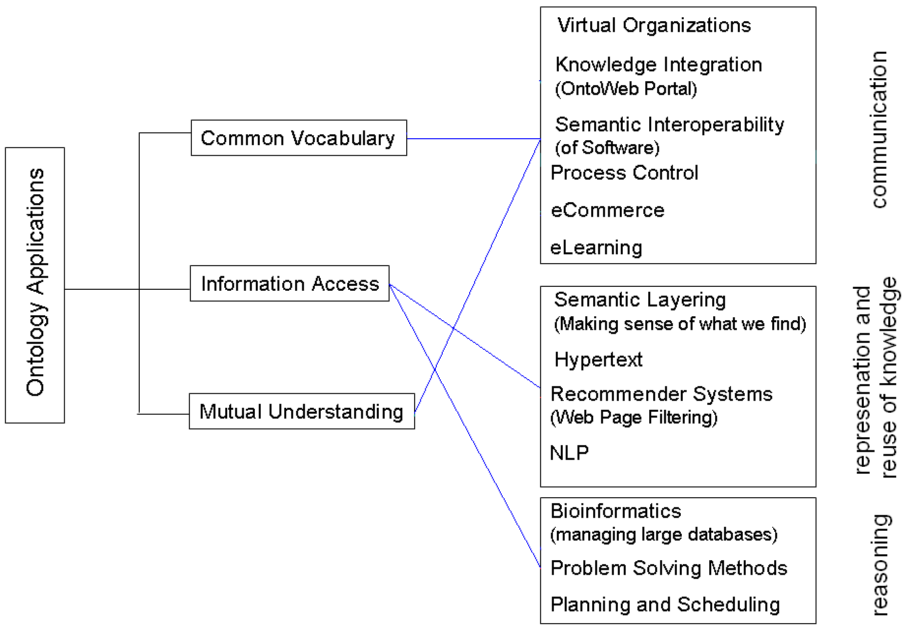

Ontology Applications
Common Vocabulary
Information Access
Mutual Understanding
Virtual Organizations
Knowledge Integration (OntoWeb Portal)
Semantic Interoperability (of Software)
Process Control
eCommerce
eLearning
Semantic Layering (Making sense of what we find)
Hypertext
Recommender Systems (Web Page Filtering)
NLP
Bioinformatics (managing large databases)
Problem Solving Methods
Planning and Scheduling
communication
represenation and reuse of knowledge
reasoning

Figure 3.2: Application fields of ontologies: a suggested classification.

- **Ontologies for mutual understanding**

  Mutual understanding considers two types of communicating agents – humans and software agents. Each of them can be on either side of the communication line.

  Communication between humans can be facilitated by ontologies by providing environments for knowledge-intensive engineering such as concurrent engineering, business process re-engineering and other.

  A big part of the ongoing ontology research driven by the core ideas that lie in the project of the Semantic Web [12] concerns *understanding between humans and software agents*, seen in the case of web resources search and use. (An application scenario of ontologies on the semantic web scale is presented in Figure 3.3.)

  Communication between software agents has been discussed above in terms of allowing the cooperation of two software applications that were initially not developed for this purpose.

Finally, the interested reader may like to consider the work of Gaitanou [49] which presents yet another classification of the ontology application fields divided into six main fields: *Semantic Web and knowledge management*, *Large scale applications – machine translation*, *E-commerce*, *Multimedia and graphics*, *Peer-to-peer networks* and *Pervasive computing environments*.

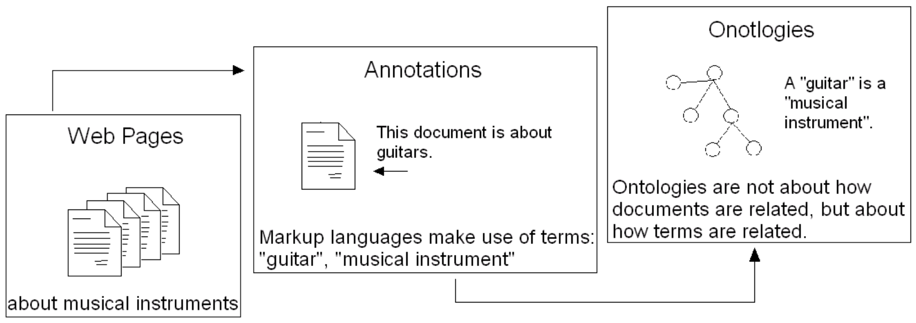

Web Pages
about musical instruments
Annotations
This document is about guitars.
Markup languages make use of terms: "guitar", "musical instrument"
Onotlogies
A "guitar" is a "musical instrument".
Ontologies are not about how documents are related, but about how terms are related.

Figure 3.3: Semantic Web Ontologies.

3.3 Modeling Ontologies

We will discuss two basic models for ontologies, based on entirely different theoretical mechanisms: *description logics*, which evolved from knowledge representation research [91] and *formal concept analysis*, which is considered as a sub-field of lattice theory and one of its prominent practical applications [51].

3.3.1 Ontologies in Description Logics

In our study, we focus on hierarchical ontologies designed for text categorization. We use order theoretic and graph theoretic approaches in order to embody their structural properties and measure their structural similarity. We infer conclusions on the ontologies semantic closeness by combing structural measures with instance-based concept similarities, based on modeling concepts as sets of text documents.

In the sequel, we will give a parallel representation of (hierarchical) ontologies within a Description Logics (DL) framework. DL stands for a set of expressive formal languages largely applied in representing ontology terminologies. We will argue that the $\mathcal{AL}$ language (standing for Attributive Language) is the sufficient language to describe the ontologies of our interest. Before we do so, however, we give some remarks on the motivation and evolution of Description Logics.

Knowledge Representation (KR) in Artificial Intelligence aims to provide a high level description of the world embedded in intelligent applications which enable a system to find implicit consequences of explicitly represented knowledge. Description logics evolved as a framework which provides a representation of the knowledge of a domain (the "world") by defining the relevant concepts (the terminology) and using these concepts to specify properties of objects and individuals occurring in the domain (the world description) [112].

In Figure 3.4, we have traced the most important moments in the KR history, which have led to the development of the Description Logics based languages. As we can see from the diagram, on one hand knowledge representation has been developing in a purely logic-oriented framework; on the other hand and mostly not independently from the former, non-logic-based approaches have been applied. The latter include semantic networks based representations, furthered by the famous experiments by Collins and Quillian [28] in which nodes characterize concepts (sets of individuals) and links between nodes characterize

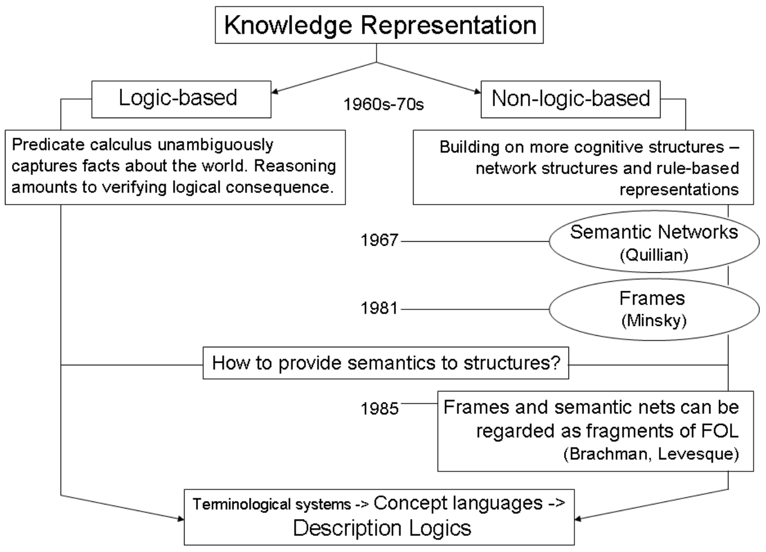

Knowledge Representation
Logic-based
1960s-70s
Non-logic-based
Predicate calculus unambiguously captures facts about the world. Reasoning amounts to verifying logical consequence.
Building on more cognitive structures – network structures and rule-based representations
1967
Semantic Networks (Quillian)
1981
Frames (Minsky)
How to provide semantics to structures?
1985
Frames and semantic nets can be regarded as fragments of FOL (Brachman, Levesque)
Terminological systems -> Concept languages -> Description Logics

Figure 3.4: KR: A historical track.

relationships. A hierarchy is defined by the `is_a` relationship: the more specific concept inherits the properties of the more general one. More complex relationships (roles) can be also represented as nodes. These kinds of relationships are also inherited by the sub-concepts in the hierarchy. Most importantly, semantic networks allowed the inference of new relationships on the basis of old ones. The observation that semantic nets can be viewed as fragments of First Order Logic, the development of Frame Languages, combined with the need to provide semantics and more expressiveness to structure-based representations led to the introduction of the Description Logics family [7].

To sum up, DLs are a family of knowledge representation formalisms equipped with a formal, logic-based semantics and reasoning is their central service. We outline three basic facts about DLs [112]:

- The basic syntactic building blocks are **atomic concepts** (unary predicates), **atomic roles** (binary predicates) and **individuals** (constants);
- The languages use a rather small set of **constructors** for building complex concepts and roles;
- Implicit knowledge about concepts and individuals can be inferred automatically by the help of **inference procedures**.

The way the entities listed above interact is sketched in Figure 3.5. The expressiveness of DL languages change with respect to the defined constructors. The Attributive Language ($\mathcal{AL}$), often considered as the minimal DL language of practical interest, is described in some more detail below.

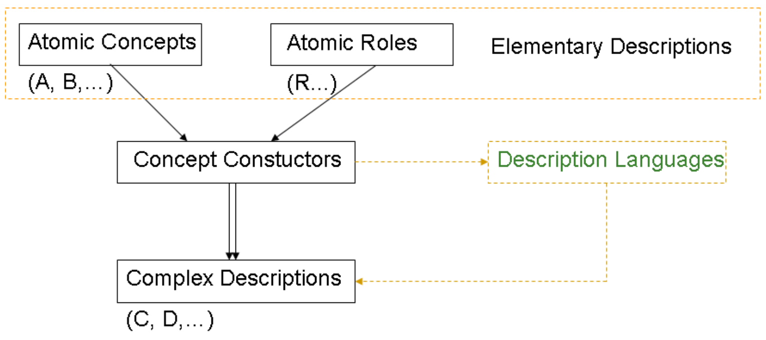


Figure 3.5: DL: Elementary and complex descriptions.

Syntax	Semantics
A| atomic concept	
$\top$| universal concept	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
$\bot$| bottom concept	$\bot^{\mathcal{I}} = \emptyset$
$\neg A$| atomic negation	$(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
$C \sqcap D$| intersection	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\forall R.C$| value restriction	$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \| \forall b : (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
$\exists R.\top$| limited existential quantification	$(\exists R.\top)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \| \exists b : (a,b) \in R^{\mathcal{I}}$

Table 3.1: DL: Syntax and semantics of $\mathcal{AL}$.

The Attributive Language ($\mathcal{AL}$)

Syntax

The basic syntax of $\mathcal{AL}$ underlies all description logics. It consists of a set of *concept names* (unary predicates), a set of *role names* (binary relations) and definitions of new concepts from concept names and role names using *constructors*. An $\mathcal{AL}$-concept is every atomic concept, the top concept $\top$ and the bottom concept $\bot$. Additionally, if C is an atomic $\mathcal{AL}$-concept then so is its complement $\neg C$; if C and D are $\mathcal{AL}$-concepts, so is their intersection (conjunction) $C \sqcap D$; if C is an $\mathcal{AL}$-concept and R is a role name, then $\forall R.C$ is also an $\mathcal{AL}$-concept (value restriction); and, finally, if R is a role name, then $\exists R.\top$ is also an $\mathcal{AL}$-concept (existential restriction) (see Table 3.1 (left)).

An extended version of $\mathcal{AL}$ logics may include *concept disjunction* $C \sqcup D$, where C and D are $\mathcal{AL}$-concepts, *full existential quantification* – if R is a role name, then $\exists R.C$ is also an $\mathcal{AL}$-concept and *number restriction* $\leq nR.C$ and $\geq nR.C$, where C is a $\mathcal{AL}$-concept and R is an $\mathcal{AL}$-role.

Semantics

Let $\Delta^{\mathcal{I}}$ be one of the possible domains over which the $\mathcal{AL}$ Description Logic quantifies – a non-empty set of objects and let $\cdot^{\mathcal{I}}$ be an interpretation function, which maps individual, concept and role names to, respectively, elements, subsets or products of the domain. More precisely, for an atomic concept C, its interpretation is given by $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. Analogously, a role R is interpreted by $\cdot^{\mathcal{I}}$

as $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

The $\mathcal{AL}$-concepts interpretation is given in Table 3.1 (right). For the extended $\mathcal{AL}$, we have additionally:

$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$,
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} |$ there is a $y \in \Delta^{\mathcal{I}}$ with $\langle x, y\rangle \in R^{\mathcal{I}}$, and $y \in C^{\mathcal{I}}\}$.
$(\leq nR.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \sharp R^{\mathcal{I}}(x, C) \leq n\}$,
$(\geq nR.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \sharp R^{\mathcal{I}}(x, C) \geq n\}$, where $\sharp N$ denotes the cardinality of a set N and $R^{\mathcal{I}}(x, C) := \{y | \langle x, y\rangle \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$. If $x \in C^{\mathcal{I}}$, we say that x is an `instance` of C in $\mathcal{I}$, and if $\langle x, y\rangle \in R^{\mathcal{I}}$, then y is called an R-`successor` of x in $\mathcal{I}$.

Formalizing ontologies in a TBox

In order to be able to do reasoning with ontologies, one needs to define a central ontological notion, the so called TBox, explained as a set of terminological axioms which describe the intensional knowledge of the world. Axioms are equivalences of the kind $C \equiv D$ or inclusions, such like $C \sqsupseteq D$ for two concepts C and D. More formally, we give the following definitions.

Definition 27 ***Subsumption and equivalence of concepts.*** *Given two concept names C and D, C subsumes D, denoted by $C \sqsupseteq D$ if and only if it is true that $C^{\mathcal{I}} \supseteq D^{\mathcal{I}}$, for all $\mathcal{I}$. The equivalence relationship, denoted by $C \equiv D$ is induced by $C \sqsupseteq D$ and $D \sqsupseteq C$, interpreted as $C^{\mathcal{I}} = D^{\mathcal{I}}$.*

Definition 28 ***A general concept inclusion (GCI)*** *is of the form $C \sqsubseteq D$, where C, D are $\mathcal{AL}$-concepts. A finite set of general concept inclusions is called a TBox. An interpretation I is a* model *of a TBox $\mathcal{T}$ if and only if it satisfies all GCIs in $\mathcal{T}$, i.e., it holds that $\forall C : C \sqsubseteq D \in \mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.*

Remark 29 *Defining axioms as inclusions only, and not as equivalences is sufficient, following definition 27. In that sense, concept definitions can be expressed by two inclusions.*

Formalizing ontologies in a TBox is a well studied task. We will sketch the general procedure which one follows.

Before introducing the specific formal language in which the ontology is to be presented (which will lay the "walls" of the TBox), one usually defines the *possible worlds* by restricting the allowed interpretations. That is done by declaring a set of GCIs. The properties of each of the elements of our world can be expressed by additional GCIs. The concepts contained in our world or application domain are defined by concept definitions of the type $A \equiv B$. A *defined concept* is every concept which is found on the left hand side of a $\equiv$-expression [44]. In doing so, it is important to ensure that there are no multiple definitions, as well as that the defined names do not occur in any of the additional GCIs.

Example: A definition of a concept:
$\texttt{AcousticGuitar} \equiv$
$\texttt{MusicalInstrument} \sqcap (\leq 6\, \texttt{hasStrings}.\top) \sqcap (\geq 6\, \texttt{hasStrings}.\top) \sqcap \exists \texttt{isHollow}.\top$

To sum up, a TBox can, on one hand, axiomatize the basic notions in an application domain (the primitive concepts) by GCIs and role inclusions. On the other hand, more complex notions (the defined concepts) can be introduced by concept definitions. The subsumption hierarchy of the defined concepts induces the *taxonomy* of the TBox [6].

World Description: The ABox

The ABox is defined as a set of assertions about the individuals, sometimes called *membership assertions.* It contains the extensional knowledge about the domain, described by the TBox.

Definition 30 ***Syntax and semantics of the ABox.*** *The ABox is a finite set of assertions of the form $C(a)$ or $R(a, b)$, where a and b are individuals, C is a concept and R is a role. The interpretation $\mathcal{I}(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model for two individuals $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ and $b^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ under the conditions:*
$\mathcal{I} \models C(a)$ *iff* $a^{\mathcal{I}} \in C^{\mathcal{I}}$ *and*
$\mathcal{I} \models R(a, b)$ *iff* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Example: the assertions Father(PETER) and hasChild(PETER, JOHN) are a part of an ABox and mean that the individual Peter is an instance of the concept "Father" and has a child, John.

Hierarchical ontologies in $\mathcal{AL}$

As stated before, we are interested in ontologies that can be structurally seen as directed rooted trees (see Definition 39, Section 4.1.1). Later, we introduce a procedure of generalization which allows by the help of operations on trees (such as a Cartesian product) to deal with more general structures, such as semi-lattices and DAGs. Nevertheless, in the center of our attention remain ontologies structured around a strict hierarchical backbone containing only subsumptional (`is_a`) relations between concepts, so called hierarchical ontologies. We will see in Section 4.2 that there is a trivial translation of hierarchical ontologies into trees, where the root node of the tree represents the most general ontology concept, all other nodes represent sub-concepts of the root and its successors and all edges represent the subsumptional relations between classes. It is just as simple to represent the hierarchical ontologies in $\mathcal{AL}$ by using the analogue to a tree-structure.

- The root node is the top concept $\top$.
- Any other node of the tree is a $\mathcal{AL}$ concept.
- The adjacency relation `is_a` between two nodes corresponds to the relation $\sqsubseteq$ between two concepts.

An interpretation $I = (\Delta^I, \cdot^I)$ consists of the set $\Delta^I = M$, where M is the set of all instances of concepts contained in a document set Δ as will be defined later and $\cdot^I$ maps a role to a subset of $M \times M$. All the definitions of the semantics of the concepts and the relations remain unchanged. For example, the concept $C_1 \sqcap C_2$ will be interpreted in the following way: $(C_1 \sqcap C_2)^I = C_1^I \sqcap C_2^I$, where $(C_1)^I \subseteq M$ and $(C_2)^I \subseteq M$. All other interpretations follow trivially from the definitions above.

The following section focuses on Formal Concept Analysis - a data driven model for conceptual knowledge representation, formally introduced as a subfield of lattice theory.

3.3.2 Ontologies and Formal Concept Analysis

Formal Concept Analysis (FCA) is another theoretical framework of representing the knowledge in a certain domain of interest, which is different from DL-based models of concepts and their relations mainly in the fact that FCA always relies on data, on some set of objects or a fragment of the world in order to construct a model. Ontologies, in general can exist without instances, simply as intensional structures; FCA is rather an artifact derived from a dataset [27].

Again, due to the fact that this work focuses exclusively on tree-structured ontologies, which categorize documents, we will discuss the Formal Concept Analysis model within the scope of hierarchical document populated structures.

Formal Concept Analysis

Formal Concept Analysis (FCA) has been introduced as a mathematical theory for concept modeling in terms of order and lattice theory. It incorporates a mathematical description of the notions of concept extension and intension. An ontology is modeled as a complete lattice where each lattice node represents a concept as a pair of a set of *objects* and a set of *attributes*. Each concept inherits the attributes of its super-concepts and each concept contains all objects of its sub-concepts. We will present several formal definitions and properties, taken from the introductory book by Ganter and Wille [51].

Definition 31 *A (formal) context is a triple $K := (G, M, I)$, where G is a set whose elements are called objects, M is a set whose elements are called attributes, and I is a binary relation $I \subseteq G \times M$. $(g, m) \in I$ is read "object g has attribute m".*

Definition 32 *For $A \subseteq G$ and $B \subseteq M$, we define the corresponding sets*

$$A' := \{m \in M | \forall g \in A : (g, m) \in I\}$$

and

$$B' := \{g \in G | \forall m \in B : (g, m) \in I\}.$$

A (formal) concept of a formal context K (definition 31) is defined as a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. A and B are correspondingly the *extent* and *intent* of the formal concept (A, B).

We note that the use of the same term "concept" in FCA and in ontologies for denoting different things sometimes causes confusion. Both approaches are just two different models for concept representation independent from one another both historically and conceptually. Ontology concepts are most close in function to the attributes of FCA for reasons related to their similar roles of unary predicates on the set of objects [144].

The following proposition presents some important properties of the intents and extents of formal concepts.

Proposition 33 *Let $K := (G, M, I)$ be a formal context and let $A, A_1, A_2 \subseteq G$ and $B, B_1, B_2 \subseteq M$. For the object sub-sets, it holds*

1. $A_1 \subseteq A_2 \Rightarrow A_2' \subseteq A_1'$
2. $A \subseteq A''$
3. $A' = A'''$

for the attributes subs-sets it holds

1. $B_1 \subseteq B_2 \Rightarrow B_2' \subseteq B_1'$
2. $B \subseteq B''$
3. $B' = B'''$

and
$A \subseteq B' \Leftrightarrow B \subseteq A' \Leftrightarrow A \times B \subseteq I$.

A proof of the proposition, although straightforward, can be found in [51].

Formal concepts can be ordered in a conceptual hierarchy naturally evoked by the subconcept-superconcept relation. A "cow" is a subconcept of an "animal" because all cows are animals. A partial order is introduced on the set of formal concepts in order to model the concept hierarchy.

Definition 34 *Let (A_1, B_1) and (A_2, B_2) be two formal concepts. It holds that $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2$ or, equivalently (due to proposition 33), $B_2 \subseteq B_1$.*

Hierarchical ontologies as concept lattices

The relation between ontologies and formal concepts has been a subject of discussion in the past years. As stated above, the main difference between FCA and ontologies, is that the former relies on instance-data to a higher degree than the latter. However, since we are studying ontologies *with* instances, we will attempt to provide a straightforward manner of stating the correspondences between the elements of an ontology (instances and concepts) and those of a formal concept lattice (objects and attributes).

In previous studies, it has been argued that, most naturally, ontology concepts are directly identified with formal concepts. However, the fact that attributes can be viewed as concepts themselves[2] furthered the development of the view that ontology-concepts are more likely to correspond to both the attributes and the objects of a formal concept lattice (see an example from the tourist domain taken from Cimiano *et al.* in Figure 3.6).

In the current section, we will discuss what is another possible approach to translate tree-structured ontologies containing text documents (instances) assigned to their nodes into concept lattices in terms of FCA. We will criticize a state-of-the-art approach (reviewed as well in Section 3.5) and discuss a more intuitive, in our view, manner of accomplishing the goal.

[2]Cited by Cimiano, Hesse argues that "[attributes] are units of thought which are gained by abstraction, and hence they are also concepts" [27].

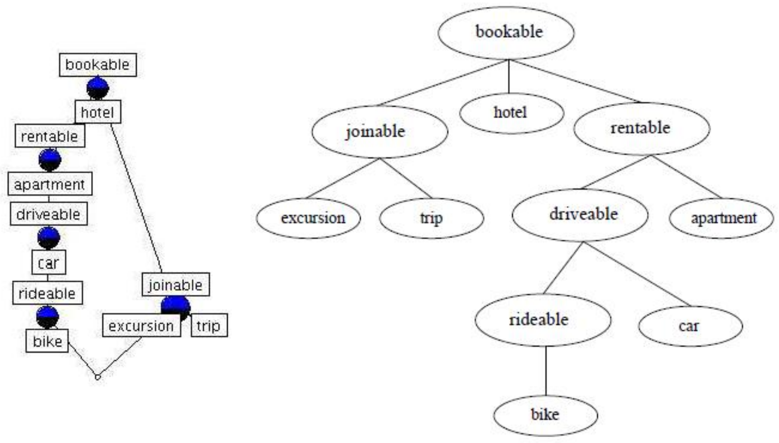


Figure 3.6: A formal concept lattice into a taxonomy (from [27]).

The FCA-Merge approach to ontology mapping, introduced by Stumme and Maedche in 2001, is a method for merging two ontologies, populated with instances taken from a set of text documents relevant to both ontologies. The approach applies natural language processing techniques and FCA to derive a concept lattice as a result of the merging procedure which is further transformed back into a merged ontology [144]. For each ontology O_i a unique formal context $K_i := (G_i, M_i, I_i)$ is generated in the following manner. The set of documents D_{O_i} corresponding to O_i is taken as the object set of the formal context, i.e. $G_i := D_{O_i}$ and the set of ontology concepts is taken as an attribute set, i.e. $M_i := C_{O_i}$. The relation $(g, m) \in I_i$ holds whenever a document $g \in D_{O_i}$ contains an instance of $m \in C_{O_i}$.

As the identification of the set of documents with the FCA object set is straightforward, we will argue that identifying the set of ontology concepts with the set of FCA attributes is, at least, not highly intuitive. A formal concept consists of attributes and objects such that, when ordered in a concept lattice, each concept inherits the attributes of its super-concepts and contains all objects of its sub-concepts. The inheritance rules that hold for concepts are different from those holding for attributes. A concept in a hierarchical ontology inherits the properties of its parent-concept. An attribute from a FCA-concept is more similar to what we have just called properties, than to the ontology-concept itself. Merely mapping the attributes to the ontology-concepts makes it unclear how exactly the subsumption of concepts is realized in terms of FCA-attributes.

We see that as a problematic side of this approach and, in order to avoid this confusion, we propose making use only of the documents assigned to the concept nodes in order to define a formal concept and translate an ontology into a FCA concept lattice.

Let us divide the set of documents in two major sets of subsets in the following manner. For any given concept C_i, let $D_{sup}^{C_i}$ denote the set of documents assigned to the concept C_i and all of its super-concepts. Analogously, let the set $D_{sub}^{C_i}$ denote the set of documents assigned to the concept C_i and all of its sub-concepts. The set $D_{sub}^{C_i}$ is the extension of the concept C_i, containing all

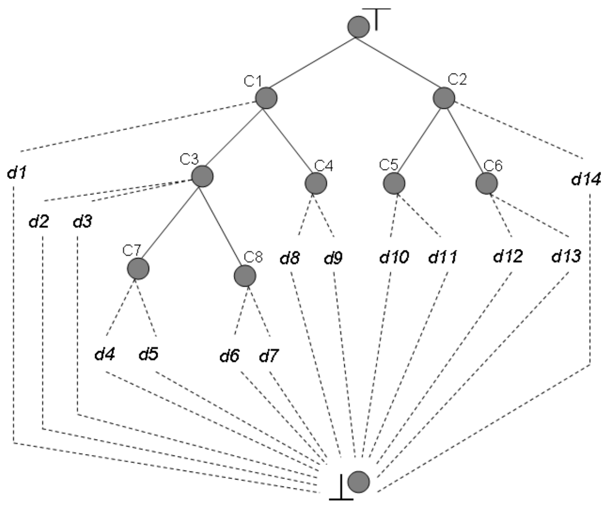


Figure 3.7: Hierarchical ontology to FCA.

concepts instances and is thus translated into the concept's object set in FCA. The set $D^{C_i}_{sup}$, on the other hand, is more similar to the concepts intent, because this is the set of documents whose associated concepts are subsumers of C_i. For that reason the set $D^{C_i}_{sub}$ is identified to the FCA attribute set of a concept.

Let n be the number of concepts in a given ontology O. In terms of a formal context, we define:

$$\cup_{i=1,...,n}\{D^{C_i}_{sup}\} := M$$

$$\cup_{i=1,...,n}\{D^{C_i}_{sub}\} := G$$

The relation I is again defined as $I \subseteq M \times G$, or equivalently $(m, g) \in I$ and one reads: a document as an object m has a document g as its attribute.

Finally, we note that some nodes, such as the root node, need not always be assigned a document to, which will deprive them of attributes. To make up for that we introduce the slack document d_0 to be assigned to the root node and additional slack documents d^0_i to be assigned to other document-less nodes.

We will illustrate the presented idea with one example. Figure 3.7 represents a hierarchical ontology with documents assigned to its concept nodes (the concepts instances). Let the set of documents be $D = \{d_0, d_1, ..., d_{14}\}$. The relation I is presented on Table 3.2.

- The top concept $\top$ is translated into the formal concept $\top := (A^\top, B^\top)$, with $A^\top = \{d_0, d_1, ..., d_{14}\} = D$ and $B^\top = \{d_0\}$.
- The ontology concept C_1 is translated into the formal concept $C_1 = (A_1, B_1)$ with $A_1 = \{d_1, ..., d_9\}$ and $B_1 = \{d_0, d_1\}$.
- The ontology concept C_3 is translated into the formal concept $C_3 = (A_3, B_3)$ with $A_3 = \{d_2, ..., d_7\}$ and $B_3 = \{d_0, d_1, d_2, d_3\}$.

I	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}	d_{12}	d_{14}
d_1	x	x	x	x											
d_2	x	x	x	x											
d_3	x	x	x	x											
d_4	x	x	x	x	x	x									
d_5	x	x	x	x	x	x									
d_6	x	x	x	x			x	x							
d_7	x	x	x	x			x	x							
d_8	x	x							x	x					
d_9	x	x							x	x					
d_{10}	x										x	x			
d_{11}	x										x	x			
d_{12}	x												x	x	x
d_{13}	x												x	x	x
d_{14}	x														x

Table 3.2: FCA: The relation I.

- The ontology concept C_7 is translated into the formal concept $C_7 = (A_7, B_7)$ with $A_7 = \{d_4, d_5\}$ and $B_7 = \{d_0, d_1, ..., d_5\}$.
- The bottom concept $\perp$ is translated into the formal concept $\perp = (A_\perp, B_\perp)$ with $A_\perp = \emptyset$ and $B_\perp = \{d_0, d_1, ..., d_{14}\} = D$.

It is straightforward to verify that the concepts, as defined here, follow the rules for formal concept subsumption as introduced in definition 34. For a sketch of a proof, let us take for instance concept C_7 and C_3, for which in the ontology tree, it holds that $C_7 \leq C_3$, where the partial order $\leq$ stands for the relation `is_a`. The same relation between the two concepts holds when taken as formal concepts, i.e. $(A_7, B_7) \leq (A_3, B_3)$, since $A_7 \subseteq A_3$ and $B_3 \subseteq B_7$.

3.4 Ontology Matching: Motivations

In the broadest sense, ontology matching is the process of finding correspondences between the elements of two or more heterogeneous ontologies. Because of the nature of our approaches to ontology matching which are combined of (sub-)approaches that are themselves heterogeneous, in this section, we will not give a formal definition of ontology matching to adher to throughout the whole study. Instead, we will discuss in a broader sense different motivational questions and aspects of the problem that form the background of this field. Later on, however, when presenting our approaches, we will formally state what is understood by matching (concepts or taxonomies) within the framework of each stage of the approaches.

The following section introduces phenomena related to the possible ambiguities emerging among communities in representing semantic knowledge which, we have claimed, underlie the problem of ontology heterogeneity, presented further. Since semantic similarity is introduced to re-establish the links between different

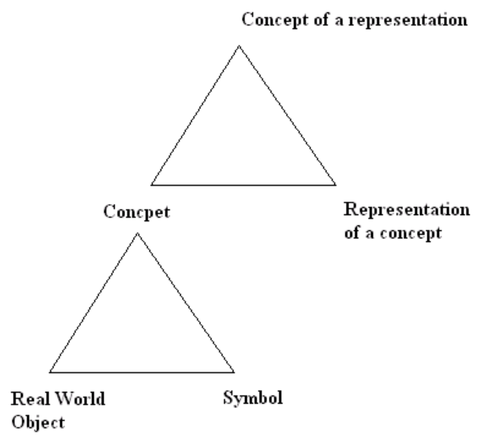


Figure 3.8: The meaning triangles of Ogden and Richards and Sowa.

conceptual representations of the same entities and thus provide basic building blocks for an ontology matching procedure, similarity will be discussed later on in relation to human concept formation and measuring semantic proximity of concepts.

3.4.1 Representation of Semantic Knowledge

In order to explain the problem of ontology matching, its motivation and possible solutions, we need to examine more deeply the representation of semantic knowledge and a phenomenon called semantic heterogeneity. Ogden and Richards [117], back in 1936, described the relation between the real world objects, the concepts (defined most commonly as mental representations) and symbols (language expressions) introducing the so called *meaning triangle*: a symbol stands for a real world object and evokes a concept; a concept refers to a real world object. Later on, in 2000 John Sowa [141] built on top of the meaning triangle the *knowledge representation triangle*, aiming to show how a person connects his concept with a certain conceptual representation. On the knowledge representation triangle's vertices we find: the concept (a vertex from the meaning triangle), a representation of a concept (which models the concept) and a concept of representation (which relates to the concept of the representation of a concept). Figure 3.8 summarizes these ideas.

Clearly, the mental representation (concept) and the choice of a symbol (word) for a given real world object may differ among different people. For example, the symbol for a given real world object, say an electric guitar amplifier, may differ because of the language different people choose to use. “Amplifier” might be the symbol chosen by an English speaking person, while it is more likely that a German native speaker chooses the symbol “Verstärker” for the same real world object. More over, even among people that have reached an agreement on the use of a common natural language, the symbolic ambiguity might still appear. It is possible that a guitar player that works everyday with electric guitar amplifiers and knows a lot about different kinds of amplifiers (for which reason he needs to distinguish between them) might call it a “Squire” (following the brand of his favorite amplifier manufacturer), someone else might still use the symbol "Amplifier" or the abbreviation "Amp".

The conceptual ambiguity appears quite often, too, since different people develop different mental representations of one and the same set of real world objects, depending on the categorization principles they decide to use and the references to the category they have among the real world objects. It is argued that this is a complex process in the core of which lie various historical and cultural conditions, as well as complex psychological processes [131]. Particularly important is the question what role similarity plays in this whole process and how is it defined and perceived, especially because similarity of concepts helps us judge on their semantic proximity. (Section 3.4.3 and Section 3.4.4 of this chapter focus on questions related to the principles of human categorization and similarity assessment.)

Finally, going up to the knowledge representation triangle, we will find different representations of one and the same real world object, since its conceptualization varies among different people. We will claim that this is where ontology heterogeneity, as defined in the introduction of the thesis, evolves from. Solving the heterogeneity problem by the definition and application of various measures of (semantic) similarity of concepts introduces rules for trnaslation between different conceptual systems - essential for the mutual understanding and interoperability between (human or artificial) agents.

3.4.2 Ontology Heterogeneity: Aspects of the Problem of Ontology Matching

Ontologies, as knowledge representation bodies, suffer the problem of knowledge representational heterogeneity described above for many reasons, mostly because of the limitations following from the decentralized and strongly human-biased nature of ontology acquisition. Ontologies are being created from different people and communities independently from one another and this process is largely manual or, in the best case, semi-automatic. In many open and evolving systems and applications with decentralized nature where ontologies are broadly applied, such as Peer-2-Peer Systems, eCommerce or the widely discussed Semantic Web [12], it is unlikely that different parties would adopt the same ontologies to represent the same fragments of knowledge. This has lead to the creation of a considerable number of ontologies, which describe similar or overlapping domains of knowledge but their elements do not explicitly match – a phenomenon that we have called in the introduction of the thesis *ontology heterogeneity*. (See Figure 3.9 for an illustration of two ontologies which share a semantic overlap).

Ontology matching amounts to reducing ontology heterogeneity and, ultimately, overcoming the barriers in front of knowledge sharing. In the beginning of the section, we spoke of the problem of heterogeneity in knowledge representation. Ontology heterogeneity originates at that problem and can occur in many different forms. Many authors have provided classifications of the different types of mismatches that can occur among schema, databases and ontologies [10, 43, 85, 139]. Following the heterogeneity typology by Euzenat and Shvaiko [43], one distinguishes between four main types of heterogeneity and we will briefly describe each of them.

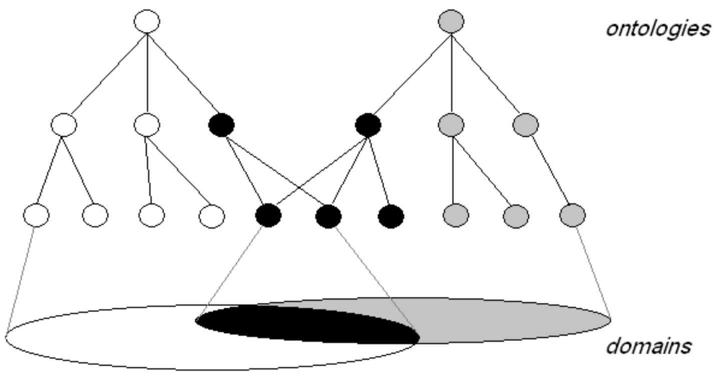


Figure 3.9: Overlapping ontologies.

Syntactical heterogeneity concerns ontologies, which are expressed in different formal languages [3]. As argued by many authors, this type of heterogeneity is among the easiest to overcome. It should be tackled on a theoretical level by defining correspondences between the constructs of the different languages.

Terminological heterogeneity is about vocabulary mismatch: differences in the choices of names when referring to the same ontological entities (concepts, relations or instances). Usually, lexical or instance-based matching techniques are applied in order to find correspondences between such entities.

Conceptual heterogeneity refers to three sub-types of differences when modeling the same domain of interest:

- Differences in coverage: two ontologies describe different or partly overlapping domains, from the same perspective and in the same detail;
- Differences in granularity: two ontologies describe the same domain, from the same perspective, but in different details;
- Differences in scope: two ontologies describe the same domain with the same level of detail, but from a different perspective.

Finally, *semiotic* or *pragmatic heterogeneity* collects mismatches in how entities are interpreted by people in a given context and is hard to model computationally.

Of course, the typology presented above is not universal and the different heterogeneity types often appear simultaneously. For instance, syntactical heterogeneity, as described here, can result in semantic differences; terminological differences, on the other hand, are also considered in various sources as syntactical; etc.

As already observed, reducing heterogeneity is achieved in terms of identifying similarity. More generally speaking, the match as an operation on structured information can be defined as an operation, which takes two ontologies as an input and produces a mapping between those elements of the two ontologies that correspond semantically to each other. Therefore, the task of ontology matching can be viewed as the identification of similarities between the different elements of two distinct ontologies, by applying a defined distance function or measure of similarity between two ontologies or their alignable elements[3] [66]. A very general definition of this process can look like that:

> Ontology matching is the process of identifying the implicitly contained similarities between the elements of two heterogeneous ontologies, which cover the same or similar domains of knowledge but their elements do not explicitly match.

We have seen that there are two notions, which lie at the core of the problem and the process of ontology mapping – the notion of *similarity* and the notion of *categorization and conceptualization*. They are strongly related and interfering concepts which have been studied broadly in many fields of natural science and humanities. Even though discussed from different angles by psychologists, philosophers and AI-people, we find it a worthwhile effort to have a glance at some prominent psychological models of human similarity assessment, category and concept formation, as well as at some philosophical discussions concerning the role of similarity in human categorization in order to provide a broader ground for the study of the problem of ontology matching in terms of conceptual similarities.

3.4.3 The Role of Similarity in Human Categorization

The relation between similarity and categorization, these two ground concepts in human cognition, has been changing in the views of philosophers and psychologists during the past decades. The intuitive idea that similarity is at the core of categorization was broadly accepted during the 1970s. The main argumentation provided by defenders of this theory like Eleanor Rosch and co-workers [131] was that similar objects are found in the same category; an object is assigned to a category, whose other member-objects it is most similar to. Categorization has inductive properties: knowing that an object belongs to a given category, we are able to infer properties of that object as long as they are common for other members of the same category (a phenomenon referred to by Sloutsky as *inductive generalization* [140]).

In the mid-seventies, Nelson Goodman, as cited by Quesada [124], criticized this view by putting forward the argument that similarity as a concept is itself vague and ill-defined. The philosopher rejected the validity of the intuitive statement that two objects are similar and thus belong to the same category because they have many common properties. Any two objects, according to Goodman, have infinitely many similar and infinitely many dissimilar features. For example, a feather and a cannonball have the common property that they weigh less than 100 kg and also less than 101 kg, etc. Therefore, it makes sense to talk about similarity only if we have specified in what respect we judge. Thus, categorization is about "respects" and not about similarity. In relation to that, in the 1980s it became a popular view that categorization cannot be accounted for by similarity, but rather by *our theories* about the kind of things (features, properties) that an object can be characterized by. These theories form explanatory systems around which our categories – fragments of our world knowledge – are organized. A series of empirical studies by Rips providing

[3]By alignable elements we mean elements that correspond to the same component of an ontology definition. Alignable are the sets of concepts of two ontologies, but not the set of concepts of one ontology and the set of relations of the other.

evidence for the validity of this “theory-based” model of conceptualization is discussed by Hampton in [67].

However, despite the harsh criticism of the similarity-based approach to categorization, similarity did come back to the categorization scenario again. As Sloutsky [140] and Hampton [67] argued, the theory-based model is problematic for it does not explain where does the conceptual knowledge which organizes the things in the world in categories originate. Most probably, they claim and provide in their turn experimental evidence, it is based on simpler perceptual components. Thus the focus has been switched back on enhanced similarity based approaches which make up for the drawbacks of the classical similarity account of categorization.

3.4.4 Accounts of Human Similarity Assessment

We will focus on a couple of standard and some recent approaches to modeling human similarity assessments – the principles that lie in the basis of how humans judge on two entities as being similar or dissimilar. Currently, there are several big groups of similarity models that contain approaches which are considered to be psychologically and cognitively plausible. We will go into four of them – *spatial* models, *feature* models, *hierarchical structure* models and models based on *transformational distance*. The models from each of the four groups have their strengths and their weaknesses and none of them is seen as the only generic similarity model, but rather as a model which is efficient with a given respect. Therefore, we will emphasize a set of relatively recent approaches which are particularly relevant for ontology engineering, semantic web applications and text categorization, as pointed out in a comparative study by Jose Quesada [124]. A more profound overview of the existing semantic similarity models is found, for example, in the doctoral thesis of A. Schwering [135].

A discussion that compares our approach to ontology matching and semantic similarity to the accounts of similarity presented below and situates it among them is found in Section 5.4.2 of the thesis.

Spatial models

The idea that internal representations of external stumili can be modeled by using a metric space was initially proposed by Roger Shepard in the late 1950's [138]. A metric space is defined by a metric, or a distance function d which takes two elements of a set and returns a real non-negative number – their distance – and has the properties minimality ($d(x,y) \geq d(x,x) = 0$), symmetry ($d(x,y) = d(y,x)$) and triangle inequality fulfillment ($d(x,y) + d(y,z) \geq d(x,z)$, where x, y and z are three elements of some abstract space; see definition 13 from Section 2.1.)

Multidimensional scaling (MDS) was proposed by Shepard and co-workers as an approach to represent closeness of data in a metric space. The main idea is that objects are represented as vectors in a multidimensional space, where the dimensions of the space are continuous features of some kind. The distance between two objects in a n-dimensional space is given by Minkowski's formula

$$d_{mink}(x_i, x_j) = (\sum_{k=1}^{n} |x_{ik} - x_{jk}|^r)^{\frac{1}{r}},$$

where r is a parameter to be fixed and x_{ik} is the k-th dimension value of x_i and x_{jk} is the k-th dimension value of the element x_j.

A prominent follower of the MDS methods is the Latent Semantic Analysis, discussed in Chapter 2 - a spatial model of similarity between documents and words which is based on projecting data onto dimensions defined by distinct contexts in text corpora (a *semantic space*). However, a serious drawback of MDS and LSA in particular in the context of semantic web applications is that they are based on using statistical information, whereas most of the data on the semantic web is written in some kind of a formal language and controlled by logical operations.

Feature models

Amos Tversky, among others, is known as a firm opponent to the spatial models. He provided empirical evidence that the three main assumptions that define a distance function are often broken when it comes to human judgments of similarity [153].

Minimality violations have been observed on judgments of the similarity of complex objects as opposed to the similarity of simpler ones. The more complex structure two identical objects have, the more they seem to be similar as compared to simpler identical objects.

Violations of the *symmetry* assumption have been pointed out by an example which now has become classical. Subjects were asked to judge the similarity of North Korea to China and the similarity of China to North Korea. The two countries have been more often judged to be similar in the first case than in the second one.

Finally, arguments that *the triangle inequality* does not necessarily hold for any three objects have been provided, as well. An example in support of that critique of the spatial model is the fact that although a lamp is similar to the moon, and the moon is similar to a football, a lamp and a football still have very little in common.

Evolving from these criticisms, feature models are based on the assumption that objects are assessed as sets of features. The similarity of x and y is measured as a function of the intersection and complements of their corresponding sets of features X and Y[4]:

$$\sigma_{tversky}(X,Y) = f((X \cap Y), (X \setminus Y), (Y \setminus X), \Theta, \alpha, \beta). \quad (3.1)$$

We recall that $\cap$ denotes set intersection and $\setminus$ denotes set difference (Figure 3.10). Θ, α and β are parameters which are most commonly set empirically and serve to attribute greater or smaller weights to either of the three set-related quantities.

We will pay some more attention to a recently proposed feature model which is among the first ones that attempt to operate on text corpora. The *Topic model* introduced by Griffiths and collaborators a few years ago [60] focuses on the relations between words in language use on the one hand and the relations between words and concepts, on the other hand. The topic model is centered around the idea that there exists a probability distribution from which a set

[4]The given formula is a general statement of Tversky's similarity, where it is not indicated how do the entities precisely relate.

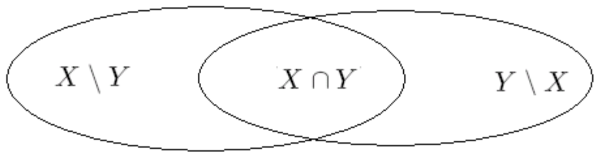
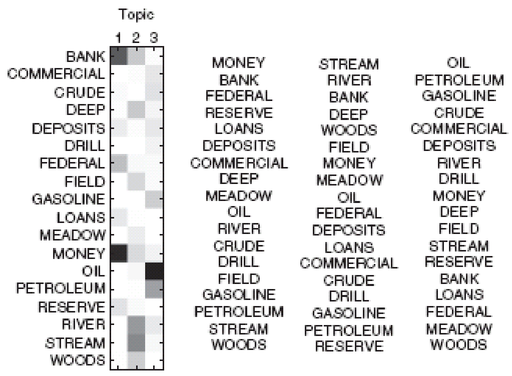


Figure 3.10: Tversky's feature model.



MONEY	STREAM	OIL
BANK	RIVER	PETROLEUM
FEDERAL	BANK	GASOLINE
RESERVE	DEEP	CRUDE
LOANS	WOODS	COMMERCIAL
DEPOSITS	FIELD	DEPOSITS
COMMERCIAL	MONEY	RIVER
DEEP	MEADOW	DRILL
MEADOW	OIL	MONEY
OIL	FEDERAL	DEEP
RIVER	DEPOSITS	FIELD
CRUDE	LOANS	STREAM
DRILL	COMMERCIAL	RESERVE
FIELD	CRUDE	BANK
GASOLINE	DRILL	LOANS
PETROLEUM	GASOLINE	FEDERAL
STREAM	PETROLEUM	MEADOW
WOODS	RESERVE	WOODS

Figure 3.11: The topic model (taken from [60]).

of words has been generated that can be inferred, called a topic. The *gist* of any set of words (like, for example, a document) can be represented by the means of a probability distribution over a set of topics. The words that are assigned a higher probability by a given topic are those that reflect that topic's content. An example, taken from the paper *Topics in semantic representation* by Griffiths *et al* [60] is shown in Figure 3.11: words are organized by probability distributions (topics); the probabilities of words under three different topics is shown in the table (left); on the right, we see the words contained in each topic ordered by probability with respect to a topic. A set of topics can be learned automatically from text corpora by the help of Bayesian statistical methods, which, according to the authors, throws a bridge to the way in which humans would form semantic representations when presented with the same collection of texts.

The topic model suggests increased association between two words if a topic assigns high probabilities to these words and decreased if this topic assigns high probability to one word, but not to the other. In that respect, the analogy with feature models based on equation (3.1) is relatively straightforward, because feature models are equally based on both common and distinctive features in order to claim similarity of entities.

Quesada [124] observes that the topic approach is particularly relevant for ontology mapping and semantic web representation problems, because it combines statistical learning with structured representations – an advantage of the method compared to Latent Semantic Analysis or semantic network approaches (not discussed here, see [28] for a reference).

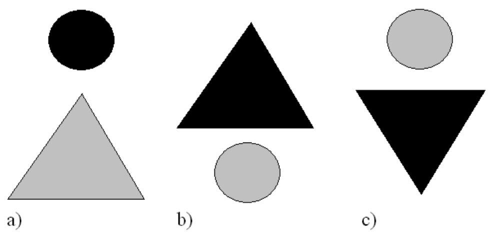


Figure 3.12: SR: Structured representations.

Hierarchical models of structured representations

Hierarchical models, also known as *alignment models*, assert that in judging the similarity of two objects or concepts not only common and distinctive features are important, but also information related to the structure of the objects and concepts to be compared. In other words, judging similarity involves a process of structural alignment. These claims were supported by a number of experiments (see, for example, [100]), which have shown that in assessing similarity the relation between objects and the relation between relations play an important role.

Let us have a look at the series of pictures a), b) and c) presented in Figure 3.12. By simply comparing features, we would have to operate with sets containing elements like “black”, “gray”, “round”, “triangular” and “medium-sized”. The structured representation model suggests that the relations between the elements of each object or concept, such as “above”, “under” or “rotated” are to be taken into account. Indeed, the features of each entitiy are organized in a hierarchy which preserve and represent the structure by the help of these relations (an example is shown on the diagram in Figure 3.13). The similarity is then assessed in terms of a series of alignments, first – on relational, and then – on feature level.

Although models based on structured representations are at least intuitively closest to the data structures on the semantic web, their application is still not evident. The main limitations of these approaches originate at the fact that their generality (performance on larger data samples) has not been empirically tested. In addition, the structured representations used in psychological experiments are manually generated and it is not clear how extracting structure representations can be done automatically within this framework [124].

Transformational distance models

It has been suggested that the distance between two entities x and y can be expressed as a function of the minimal number of operations that are needed in order to transform the entity x into an entity x' which is identical to y. Such a function is called a transformational distance between x and y.

Transformational distance models are among the most general similarity models. They are based on the assertion that two entities are similar if their transformational distance is small [65].

We have discussed the *edit distance* in the introduction to graph theory and the related *string edit operation* in the introduction to text categorization (Chapter 2). These techniques provide examples of transformational distance

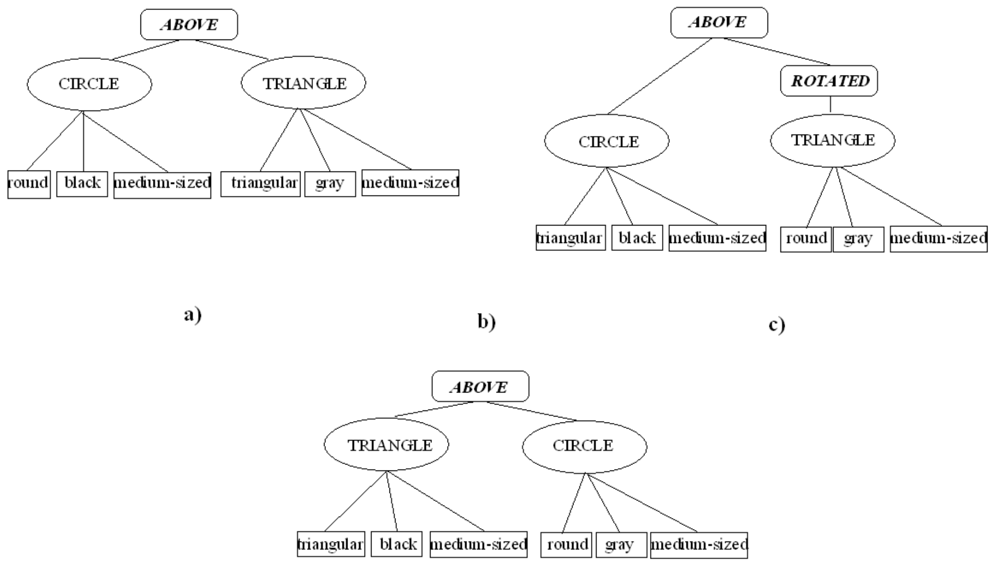


Figure 3.13: SR: Hierarchically organized features.

models. Among other promising approaches, we will outline the *syntagmatic paradigmatic (SP) model* [36] which is also related to string edit approaches. It is as well a presumably good candidate to solve semantic representation problems on the semantic web, the main reason for which lies in the fact that the SP model works from plain text and the expensive step of structuring flat data is not required [124].

The SP model has been recently proposed by Dennis and colleagues [35]. The suggestion is that sentence processing in general language acquisition is characterized as retrieval from memory [36]. The interpretation of verbal stimuli is done through two types of word associations – *syntagmatic* associations of words that tend to appear together in the same sentence (like "read" and "book"), and *paradigmatic* associations of words which tend to appear in the same context, although not necessarily together in the same sentence (like "book" and "article"). In language acquisition, a target sentence is aligned to sentences retrieved from memory by the help of sets of syntagmatic and paradigmatic constraints. The set of thus retrieved sentences constructs the interpretation of the stimulus sentence. Two sentences are assessed as similar or as conveying the same information if pairs of words from both sentences are aligned to the same sets of words from the sentences retrieved from memory through SP associations, i.e. if they have the same interpretation.

3.4.5 Terminological Disambiguation

We will close this section with some remarks on the terminology used by ontology matching researchers and practitioners.

The study of ontologies, especially in the context of Internet and web-based applications, is a relatively new field of Artificial Intelligence and Computer Science. This is one reason why, anyone who has been confronted with research papers written by different authors on the topic will have noticed a certain lack of agreement on a common terminology. Particularly speaking of ontology matching, it is a common thing that different authors use different terms in order to address similar or identical concepts. Since we do not make an exception from this practice, in this section, we will attempt to disambiguate between the different terms used to denote ontology matching and similar techniques by defining them in the way that they are used throughout this work. A broader list of terminological definitions is found in the Ontology Matching book by Jérôme Euzenat and Pavel Shvaiko [43].

Ontology Matching is the most general term standing for the process of searching and identifying correspondences between the elements of two distinct ontologies and the relations that hold between them. Throughout this thesis, we also use this term in the most general sense to denote a procedure or an algorithm for reducing heterogeneity between two source ontologies.

Ontology Mapping is a stricter notion than matching, related more closely to the mathematical concept of mapping. Ideally, a mapping takes the elements of one ontology and, through a defined mapping function, finds their image on the set of elements of a second ontology. Throughout this thesis the terms mapping and matching are used in a more or less interchanging manner. However, a priority is given to the use of the term "mapping" as referring to finding correspondences among two sets of cross-ontology **concepts**, whereas "matching" denotes rather the general process of *mapping* the concepts and other elements of two **ontologies**.

Ontology Alignment is a structured set of correspondences between the entities of two or more ontologies. "Structured" indicates that correspondences are looked for only among sets of entities of the same type (for example concepts, relations, instances, etc.).

Ontology Reconciliation, as the name indicates, aims at reducing the differences between two (or more) ontologies, which requires changes in one or in all input ontologies, until their content is harmonized. This term is used by defenders of the idea that the creation of a globally consistent ontology is not only impossible but also not necessary; instead, ontologies have to be enabled to co-operate locally, only with respect to a given application [66].

Ontology Merging, finally, is often of a very big practical interest and consists, contrarily to ontology reconciliation, in the creation of a new ontology from two (or more) source ontologies, which have some semantic overlap [144], by executing a matching or alignment procedure.

3.5 A Classification of Ontology Matching Techniques and Approaches

We will review the state of the art in the field of ontology matching by outlining the most relevant with respect to our study theoretical and practical endeavors. In doing so, we will group similar approaches together by first presenting two classifications of ontology matching approaches. The first one is provided by Rahm and Bernstein (2001) and is among the most general available frameworks [125]. The second one builds on the first one and was published by Euzenat and Shvaiko in their ontology matching book (2007) [43].

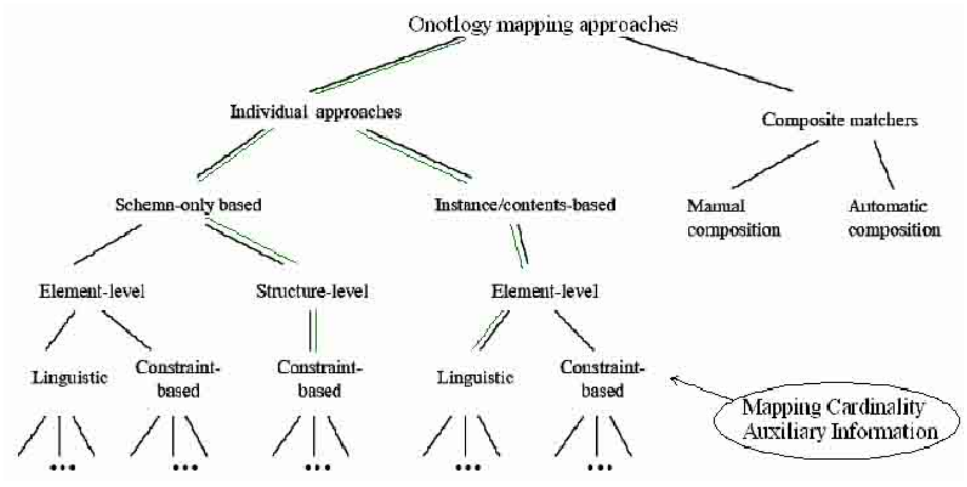


Figure 3.14: Matching approaches (Rahm and Bernstein).

Rahm and Bernstein's Classification

In 2001, Rahm and Bernstein [125] published a classification of schema matching approaches which can be seen in Figure 3.14. We will explain the nodes of the classification tree and will position our approaches within the proposed framework. Note that, in the diagram, we have replaced the word "schema", originally used by the authors, by the word "ontology"; however, both names will be used in the remainder of the section.

On the down most level of the taxonomy we find opposed *Linguistic* and *Constraint-based* approaches. Linguistic approaches rely on names and textual description of ontology elements and their relations. Most of the concepts in an ontology contain a textual definition – a natural language description defining them, which enables a linguistic evaluation of the similarities. Constraint based approaches make use of relationships.

On the next upper level of the matching approaches tree are classified *Element-level* versus *Structure-level* techniques. This discriminates mappings performed on single ontology entities from those performed on entire ontological structures.

Further on, Rahm and Bernstein consider that schema matching approaches can take into account either only information contained in the structure of the ontologies – *Schema-only-based*, or they can as well make use of the instances contained in these ontologies, or their data content – *Instance / Content-based*.

The highest level class-separation in Rahm's classification tree concerns the use of multiple matchers as opposed to individual matching techniques. One one hand, one may decide to use an *individual approach*, which relies on only one mapping criterion. On the other hand, one could apply a *composite matcher* that can be *manually* or *automatically* composed from several individual matching approaches.

Finally, the result of the mapping may relate one or more elements from

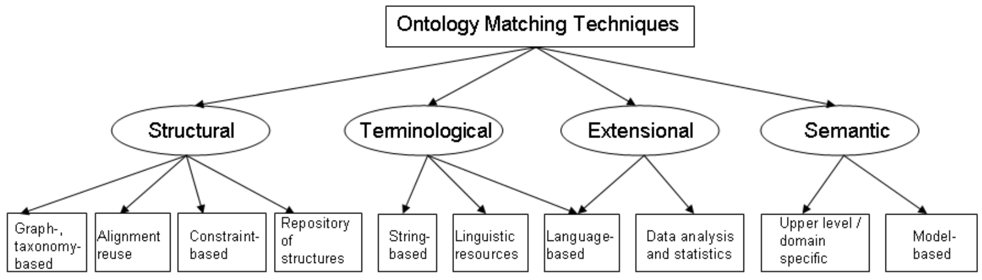


Figure 3.15: Matching approaches (Euzenat and Shvaiko (a fragment)).

the first schema to one ore more elements of the second one. That defines the *mapping cardinality* which results in four possible cases 1:1, 1:n, n:1 and n:m map. During the matching procedure, *auxiliary information*, such as dictionaries, thesauri or previous matching descriptions, can come in use, too. Thesauri help explore and use synonyms and super- or sub-ordinate relations between terms. Dictionaries are useful for translating from one language to another, which is particularly helpful when dealing with ontologies in multilingual environments, as well as for providing explicit semantics of concepts in terms of linguistic definitions.

The ontology matching procedures that we are about to present in the next chapter of the thesis are composite approaches, because they combine two different ontology matching criteria – structural and instance-based. In Figure 3.14, the path that these approaches follow in Rahm and Bernstein's tree is traced in double line[5].

Euzenat and Shvaiko's Classification

Euzenat and Shvaiko, in their 2007 book [43], provided a very detailed classification of ontology matching approaches, which builds on Rahm and Bernstein's taxonomy, but is more exhaustive and granular, taking into consideration the advances made in this dynamic field in the time gap between the two publications. In building their classification, the authors have used as guidelines four main criteria: *exhaustivity* (the sub-categories of a given category all together contain exactly the extension of that category), pair-wise *disjointness* of the categories, *homogeneity* of classes and, finally, adding and modifying classes until *saturation* has been reached.

In Figure 3.15, we have presented a fragment of Euzenat and Shvaiko's classification which we consider to be granular enough for the purposes of our study and for giving a ground for situating our approach and related mehtods. We will discuss in more details each of the different categories of methods by putting a closer focus on approaches which are directly relevant to ours.

[5]The double-line path in Figure 3.14 passes through individual approaches, but goes further down in two different directions, showing that it combines two individual approaches.

3.5.1 Structural and Terminological Approaches

Structural and terminological approaches are often used in combination. In the current subsection, we will also discuss them together, giving more attention to structural techniques to emphasize their importance and relevance to our approaches.

Basic assumptions and similarity measures

Terminological methods comprise two major groups of approaches – those that use strings in order to match names of entities, and those that rely on linguistic information contained in dictionaries and thesauri combined with techniques from Natural Language Processing in order to compare the similarity of terms and their relations and overcome problems evolving from *synonymy* and *polysemy*. Both groups of techniques have been discussed previously in this thesis (see Chapter 2, Section 2.1.3 and Section 2.4.4) and, therefore, we will not delve into further details. Instead, we pay more attention to structure-based mapping techniques, also because structural similarity is central to our approach.

The structure of an ontology can be studied on two different levels with respect to either a single ontology element, known as *internal structure*, or the way in which a set of elements are related, known as *relational structure*. Methods based on the former structure type look into similarities of the sets of properties of two elements, the datatypes used to describe them or their properties, the cardinalities that sets of values of two properties are "allowed" to reach, etc. These approaches are suited to schema matching problems where one disposes readily with an internal structure of the database entities.

In our approach, however, we do not rely on the explicit availability of such structural information, but we are rather concerned with comparing blocks of elements together with the relations that hold between them. This is exactly the focus of relational structure based methods; the matching problem is typically situated in a graph-theoretical framework where an ontology is modeled as a graph whose vertices and edges are labeled by concepts and relations names, correspondingly. The matching of such "ontology graphs" is usually reduced to solving a graph isomorphism problem and identification of a maximal common sub-graph of two graphs. All graph-matching techniques discussed in Section 2.1.3 and in the sources cited there are suitable for approaching this problem. However, we will look closer at methods and measures designed specifically for matching tree-structured ontologies for two reasons: first, a taxonomy (a tree of concepts related by subsumption) is the backbone of an ontology and a lot of theoretical and practical efforts have been dedicated to mapping taxonomic structures, and, second, our procedures for ontology mapping rely heavily on the hierarchical relation of concepts within the source ontologies, generalizing to non-hierarchical structures on the basis of the underlying hierarchies.

Most approaches to ontology matching, which have a hierarchical structure similarity measurement component, are based on looking into the structures of two or more input ontologies separately from one another, and not to comparing their structures explicitly. The measures of concept similarity that they utilize concern concepts within one single hierarchy. Why is that nevertheless interesting for matching different ontologies?

The common intuition is that as long as two classes from two distinct ontologies have been identified as similar (by the help of terminological or other approaches), based on the relation of each of these classes with the rest of the classes in their taxonomies, we can trace up or down in the hierarchies other pairs of similar classes. Below, we will give definitions of some concept similarity measures and distances which have been applied to these ends so far.

Let $H = (C, \leq)$ be a hierarchy on the set of concepts C. The most intuitive method of evaluating the proximity of two nodes within a single hierarchy is to measure the length of the shortest path connecting them (see, for example, [43]).

Definition 35 *The shortest path distance between two elements $c', c'' \in C$ is given by*

$$\delta_{path}(c', c'') = min_{c \in C}(n_e(c', c) + n_e(c'', c)),$$

where n_e returns the number of edges between two nodes.

Wu and Palmer, as referred to in [43], noticed that simply counting the edges between two classes is not a valid criterion: nodes closer to the root of a hierarchy might be totally dissimilar, although very close in terms of path distance. They proposed a new measure of similarity which takes into account that observation.

Definition 36 *The similarity of two elements $c', c'' \in C$ is measured by*

$$\sigma_{wp}(c', c'') = \frac{2n_e(c' \wedge c'', root_H)}{n_e(c', c' \wedge c'') + n_e(c'', c' \wedge c'') + 2n_e(c' \wedge c'', root_H)},$$

where $root_H$ denotes the root of the hierarchy H, $c' \wedge c''$ is the least common subsumer of c' and c''[6] and n_e is as defined in definition 35.

As we can see, the denominator of σ_{wp} in definition 36 is very similar to δ_{path} from definition 35[7]. Wu and Palmer's measure of similarity is based on the previously introduced path-length distance, scaled by the path-length distance of the nodes least common subsumer to the root of the hierarchy, expressed by the quantity $n_e(c' \wedge c'', root_H)$.

Finally, as we will see below, the Jaccard similarity coefficient is widely used for measuring instance-based similarity of ontologies. However, a structural similarity measure which applies it has been also introduced [43].

Definition 37 *Let $C^+(c)$ denote the set of superclasses of $c \in C$. (Conventionally, $c \notin C^+(c)$.) The similarity of two elements $c', c'' \in C$ is given by*

$$\sigma_{jac-str}(c', c'') = \frac{|C^+(c') \cap C^+(c'')|}{|C^+(c') \cup C^+(c'')|}.$$

In the following, we will discuss several state-of-the-art approaches that combine terminological with structural similarity criteria for matching ontologies and utilize the measures and distances introduced above.

[6]Note that using the notation $c' \wedge c''$ is a way of avoiding the longer $min_{c \in C}$ notation introduced in the definition above.

[7]Their roles of numerator and denominator in the two formulas are inversed, for one is a measure of *similarity* and the other – of *dissimilarity*

Related approaches

The Anchor-Prompt algorithm. The system was developed and implemented by Natasha Noy and colleagues in the beginning of the decade and has since been recognized as an important state-of-the-art contribution to the field [114].

Anchor-Prompt uses the common graph representation of ontologies – each class corresponds to a node of the graph, an arc between two nodes exists whenever a relation between the corresponding classes is available. The algorithm starts by manually or automatically selecting a set of pairs of similar concepts from both ontologies – the so called "Anchors" – usually identified through lexical matching. The main idea which the procedure further builds on is that if there have been found two pairs of similar concepts (two "Anchors") and there exists a path connecting the concepts in each of the two ontologies, it is very likely that the entities found on those paths are also similar. The algorithm works by simultaneously traversing the paths and testing the similarity of the classes found on each step of the traversals.

The Cupid system. As we have seen in the previous section, Bernstein and Rahm have contributed a lot for providing a systematical account of the existing ontology matching approaches. Cupid, the system that they proposed together with Madhavan builds on many of the existing to the date of its development schema matching approaches, including linguistic-based, element-based and structure-based matching [96]. It is a generic matching approach, which has not been developed in the light of a particular intended application.

Cupid works in three steps. The first step is the linguistic analysis of the individual schema elements (names, data types, etc.). The system uses a thesaurus to identify synonyms and abbreviations. The second phase consists in a structural matching. Nodes in the schema-graphs whose parents have been matched in the first step are considered as matching candidates. In both phases, a similarity coefficient is calculated for each match-pair. The final step generates the actual mapping by picking up those pairs of potential mappings which have maximal similarity coefficients.

The Onion tool. Mitra and Wiederhold [107] argued against the need of constructing and maintaining a global consistent ontology as a result of an ontology matching procedure. Instead, they proposed an approach to *ontology-composition* which assists the linking of certain parts of the input ontologies composing smaller domain ontologies only for the needs of a given application. The Onion tool for *ontology articulation* which they presented serves to enable querying simultaneously multiple ontologies represented as labeled graphs. It is based on an algebra for knowledge composition, featuring operators like *union*, *intersection*, *difference*. The alignment consists of a set of *articulation rules*.

Figure 3.16 provides a sketch of the main ideas behind Onion. Considered are two sources of information, "Factory" (Source 1) and "Store" (Source 2). They must share information in order to enable queries, such as "Purchasing" (Application context 1).

The Chimaera tool. Proposed by McGuinness and colleagues in the early 00's [102], Chimaera aides an ontology engineer in a process of merging two or

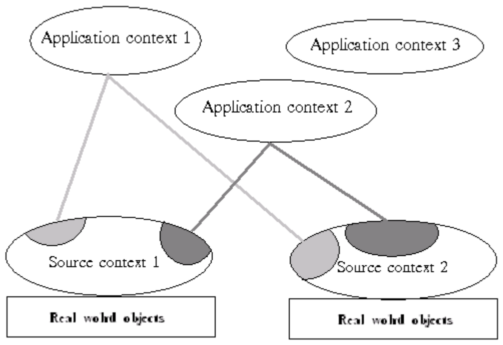


Figure 3.16: The ONION tool (adapted from [107]).

more source ontologies. It relies mainly on terminological criteria (similarities of the vocabularies used for concepts indicates similarity of their semantics) in combination with relation-based criteria (using the taxonomic structure of the terms in order to judge on the semantic nature of the respective concepts).

The authors make the observation that ontologies are handled predominantly by non-experts in knowledge representation and engineering – a reason to focus on user-friendliness in the development of their tool. In an easy and intuitive to work on user interface, the tool suggests to the engineer sets of potentially similar concepts-candidates for merging (for example, the concept "mammalia" from one ontology and the concept "mammal" from another).

A newer version of the tool contains a diagnostics component, which evaluates (partial) correctness and completeness of the resulting merged ontologies.

3.5.2 Extensional Approaches

Basic assumptions and similarity measures

Extensional ontology matching, also known as instance-based matching, comprises a set of theoretical approaches and tools for aligning two or more heterogeneous ontologies based on their extensions – the instances that populate their concepts. The question of how a set of instances of a concept is defined will be considered in Chapter 4 of this thesis. We mention here that in hierarchically structured data, a concept can be defined as a set of those instances that are directly assigned to it, or as a set of all instances assigned to the concept and its successors in the taxonomy, what we will later call *hierarchical* and *non-hierarchical* instantiation. An example of a hierarchical instantiation is given in Figure 3.17, D_1, D_2 and D_3 are the sets of instances of the leaves of the tree. The very question of what is an instance – an object, a word, a text document, an image, etc., will be also discussed in Chapter 4, because it is relevant to a particular approach and application, not in general. For the moment, we will content ourselves with using an abstract notion of an "instance", understood as some kind of a real-world data entity, member of a given class within an ontology.

Various concept similarity measures have already been proposed together with matching systems that employ them. Often in applying these measures it is necessary to consider a set of thresholds controlling data sparseness or

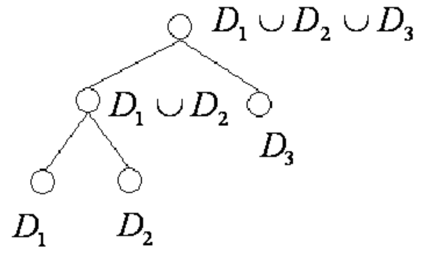


Figure 3.17: Hierarchical instantiation of a taxonomy.

defining the degree of similarity. For an overview of instance-based mapping in terms of measures, thresholds and types of concept instantiation, we refer to the empirical study carried out by Isaac *et al.* [72].

A common approach to modeling concepts by their instances is the set-theoretic approach. The relatedness of a pair of concepts is an outcome of a properly chosen measure of similarity, based on estimations of the intersections of two sets of instances. Two concepts A and B are considered similar when $A \cap B \approx A \approx B$ and dissimilar when $A \cap B = \emptyset$. The problematic aspect of this technique is, obviously, the very strong assumption of presence of identical instances in both instance sets which would ensure the presence of a non-empty intersection.

Alternatively, a measure of concept similarity can be introduced on the basis of the cosine similarity measure between two class centroids. This implies modeling a class by the average vector of the instances that it contains. We will call this measure *prototype similarity*, given by

$$sim_{proto}(A,B) = s\Big(\frac{1}{|A|}\sum_{j=1}^{|A|}\mathbf{i}_j^A, \frac{1}{|B|}\sum_{k=1}^{|B|}\mathbf{i}_k^B\Big). \tag{3.2}$$

where $s(\cdot,\cdot)$ is the standard cosine similarity and $\mathbf{i}^A$ is an instance of the concept A. A drawback of that approach is that it tends to flatten the structure and, although appropriate for leaf nodes, is prompt to fail for higher level concepts.

Among the most popular choices of a similarity measure is the *Jaccard coefficient* [39] as well as a couple of standard statistical measures which have been already applied for extracting semantics out of natural texts based on term co-occurrence, such as *mutual information* (MI) and *information gain* (IG), which have been discussed in Section 2.4 of the thesis. The Jaccard coefficient similarity is defined as it follows.

Definition 38 Jaccard Coefficient. *Let $P(X)$ be the probability of a random instance to be an element of X and let A and B be two sets. The Jaccard coefficient is a similarity function defined as:*

$$\sigma_{Jacc}(A,B) = \frac{P(A \cap B)}{P(A \cup B)}. \tag{3.3}$$

Isaac *et al.* [72] observed that the similarity defined by (3.3) does not take into account the relative “quality” of the matching between A and B. A and

B are likely to be judged equally similar if the match (A, B) is based on 100 instances or on only one. To correct this problem, Isaac proposed the *corrected Jaccard similarity* in which less frequently co-occurring annotations are assigned a smaller score (the factor 0.8 is chosen on the bases of a series of empirical tests)[8].

$$\sigma_{Jacc-corr}(A,B) = \frac{\sqrt{P(A \cap B) \times (P(A \cap B) - 0.8)}}{P(A \cup B)}. \tag{3.4}$$

Finally, note that $P(A \cap B) = P(A,B)$ and $P(A \cup B) = P(A,B) + P(A,\overline{B}) + P(\overline{A},B)$, where the entity $P(A,B)$ denotes the probability that a random instance belongs to both A and B (the *joint probability* of A and B). The task of measuring the similarity of two concepts is then redefined as finding an appropriate estimation of each of their four joint probabilities, based on the instances that they contain. We will see a way to do this in one of the related approaches [39] described in the next section.

Related approaches

We will review several instance-based approaches which we find intriguing and central to the state-of-the-art of the field. A more detailed description will be given to those of them which are to a higher degree relevant to our matching approach.

The GLUE tool. In their paper *Learning to map between ontologies on the semantic web* from 2002 Doan *et al.* introduced the instance based inter-taxonomy mapper GLUE, based on machine learning techniques for semi-automatic derivation of concept similarity assertions [39]. The method is based on one central claim, which is that we need not to commit to a particular definition of similarity. Once we have an approach to calculate the joint distributions of the classes, a similarity measure of any kind can be computed over these terms.

We will present how this is done in Doan's paper. Let $O_1 = (C_1, \texttt{is\_a})$ and $O_2 = (C_2, \texttt{is\_a})$ be two ontologies, let I_{O_1} and I_{O_2} be the sets of instances belonging to the ontologies correspondingly and let $A \in C_1$ and $B \in C_2$ be two different concepts of each ontology, viewed as sets of instances. A term like $P(A,B)$ can be estimated by the fraction of instances that belong to both A and B. The task is thus reduced to deciding for each instance whether it belongs to $A \cap B$ or not. Given that the input includes instances of A and instances of B separately from one another, the problem can be addressed by machine learning techniques for classification (such as the Support Vector Machines (SVM), the k-Nearest Neighbors (kNN) or the Naive Bayes learner (NB), used by GLUE) to create a classifier on A and apply it on B and vice-versa.

The quantity $P(A,B)$ is approximated by using the cardinalities of both sets and their intersections in each ontology.

$$P(A,B) = \frac{|A \cap_{O_1} B| + |A \cap_{O_2} B|}{|I_{O_1}| + |I_{O_2}|}, \tag{3.5}$$

[8]Isaac *et al.* [72], as well as other authors [162] use the cardinalities $|X|$ instead of the probabilities $P(X)$, used by [39] and [150], to formulate the Jaccard similarities. It is straightforward to prove, however, that both ways of notation are equivalent.

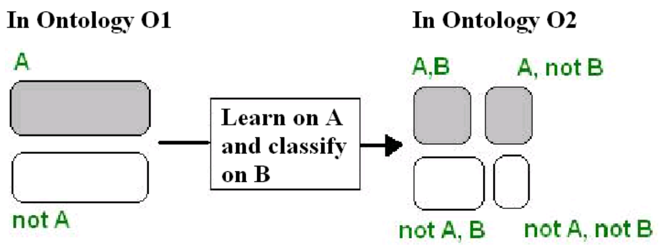


Figure 3.18: Approximating joint probabilities by a machine learning classifier.

where $\cap_{O_1}$ denotes that we are intersecting instances belonging to O_1 only. The probability of an arbitrary instance to belong to both A and B can be approximated in terms of the number of instances of A and B in both ontologies and the whole number of instances contained together in O_1 and O_2 [156].

Let us consider $B \in O_2$. We separate the instances of O_2 into instances that belong to B and that do not belong to B. We do the same with the instances in $A \in O_1$ and use the separated data in O_1 as positive and negative examples on which a machine learning classifier is trained. The classifier is applied on the instances in B and thus we come up with an estimation of the quantity $|A \cap_{O_2} B|$. We repeat the procedure inversing the roles of A and B in order to obtain $|A \cap_{O_1} B|$ (Figure 3.18). The same algorithm is applied for the other joint probabilities until we have approximations of all three of them. The similarity between the concpets A and B is given by the Jaccard coefficient (3.3) estimated as a function of the joint probabilities.

Further, GLUE computes the $|C_1| \times |C_2|$ similarity matrix of all possible pair-wise concept similarities. In order to find the best possible mapping configuration, the procedure combines the observed similarities with domain-specific constraints and heuristic knowledge.

A related to GLUE approach, based on Support Vector Machines and using literature-based corpus was proposed by Lambrix *et al.* in [89].

The FCA-merge approach. FCA-MERGE, based on Formal Concept Analysis (introduced in Section 3.3.2) was proposed by Stumme and Mädche [144] and is another approach which relies on the assumption that two ontologies use the same instances taken from a set of text documents relevant to both ontologies. It provides its own mechanisms of extracting instances of concepts from text corpora, answering a basic critique to some instance-based approaches that rely on common instances, that source ontologies are unlikely to share the same sets of instances. The approach applies natural language processing techniques and FCA to derive a concept lattice which is further transformed into a merged ontology. As this summary of FCA-MERGE suggests, a translation of an ontology into a concept lattice and back to an ontology takes place during the merging procedure.

The generation of a formal context from an ontology is done as it follows. For each ontology O_i (as defined in Section 3.1), a unique formal context $K_i := (G_i, M_i, I_i)$ is generated in the following manner. The set of documents D_{O_i} corresponding to O_i is taken as the object set of the formal context, i.e. $G_i := D_{O_i}$ and the set of ontology concepts is taken as an attribute set, i.e. $M_i := C_{O_i}$.

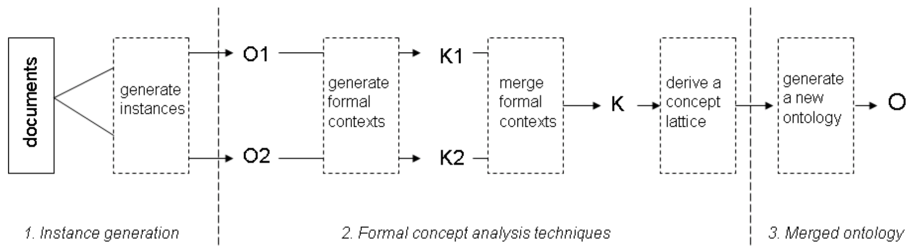


Figure 3.19: FCA-MERGE.

The relation $(g, m) \in I_i$ holds whenever a document $g \in D_{O_i}$ contains an instance of $m \in C_{O_i}$. By using FCA techniques, a concept lattice involving both ontologies is generated from merging the formal contexts corresponding to each of them. Finally, a merged ontology is derived from the concept lattice.

The method is sketched in Figure 3.19 in three steps. Step 1 consists in generation of instances (can be skipped if the source ontologies share the same instance set); the application of FCA techniques for deriving a merged concept lattice takes place in step 2; Step 3 generates an ontology out of the merged concept lattice.

Mapping via classification. An interesting recent approach to concept mapping was proposed by *Wang* and colleagues [161]. Its essential idea consists in replacing the mapping problem by a classification one by introducing a *similarity space* in which every point represents a pair of matched concepts. Labeling correct matches with a positive label and incorrect ones – with a negative label, allows for a machine learning classifier to learn on training data[9] and automatically classify new pairs of concepts as mappings of similar concepts or not.

The method relies on the fact that each instance is represented by a set of features (if it is a book, these could be the *title*, the *author's name*, the *publisher*, etc.). Concepts, on their turn are also represented as sets of features by extracting information about the term occurrences of each of the features of the instances that are contained in this concept. In our book example, a concept will contain the features *title*, *author's name*, *publisher*, each of which will be represented as a vector of the frequencies of occurrence of each of the values of *title*, *author's name* and *publisher* as features of the instances. In other words, every concept feature contains counts of the occurrences of all values of the instance features. Finally, for a given pair of concepts, a feature vector is constructed to reflect the mapping of that pair of concepts. Every component of this feature vector is the cosine distance between the corresponding concept features (e.g. the cosine between all "title" term-occurrences in concept 1 and all "title" features in concept 2).

The idea of extracting features reflecting a pair of concepts from the features of the instances of that concepts is sketched in Figure 3.20.

In that way a feature vector – a point in the so created "similarity space" –

[9]Although not explicitly stated by the authors, it is assumed that the class information in the training set is provided by experts in the field (human supervisors) or it comes from previous mappings performed by another mapping technique.

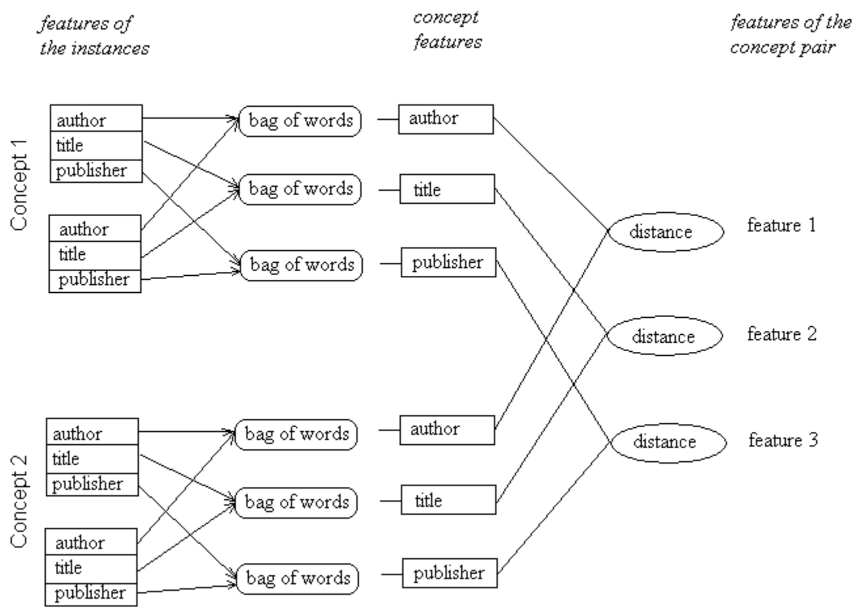


Figure 3.20: Feature extraction (adapted from [160]).

is associated to every pair of possible mappings. Since not every point in that space will correspond to a real mapping, the authors suggest the hypothesis that there can be found a correlation between the label of a point (*positive*, if the point represents a "true" mapping and *negative*, otherwise) and the points position in the space which gives rise to reformulating the mapping problem into a binary classification task.

The Caiman tool. Lacher and Groh [88] contributed to the ontology mapping research with their system CAIMAN, created to facilitate the retrieval and publishing of documents among communities.

CAIMAN adopts an extensional ontology modeling approach by defining concepts as sets of relevant documents, coded as TF/IDF vectors. In the mapping process, every concept in each of the source ontologies is represented as a feature vector by using the well known Rocchio algorithm – the concept's feature vector is calculated as the average of the documents that are assigned to this concept. The similarity of two concepts from two different ontologies is assessed by measuring the cosine distance between their corresponding feature vectors, denoted by $p(a_i, b_j)$ for two concept vectors a_i and b_j.

For a fixed i, the algorithm which CAIMAN exploits measures the quantity $p(a_i, b_j)$ for all j and assigns a mapping between a_i and b_{j_k} for the node b_{j_k} which produces the highest value of p. If two nodes b_{j_l} and b_{j_m} produce very similar p-values, the cosines of their parents is calculated until the difference between their corresponding p-values becomes significant (with respect to a fixed threshold). This final characteristic of the method puts it a step closer to structure based matching techniques, because it suggests using hierarchical information from the ontology graphs.

Matching of instances
Thor *et al.* [147] proposed a matcher operating on ontologies of strictly hierarchical type designed predominantly in a bottom-up manner for the purposes of e-commerce. The system is of highly narrow application domain, because it relies on the existence of an unique id-key of every concept instance (a plausible assumption for a restricted category of ontologies such as on-line catalogs). The ontology matching problem is then reduced to an instance catalog-id alignment problem.

Folksonomy concept mapping. *Folksonomies* are data structures which result from a collaborative bottom-up annotation process. Wartena and Brussee [162] proposed an instance-based method for mapping folksonomic concepts, based on the distribution of tags in each concept. The main argument which enables the application of a similarity measure based on these distributions is that, although the process of tagging is highly human biased and noisy (tags like "to read" or "interesting" do not bring any qualitative information for the tagged article), only the relevant tags will reach high frequencies. Halpin *et al.*, as cited by Wartena and Brussee have proven that the distribution of tags (for entities which are being frequently tagged) converges over time.

Concepts as document vectors. An ontology mapping procedure, based on vector distances and bearing characteristics of structural as well as instance-based techniques was suggested by Su [145]. The main assumption of the approach is that one has two input ontologies, each with a set of documents assigned to its relevant concepts. Every leaf-node concept is modeled as a vector which is computed as the average vector of the vectors of all documents assigned to it. The non-leaf concepts are represented as feature vectors which are computed from the feature vectors of their descendants. For every pair of concepts, a similarity measure based on the cosine vector distance is calculated.

A few more instance-based mapping methods are discussed in [43], while [46] presents a benchmark for instance-based ontology mapping in order to facilitate the evaluation of the different techniques proposed in the literature. Additional overview of ontology reconciliation techniques is found in [66] by Hameed *et al.*. The study aims at explaining, from an engineering and enterprise viewpoint, where the problem of ontology mapping evolves from and overviews some up-to-date tools for merging of ontologies. A strong point is the attempt to study the ground reasons for the emerging ontology mismatches by providing multi-perspective classifications of ontology dissimilarities. Finally, a comparison of commonly applied ontology distances was recently proposed by David *et al.* in [41].

3.5.3 Semantic-based Approaches

The group of semantic-based approaches unites methods, which rely on logical deduction in order to justify and verify a set of previously generated mappings. As this definition suggests, these methods usually consist of two main parts:

1. **Anchoring the source ontologies.** This is the process of initial alignment of two (or more) source ontologies by matching them against an existing external resource of some kind. This can be a formal top-level

ontology, such as DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [50] and SUMO (Suggested Upper Merged Ontology) [121], a formal domain specific ontology like, for instance, the FMA (Formal Model of Anatomy) ontology in the medical domain, or an informal resource, like WordNet[10]. The main characteristic of this method is that the input ontologies (the mapping candidates) are first aligned to (parts of) the background ontology. Checking if the concepts and relations of the source ontologies correspond to one another is performed by the help of reasoning services in the background ontology.

2. **Applying deductive techniques.** The second part of the mapping procedure consists of verifying the consistency and the completeness of the correspondences found in the first phase and entailment of new alignments. To these ends, on applies techniques from propositional or description logics for verifying semantic satisfiability of the correspondences and deduce new knowledge.

We will illustrate how semantic-based approaches look like in action by the help of a small example. Let our task be to match two music-ensembles related ontologies, denoted O_1 and O_2. We decide to use the (fictional) Music Ensembles Ontology (ME), which provides formal domain knowledge, as a background source to which our input ontologies will be anchored. In result, we come up with a set of assertions binding the elements of the two ontologies with relations, such as $\equiv$, $\sqsubseteq$, etc.

Let the concept `Orchestra_O_1` from O_1 be mapped to the concept `MusicEnsemble` from the FMA ontology and let the concept `Player_O_2` from O_2 be mapped to the concept `Musician` from the ME ontology. This immediately entails that the concept `Player_O_2` is a part of the concept `Orchestra_O_1`, because a part-of relation holds between the concepts `Musician` and `Orchestra` in the ME ontology[11]

Now let us have the following two parts of O_1 and O_2 (written in description logics):

$$\texttt{ChamberOrchestra} = \texttt{Strings} \sqcap \leq 15\ \texttt{musician},$$

$$\texttt{Orchestra} = \texttt{StreichAndWoods} \sqcap \leq 50\ \texttt{player}^{12}.$$

The former means that a chamber orchestra is a collection of not more than 15 string instrumentalists and the latter means that an orchestra is a collection of not more than 50 strings and woods instrumentalists.

Let us have the following initial alignments from the anchoring stage of the mapping:

$$\texttt{musician} = \texttt{player}$$

[10]http://wordnet.princeton.edu/

[11]As observed in [43], this mapping of concepts and their relations might be less straightforward when using WordNet where the word "Player", for instance, has many different senses among which one has to disambiguate.

[12]The word "Streich" is often used to denote the part of an orchestra containing only string instruments. The word "Woods" here stands to denote the wood instruments section of an orchestra. The concept "Streich and Woods" denotes the union of both sections.

$$\texttt{Strings} \sqsubseteq \texttt{StreichAndWoods}.$$

The latter two alignments entail that a chamber orchestra is a subcategory of orchestra:

$$\texttt{ChamberOrchestra} \sqsubseteq \texttt{Orchestra}.$$

Semantic-based approaches do not fall in the focus of the work carried out in this thesis; therefore, we will not delve into further details. We propose the interested reader to consult the work of Bouquet, Shvaiko, Serafini and others (cited in [43]), the IF-Map method by Kalfoglou and Schorlemmer [79], the review of semantic metrics by Hu *et al.* [71], as well as related publications by Kühnberger and Ovchinnikova [118, 120].

3.6 Summary: Ontology Matching

Throughout the sections above, we have discussed different issues related to the interdisciplinary domain of research known under the name of ontology matching. We have seen that there are many possible definitions of an ontology, just as there are plenty of application fields, ranging from knowledge integration, semantic interoperability, through natural language processing, problem solving and reasoning, to facilitating business interaction and increasing the efficiency of production. In view of the variety of definitions and application fields, an ontology, in the broadest sense, is understood as a collection of concepts and relations defined on these concepts, which altogether describe the knowledge in a certain domain of life and provide basis for reasoning and inferring new facts.

Ontology matching is defined as the process of finding correspondences between the elements of two or more ontologies, which are assumed to cover the same or similar domains of knowledge, but their components are not explicitly mapped to one another. We have attempted to study the reasons why such conceptual and terminological discrepancies appear from a psychological and broad computational points of view and have provided an account of the role of similarity in human concept formation.

From the perspectives of applications and computational models, we have presented a classification of the existing theoretical and practical contributions (matching systems, procedures / algorithms and tools), organized in classes with respect to their conceptual and methodological characteristics. We have particularly emphasized on two major groups of techniques, which are directly relevant to the approach developed in this thesis: structural and instance-based techniques.

The basic conclusion concerning the performance and efficiency of structure-based techniques is that structure, although an important bearer of the semantics of a group of concepts, does not provide sufficient ground to build a self-dependent matching procedure. Structural techniques have to be, therefore, used in combination with terminological and / or extensional approaches.

Instance-based techniques, on the other hand, prove to be a more reliable indicator of semantic proximity of concepts and ontologies. In consequence, more and more recent research efforts are invested in that direction. They can be used in combination with other techniques, but also as an independent indicator of semantic similarity, as it has been observed by many authors (see

Section 3.5). The biggest challenge for instance-based approaches remains to overcome one of their most frequent and least realistic assumptions that the same set of instances exists for both (or the many) source ontologies to be matched.

In the next chapter of the thesis, we will expand on the theoretical grounds of our own contribution to the field. We will combine structural and extensional methods, developing graph-theoretical and machine learning approaches to address the problem of ontology matching, attempting to answer frequent critiques to some of the existing approaches, reviewed in the chapter that we have just closed.

Chapter 4

Composite Approaches to Ontology Matching

Among the main contributions of this thesis are two novel procedures for ontology matching that combine structural and extensional techniques yielding assertions on the overall semantic similarity of two source ontologies. The first procedure, introduced in Section 4.4, relies mainly on instance-based matching techniques; the structure of the source ontologies comes in use in the definition of a recursive algorithm for instance-based concept similarity verification, which optimizes the search of similar concepts among the source concept trees. The second matching procedure, discussed in Section 4.5, has a stronger structural component and explores different possible cases with respect to the structure and the extension of two (hierarchical) source ontologies. The outcome of the first procedure is a set of concept alignments, whereas the second procedure gives indications how both ontologies (or their overlapping parts) can be merged together, if from an engineering and application viewpoint this is necessary.

In the next couple of sections, we will describe separately the components of the two composed approaches before we explain how they will combine for the goals of an overall ontology matching task. We start by describing the environment, in which the task will be accomplished by stating our main assumptions (Section 4.1). We proceed to discuss the structural properties of ontologies and to define their similarity in terms of a graph-isomorphism problem (Section 4.2). The central part of this chapter and of the theoretical and practical contribution of the thesis is the study of instance-based measures of ontology similarities and how they can be combined in an independent extensional ontology matching procedure (Section 4.3). After presenting the two matching approaches (Sections 4.4 and 4.5), we close the chapter by a short summary of our main results (Section 4.6).

4.1 Assumptions

Clearly, given the complex structure that an ontology may possess with respect to the various application fields and relation definitions that it may contain, it might become prohibitive or simply lack sense to discuss the problem of ontology similarity from a very general point of view. Or, as Welty and Guarino [62] observed, talking about ontologies only makes sense regarding a certain context. Therefore, we will start by specifying the frames in which the study of ontology similarity will be carried out in terms of the following basic set of assumptions.

The focus falls on ontologies which:

- are hierarchical, i.e. make use of but are not restricted to declarative hypernym / hyponym (`is_a`) relations (in a broader sense, that means that there exists a hierarchical structural body of each ontology);
- are designed to categorize text documents[1] with respect to different topics;
- are populated – there exists a set of documents correctly assigned to their nodes;
- share an extensional overlap.

In the sequel, we will discuss in more details the assumptions on the list above.

4.1.1 Definition of a Hierarchical Ontology

In Chapter 3 (Section 3.1), we saw that there is little agreement among authors on a common definition of an ontology. If not partially underlying it, this is at least symbolic for the problem of ontology mapping: few people would have a common clearcut convention of what an ontology is – clearly, a fact which does not help to provide a common ground for ontology acquisition and construction.

We will make our own contribution to the list of ontology definitions, some of which we have discussed in the previous chapter of this thesis. Explicitly, in our study, we deal with hierarchical, tree-structured ontologies designed to categorize text documents such as web pages with respect to their contents. Generalizations to non-hierarchical ontologies are done under the assumption that a hierarchical body can be extracted out of each of the considered source ontologies.

Definition 39 *A hierarchical ontology is a pair* $O := (C_O, \texttt{is\_a})$*, where* C_O *is a finite set whose elements are called concepts and* `is_a` *is a partial order on* C_O *with the following property:*

- *there exists exactly one element* $A_0 \in C_O$ *such that* $\{B \in C_O, B \neq A_0 | (A_0, B) \in \texttt{is\_a}\} = \emptyset$*,*
- *for every element* $A \in C_O$*,* $A \neq A_0$*, there exists a unique element* $A' \in C_O$ *such that* $(A, A') \in \texttt{is\_a}$*.*

[1]As noted already, the procedures described below generalize to instances of other nature (e.g. images), under the assumption that they can be represented as real-valued vectors of some kind. The term "document" is used throughout the text to substitute the more abstract term "instance"; however, "document" can be also seen as used abstractly to denote a real word data entity, which need not be limited to a text written in a given natural language.

As commented previously (Section 3.1), the case $C_O = \emptyset$ is trivial and considered as excluded, although the given definition does not explicitly suggest so.

Finally, note that definition 39 is purely intensional and does not imply the existence of instances of the ontology concepts. Therefore, throughout the succeeding sections, we will throw light upon the question of how do we assume concepts, instances and documents are related.

4.1.2 Documents, Instances, Concepts, Ontologies

There is just as little agreement about the relation between documents, instances, concepts and ontologies, as there is about a common definition of an ontology and the former seems to be a direct consequence of the latter. The definition of a hierarchical ontology we have just given clarifies the relation between the concepts within a single ontology. We will continue by explaining the assumed relation between documents, instances and concepts in our ontology matching scenario.

In a bottom-up setting, a set of documents can serve as a basis to extract ontological information. Methods which provide solutions in that direction have been discussed in the previous chapter (see for example [35, 124, 144]). On the other hand, one may adopt a top-down approach by first creating an ontology and only afterwards populating it with instances. These could be words or expressions contained in properly classified documents, or the documents themselves [25]. This is the strategy applied in many ontology engineering solutions aiming to "fit" the knowledge of a domain into a formally and formerly created structure. A practical reason for that is the fact that it is computationally less costly to populate a created ontology with instances than to extract structure out of a flat pool of instances, what the alternative bottom-up approaches suggest.

In either way, we assume that the ontology population phase had already taken place. More precisely, we assume the existence of an extension of each ontology which is a collection of annotated text documents (web-pages) assigned to that ontology and distributed among its nodes. We will use the documents, assigned to a given concept, as instances of that concept in order to model it.

After clearing up the question of what we consider to be a legitimate concept instance there still remains an important question unanswered, namely *how is the set of instances of a given concept defined?* Assuming that we have a set of annotated instances for each ontology (i.e. for each instance there is a pointer to which ontology concept it refers to), there are two possibilities. The first one is to ignore the hierarchical structure of the underlying taxonomy and take as instances of a given concept only those that are directly assigned to it and let us call that a *non-hierarchical instantiation.* The second possibility is to include in the set of instances of a concept all instances assigned to that concept and all of its descendants in the concept hierarchy – a *hierarchical instantiation* [72]. An illustration is provided in Figure 4.1: in a hierarchical instantiation setting the node $c2$ would contain the documents set $\{d1, ..., d6\}$ and $c1$ would contain the complete document set $\{d1, ..., d8\}$. In a non-hierarchical setting, the document $c2$ would only contain the document d_1, etc.

Usually, the choice of one of the two types of instantiations is motivated by semantic considerations dependent on the particular intended application. In

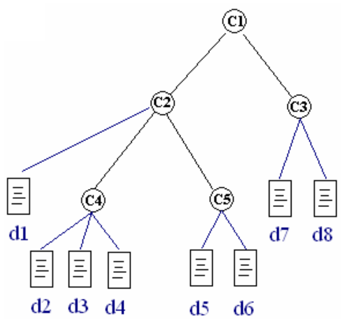


Figure 4.1: Example: a document populated taxonomy.

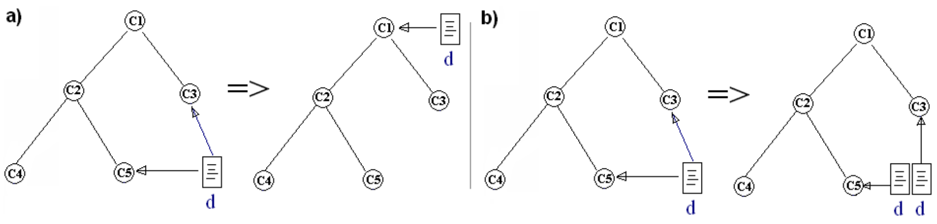


Figure 4.2: Doubly annotated documents.

some web directories, like for instance *Yahoo!*, the assignment is understood as direct assignment of documents to a node and not to its super-nodes in which, naturally, not all taxonomy classes are assigned documents. Concerning our approach, the instance-based concept similarity measures, presented in Section 4.3 perform independently on the assumed type of concept instantiation. Accordingly, Isaac *et al.* have reported that in an instance-based matching task, it is expected the results in both hierarchical and non-hierarchical instantiation to be very competitive [72]. Nevertheless, for the purposes of our overall ontology matching approaches, we adopt a hierarchical instantiation of concepts, for reasons to become clear during the presentation of the proposed procedures.

Finally, a document may be assigned through its annotation to more than one ontology nodes and to more than one ontologies. Cross-ontology doubly (or multiply) annotated instances are considered separately and independently for each ontology. A doubly (or multiply) annotated document within a single ontology can be considered to be an instance of the least common subsumer of the concepts, which it is originally assigned to (as shown in Figure 4.2a), or a copy of it can be kept in each of the nodes to which it is originally assigned (Figure 4.2b). If the quantity of multiply annotated documents in an ontology is significant, taking the former approach can be problematic, risking to lift a lot of instances up in the hierarchy, thus weakening the structure. Therefore, we suggest to keep documents as instances of their respective nodes, as indicated by their annotation.

We will place in a formal frame the preceding discussion about the association of documents to an ontology and their distribution among its nodes.

Let $\Omega = \{O_1, ..., O_i, ..., O_k\}$ be a finite set of ontologies and $\Delta = \{\mathbf{d_1}, ..., \mathbf{d_j}, ...\mathbf{d_l}\}$ – a finite set of text documents about a given domain of interest. Each text document is represented as an n-dimensional TF/IDF vector (as described in Chapter 2 (Section 2.4) and in [74]). Let

$$\gamma : \Omega \to 2^{\Delta} \tag{4.1}$$

be an injection from the set of ontologies to the power set of the documents. For every $O \in \Omega$ and $D_O \in 2^{\Delta}$, so that $\gamma(O) = D_O$ there exists an injection

$$g : C_O \to 2^{D_O}, \tag{4.2}$$

which assigns documents to the ontology nodes. For the purposes of our study, we assume that the function g maps every element of C_O to a set of documents and assigns to a class the union of the sets of documents assigned to all nodes subsumed by this class.

4.1.3 Types of Document Intersection

An entity that plays a key role in the approach described in Section 4.5 is the intersection of the sets of documents D_1 and D_2 assigned respectively to two source ontologies O_1 and O_2. The relative size of the intersection indicates how similar the extensions of both ontologies are and to what extent the domains that they cover overlap. In our procedure this quantity plays a twofold role. In the first place, it comes to confirm or reject one of our basic assumptions, namely that the ontologies of interest share a certain extensional overlap, i.e. that they are good mapping candidates. Further on, at the final mapping stage, combined with the structural and instance-based mapping techniques, it helps yield quantitative judgments on the granularity and population differences between the inputs and how these differences can be overcome.

However, working with the mere intersections is too naive when the sets of elements are vectors of natural language text documents. It is very likely that documents from both sets that are very similar, yet not identical, will remain out of the intersection. For example, an article and its abstract might have different TF/IDF vectors although they are semantically very similar. Therefore, we need a theoretical approach to deal with that by taking into account not only identical, but also to a large extent similar documents. We will distinguish between three different types of intersection: the *strict intersection*, or the standard set intersection, the *relaxed intersection*, containing the strict intersection and documents from both sets which are close in a vector space, and the *modified relaxed intersection*, containing the first two plus documents which are close in a semantic space.

We make the convention that the use of the term "intersection" in the second and the third type of intersection described above is a slight abuse to the common notation, since, as it will be seen further, it carries the characteristics of both an intersection *and* a union. A possible fuzzy set formulation of the problem, which helps to avoid this terminological conflict, is presented at the end of the section.

A Distance Metric on a Set of Documents

As discussed previously (cf. Chapter 2), the distance between two documents can be measured in a vector space constructed from the terms occurring in the documents or their combinations used as dimensions. There are different plausible choices of a distance metric or a similarity measure defined on a set of documents coded as TF/IDF vectors (or any other numerical format) living in a multi-dimensional space of some kind. The particular choice is task and data dependent and in most of the cases, once the document vectors are computed, it is easy to try out different distance measures and pick out the most appropriate one. In Section 2.4, we have introduced the well-known Euclidean distance and cosine measure of similarity. A review of some more commonly used distances is found in [87] and the sources cited there. For the purposes of our study at this point we need not commit to one particular distance metric. Instead, we will introduce an abstract distance between two documents, $dist(d_1, d_2)$, with the convention that *dist* is a metric and defines a metric space on the set of documents.

Let D_1 and D_2 be two document sets corresponding to the ontologies O_1 and O_2, respectively. We will consider two documents $\mathbf{d_i^1} \in D_1$ and $\mathbf{d_j^2} \in D_2$ taken from these sets *similar* if their distance is smaller than a fixed threshold, i.e.

$$dist(\mathbf{d_i^1}, \mathbf{d_j^2}) \leq c_d,$$

where c_d is a strictly positive parameter (set by the user). We should keep in mind that taken into account the fact that the ontologies we are dealing with are domain specific (i.e. the documents they organize are already rather similar), small variations of the constant c_d would lead to important differences in the similarity judgments.

Strict and Relaxed Documents Intersections

We will introduce the notion of relaxed intersection which integrates similar documents from both sets as opposed to the standard strict set intersection.

Definition 40 *A Strict Intersection (SI) of the sets of document D_1 and D_2 is the set*

$$SI(D_1, D_2) = D_1 \cap D_2$$

and their Relaxed Intersection (RI) is the set

$$RI(D_1, D_2) = \{\mathbf{d_i^1}, \mathbf{d_j^2} | dist(\mathbf{d_i^1}, \mathbf{d_j^2}) \leq c_d, \mathbf{d_i^1} \in D_1, \mathbf{d_j^2} \in D_2\}.$$

We note that $SI(D_1, D_2) \subseteq RI(D_1, D_2)$. The difference between both is shown in Figure 4.3.

An Account for the Semantic Similarity of Documents

The relaxed intersection helps to include into consideration documents from both input sets which are not identical but rather similar. Still, it does not account for documents which are of substantially different word content, because the similarity is measured in terms of a distance in a term-vector space (i.e. word-co-occurrences are what count), and not in terms of the semantic closeness of the documents. Two documents that use semantically related, but distinct words will show no similarity by measuring their vector distances and will neither fall into the strict, nor into the relaxed intersections introduced above.

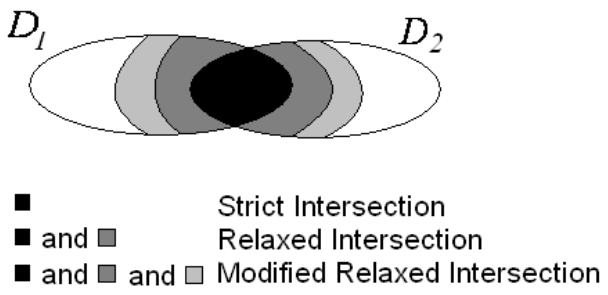

D_1
D_2
■ Strict Intersection
■ and ▩ Relaxed Intersection
■ and ▩ and ▨ Modified Relaxed Intersection

Figure 4.3: Types of document sets intersections.

Kandola, Shawe-Taylor and Cristianini [81] proposed a method for representing and computing semantic proximity of documents by the help of a kernel based similarity measure using a semantic proximity matrix. The method is reviewed in Section 2.4 of the second chapter of the thesis (see equation (2.42)). Another method by the help of which one extracts information about the semantic closeness of terms and documents is based on Latent Semantic Analysis (also discussed in Section 2.4).

We suggest that in measuring the documents set intersection it is important to include documents which share semantic commonalities, not accounted for by their term occurrences vectors. Let $\sigma_{sem}(\mathbf{d_i}, \mathbf{d_j})$ be a measure of semantic similarity[2] of two documents $\mathbf{d_i}$ and $\mathbf{d_j}$ based on either semantic proximity matrices (equation (2.42)) or LSA (in this case this is again most commonly the cosine distance, but applied in the semantic spaces generated by LSA, and not in the original vector space). Our proposal is to apply σ_{sem} on the set of documents which remain from the initial sets D_1 and D_2 after removing their relaxed intersection (RI), i.e. we apply the $\sigma_{sem}(\mathbf{d}, \mathbf{d'})$ for all $\mathbf{d} \in D_1 \setminus RI(D_1, D_2)$ and for all $\mathbf{d'} \in D_2 \setminus RI(D_1, D_2)$.

Let the documents from D_1 and D_2 which are found to be semantically close via σ_{sem} form the set

$$SC(D_1, D_2) = \{\mathbf{d_i^1} \in D_1, \mathbf{d_j^2} \in D_2 | \sigma_{sem}(\mathbf{d_i^1}, \mathbf{d_j^2}) \geq c_{sem}\}, \tag{4.3}$$

where c_{sem} is a parameter to be fixed.

We define the modified relaxed intersection of documents which contains the relaxed intersection from definition 40 and the semantic intersection defined in (4.3) as (see Figure 4.3):

$$RI_{mod} = RI \cup SC. \tag{4.4}$$

In our procedure, we will use the modified relaxed document intersection instead of the strict one and by documents set intersection we will mean the modified relaxed intersection, except if specified otherwise.

As a final remark, we note that the modified relaxed intersection provides the most general notion of semantic proximity of two sets of documents and therefore it is preferable to work with. However, dependent on the corpus, it is

[2]For the time being assume this is an abstract similarity measure, its explicit definition being application relevant.

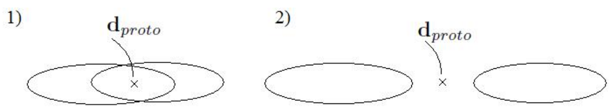


Figure 4.4: Prototype document in case 1) and case 2).

theoretically possible to achieve satisfactory results by using only the strict or the relaxed intersection.

A Fuzzy Set Formulation

Exploring the potential of expressing the relaxed intersection of documents by the help of a fuzzy set formalism is a worthwhile effort. Since the fuzzy set theoretic approach remains out of the scope of the thesis, we will only sketch a possible approach to this problem.

Fuzzy set theory has been introduced by L. A. Zadeh in the sixties of the past century as a generalization of the standard theory of sets [169]. A fuzzy set is defined on a given space of objects X as a couple (A, f_A), where A is characterized by f_A – a function that expresses the degree of membership of every element of X to A by assigning to every element $x \in X$ a value from the interval $[0, 1]$.

In our scenario, we start by considering two cases: 1) the strict intersection $D_1 \cap D_2$ is non-empty and 2) the strict intersection $D_1 \cap D_2$ is the empty set. In both cases, we aim at defining a special document vector, $\mathbf{d}_{proto}$, which is a prototype of the shared commonality of both sets. In case 1), we formulate $\mathbf{d}_{proto}$ as the average vector of all vectors contained in $D_1 \cap D_2$; in case 2), we compute the pairwise distances between each vector in D_1 and each vector in D_2 and define $\mathbf{d}_{proto}$ as the average of the two closest vectors from both sets (see Fig. 4.4).

We proceed to translate the notion of a relaxed document intersection in a fuzzy set formulation. We take the union of both document sets $D_1 \cup D_2$ as the space X. The relaxed intersection of D_1 and D_2 will be defined as the couple (A, f_A) where the membership function f_A relates to the distance of each element of $D_1 \cup D_2$ to the prototype vector and is defined as $f_A(\mathbf{d}) = 1 - dist^*(\mathbf{d}, \mathbf{d}_{proto})$, where $dist^*$ is the distance function $dist$ scaled in the interval $[0, 1]$. The modified relaxed intersection is defined in a similar manner, by first taking the membership function $f_A^{sem}(\mathbf{d}) = \sigma^*_{sem}(\mathbf{d}, \mathbf{d}_{proto})$, where σ^*_{sem} is the scaled version of σ_{sem} in the interval $[0, 1]$ and then computing the fuzzy union of f_A and f_A^{sem}. We recall that, in fuzzy set theory, the latter will be expressed by $Max[f_A(\mathbf{d}), f_A^{sem}(\mathbf{d})]$, $\forall \mathbf{d} \in X$.

In practical terms, fixing a threshold of degree of membership enables us to define a notion of set intersection based on the fuzzy formulation above, which gives crisp criteria of which documents are to be taken into account. Figure 4.5 gives the analytical interpretation of the fuzzy union and the fixed threshold resulting in the definition of a modified relaxed intersection in the case when X is the real axis. We note that the fuzzy approach depends on only one parameter, namely the threshold of degree of membership, whereas the approaches that rely on crisp set definitions (described in the beginning of this section) imply the choice of two parameters.

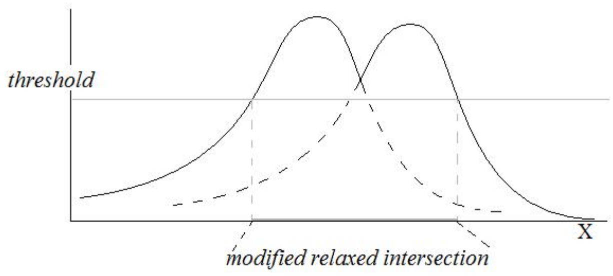


Figure 4.5: The bell-shaped curves represent the functions f_A and f_A^{sem}. The fuzzy union $Max[f_A(\mathbf{d}), f_A^{sem}(\mathbf{d})]$ is depicted by the continuous segments of the two curves.

4.2 Structural Ontology Similarity

We proceed to explore the advantages of a properly chosen graph representation of ontologies for the efficiency of the structural ontology similarity determination. We start by presenting hierarchical ontologies by trees; later on, we discuss the possibility of presenting more general ontologies (containing other than subsumptional relations) as Cartesian products of trees. A distance metric on a set of graphs is defined to account for the structural ontology similarity.

The basic definitions from graph and lattice theory which we will need have been given in Section 2.1.

4.2.1 Ontologies as Graphs and a Structural Similarity Measure

Representing ontologies as graphs is a trivial task which consists of aligning an ontology's concepts and relations with a graph's nodes and edges, respectively. Formally, this will be done in the following manner.

We start by representing hierarchical ontologies as trees. Let Γ be a set of directed rooted trees $G(V, E)$ and Ω be a set of hierarchical ontologies, as defined above (Definition 39). We assume that for every $O \in \Omega$ there exists $G \in \Gamma$ so that $|C| = |V|$, where $|\cdot|$ denotes set cardinality and there exists an isomorphism

$$f : C \to V \tag{4.5}$$

i.e.

$$(c_i, c_j) \in \texttt{is\_a} \Leftrightarrow \langle f(c_i), f(c_j)\rangle \in E, \tag{4.6}$$

where $c_i, c_j \in C$.

We give the following definition.

Definition 41 *Let O be a hierarchical ontology. A* ***hierarchical ontology tree*** *corresponding to O is a directed rooted tree $G(V, E)$ such that*

(1) $V = C$

(2) $E \subseteq C \times C$ such that $\langle f(c_i), f(c_j)\rangle \in E \Leftrightarrow (c_i, c_j) \in$ `is_a`.

We note that strictly speaking, the edge relation is not necessarily transitive in contrast to the partial order `is_a` and therefore, for the purposes of the definition above we should relax the transitiveness of `is_a`.

Analogously, general ontologies are represented as general graphs by defining an isomorphism in a similar manner as in the definition above. We recall the ontology primitives defined by Maedche [97] by introducing a slight modification[3]:

- $\mathcal{L}_C$ – a set of labels for concepts;
- $\mathcal{L}_R$ – a set of labels for relations;
- $\mathcal{C}$ – a set of concepts;
- `is_a` – a partial order on $\mathcal{C}$;
- $\mathcal{R}$ – a set of binary non-taxonomic relations;
- $\mathcal{F}$, $\mathcal{G}$ – mappings relating concepts and relations with their labels.

Let the list above be denoted *ontology primitives (4.2.1).*

Definition 42 *Let O be an ontology which is defined by the ontology primitives (4.2.1) . A graph $G(V, E)$ corresponding to O is constructed in the following manner.*

(1) $V = C$ forms the set of vertices via the isomorphism f;

(2) $E \subseteq C \times C$ such that $\langle f(c_i), f(c_j)\rangle \in E \Leftrightarrow \{(c_i, c_j) \in$ `is_a` or $(c_i, c_j) \in \mathcal{R}\}$;

(3) $L_V = \mathcal{L}_C$;

(4) $L_E = \mathcal{L}_R$;

(5) the mappings h_V and h_E are identical with $\mathcal{F}$ and $\mathcal{G}$, respectively.

L_V, L_E, h_V and h_E are, respectively, sets of vertices and edges labels and functions mapping them to graph vertices and edges, as introduced in definition **??** (Section 2.1).

The distance function which will be used as an indicator of the structural similarities of two ontologies is Bunke's graph distance, introduced in Chapter 2. We recall its definition.

Definition 43 *Let $|G|$ denote the number of vertices in a graph G. The distance between two non-empty graphs G_1 and G_2 is defined as*

$$d(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{max(|G_1|, |G_2|)}. \tag{4.7}$$

The same metric is applicable if instead of nodes we take arcs. To define it, consider in (4.7) the denotation $|G|_{arc}$ instead of $|G|$ which stands for the number of arcs in a graph G. The rest remains unchanged.

Working with Bunke's distance function has several advantages with respect to our ontology matching strategy. To start with, (4.7) *is* a metric (proved

[3]The original set of primitives includes one set of labels for concepts and relations. We have split this set in two in order to avoid the possibility to assign a relation the label of a concept or the other way round.

by Bunke *et al.* in [20]) – a useful property enabling us to define classes of equivalences among ontology structures. The distance is based on maximal common subgraphs. In the case of trees, there exist tractable algorithms of *mcs* identification (see, e.g. Section 2.1). Finally, (4.7) computes the ratio between the *mcs* and the cardinality of the bigger of the two trees – a useful step towards judging conceptual differences in the granularity of the source ontologies, which is an important outcome of the ontology matching procedure, to be discussed.

4.2.2 Discussion: Generalizing in Terms of Cartesian Product of Trees

We will discuss the question whether ontologies of a more general kind than hierarchies can be modeled by representing them as a product of hierarchies and letting them inherit the subsumptional properties of the hierarchical structures that compose them.

Similarly to Section 2.1, we will use the following notations. G, H denote directed rooted trees and $V(G)$, $E(G)$ – sets of nodes and arcs of a tree G, respectively; we will use simply the notation V and E where no disambiguation between the sets of nodes and edges of different graphs is needed. Finally, x, x', x'',... denote elements of $V(G)$ and y, y', y'',... – elements of $V(H)$.

We aim to provide a formal framework which allows for the construction of more general structures out of a set of trees. We consider hierarchical, tree-like ontologies, where the adjacency relation in a tree is interchangeable terminologically with the subsumptional (`is_a`) relation in a hierarchy, as indicated by definition 41. We define an operation on the set of trees which will lead us to more general structures capable of modeling more general (than hierarchical) ontologies [45]. We suggest that the Cartesian product is an appropriate choice to these ends.

Definition 44 Cartesian product on graphs. *The Cartesian product of two graphs G and H is a graph $G \circ H$ such that:*

1) the vertex set of $G \circ H$ is the Cartesian product of the vertex sets of G and H, i.e. $V(G) \times V(H)$;
2) two vertices (x, y) and (x', y') of $G \circ H$ are adjacent if and only if either $(x \texttt{ adj } x'$ and $y = y')$ or $(x = x'$ and $y \texttt{ adj } y')$.

For trees, a Cartesian product is defined analogously. It is straightforward to verify that the product is not a tree; an example of a product of two simple trees is given in Figure 4.6.

More precisely, the resulting product tree defines a directed acyclic graph (DAG) – graph structure which shares commonalities with a tree, but allows for multiple inheritance[4]. A DAG provides a good model for ontologies which have `part_of` relations in addition to the `is_a` relations. For instance, declaring `part_of`(a, b) and `part_of`(a, c) indicates that a is part of both b and c (for example, a wooden board is a part of a table and of a skateboard); declaring `part_of`(a, b) and `is_a`(a, c) indicates that a is a part of b and is a type of c (for example, a wooden board is a part of a table and is a construction material).

[4]As introduced in Section 2.1, a DAG is formally defined as directed graph with no directed cycles. A partial order is trivially defined on the set of a DAG's vertices.

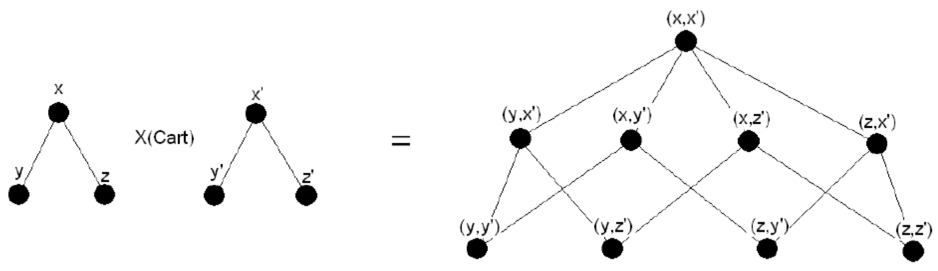


Figure 4.6: Example: Cartesian product of two simple trees.

Finally, a hierarchy with multiple inheritance can be formed by the `is_a` relation only (like for example in some programming languages, e.g. C++) by declaring `is_a`(a, b) and `is_a`(a, c) (e.g. a wooden board is a construction material, but also a piece of school furniture). All these declarations can be simultaneously represented and modeled by a DAG, but not by a tree.

Finally, we note that the resulting DAG graph can be viewed as a semi-lattice, keeping in mind the remark made above, concerning the transitivity of the partial order as opposed to the non-transitiveness of the adjacency relation.

Levels

We explore and generalize the notion of *levels*, which is well defined on trees, in the case of semi-lattices and DAGs resulting from the Cartesian product of two trees by focusing on the relation between the levels of the input trees and the levels of the resulting product tree.

Let $G(V, E)$ be a directed rooted tree. Let n be a function that maps a node to its corresponding level, $n : V \to \mathbb{N}_0$, where $\mathbb{N}_0$ is the set of natural numbers and 0. The map n determines the level of a vertex x from V, surjectively. By definition, the root node is assigned a level 0, all direct descendants of the root are assigned level 1, all their direct descendants – level 2, etc.

Now let $n(x)$ and $n(y)$ be the corresponding levels of vertices $x \in V(G)$ and $y \in V(H)$. As we have just seen, the Cartesian product $G \circ H$ of two trees G and H is not a tree. Nevertheless, we introduce the notion of a level in the product, as well, in the following manner.

Definition 45 *The level of the vertex $(x, y) \in V(G \circ H)$ is defined as the sum of the levels of x and y.*

$$n(x, y) = n(x) + n(y). \tag{4.8}$$

One can easily verify the definition in the example given in Figure 4.6: the sum of the levels of the roots x and x' is 0 which is the level of the product tree root (x, x'), and so on. Moreover, levels defined in that way can be acquired for the product of two product trees (like the one on the right side of Figure 4.6). These observations lead us to conclude that the notion of levels is very flexible and we relate this to the variety of relations which can be present in an ontology.

To conclude, we note that a relation of subsumption, modeled by a tree (or, more precisely, a tree-like semi-lattice), is core for the ontology, but in an ontology in general we might as well have other relations. However, our claim is that some non-subsumptional relations, like `part_of` can be modeled by partially ordered structures, such as semi-lattices and DAGs as well, and levels can be defined in a non-subsumptionally related structure of concepts, such as an ontology in general, as long as the ontology can be represented as the product of two trees or two product trees.

In the following section, we will move the focus from studying the structural properties of ontologies towards looking into extension-based models for concepts and their similarity.

4.3 Instance-based Ontology Matching

As this has been discussed previously, the problem of ontology heterogeneity, which gives rise to the need for matching, stems to a great extent from the fact that the nature of the process of ontology acquisition is decentralized and strongly human-biased. Ontology matching aims at overcoming the ontology heterogeneity by identifying similarities between the ontology elements (concepts, relations, instances). In the introduction, we have pointed out that this task can be approached and accomplished by the help of various theoretical mechanisms. In the previous section, we have described how ontology similarity can be furthered by purely structural methods using graph representation of ontologies and solving an isomorphism identification problem. However, it has been observed already by many authors that structural mapping techniques, although important, are not likely to provide an independent solution of an ontology matching task and therefore are commonly not considered and applied self-dependently, but rather as a support and in combination with approaches of other nature [43].

In the current section, we will expand on another general type of ontology matching methods, known as *instance based* or *extensional* ontology matching methods. This comprises a set of approaches for measuring the semantic similarity of two ontologies based on their extensions – the instances that populate their concepts. Commonly for such techniques, a set theoretic approach to modeling concepts is adopted in which a concept is defined as the set of its instances and the relatedness of a pair of concepts is estimated on the basis of the intersection of their instance sets.

In Section 4.1 of this chapter, we have put forward the question how the set of instances belonging to a concept is defined. We recall that with respect to whether or not inheritance via subsumtion among concepts is taken into account in defining concepts instance sets, one distinguishes between hierarchical and non-hierarchical instantiation. The former presupposes that concepts inherit the instances of their predecessors in the hierarchy, the latter does not. In either case, a clear definition of the set of instances of a concept is to be provided before an instance-based concept similarity measure of some kind can be introduced.

In the current section, we will focus on several aspects of instance-based concept similarity identification and measurement. The contribution can be considered in four separate but related aspect.

First, we present a novel technique for detecting potential mappings between concepts, based on Principal Component Analysis (PCA) and Discriminant Analysis (DA) (Section 4.3.2). It relies on discovering similarities according to the structure of both input ontology instance sets. The procedure can be used self-dependently, or in combination with another mapping technique. In the latter case, it serves as a procedure for narrowing down the number of concepts considered as candidates for a similarity check (i.e. potential concept matches), the actual similarity measurement being left to be performed by a concept similarity measure of other kind.

Further, we propose an independent measure of instance-based concept similarity which uses variable selection techniques for class discrimination (Section 4.3.3). The instances in our study are natural text documents assigned to the nodes of each ontology and coded as TF/IDF vectors [74] or other vector representation (e.g. term frequencies or raw occurrence counts) depending on the selection technique to be used. We suggest to apply variable selection mechanisms in order to score variables (terms in the TF/IDF vectors), with respect to their importance for a given concept, or the role they play for separating its instances from the rest of the instances of the same ontology. The proposed measure of similarity is based on comparing characteristic variables for two concepts taken from two different ontologies. In addition to the suggested measure, we explore the possibility of applying standard statistical correlation measures, calculated on the variables scores, which have the advantage of being parameter-free. Finally, note that the choice of a variable selection procedure within this setting is left to the user. We discuss, however, various standard variable selection techniques which are good candidates to serve as an input for the proposed similarity measures.

The third contribution contained in the section is a novel variable selection criterion based on Support Vector Machines, which, we will argue, outperforms standard selection techniques (Section 4.3.4). An evaluation of the performance of the similarity measure using standard techniques compared to its performance when using the SVM-based criterion is proposed to the reader in the last but one Chapter (5) of this thesis.

In a separate section (Section 4.4), we propose an overall extensional ontology matching procedure, which is directly based on the suggested concept-to-concept similarity criteria, optimized by taking into account the structure of the ontologies. The procedure builds on the observation that merely mapping concepts from one ontology to concepts from another is both computationally expensive and conceptually wrong, for in this case structure is completely overlooked.

Comparison to related approaches

Section 3.5 of Chapter 3 scrutinizes related work in the field. We underline again the importance of the study of Isaac *et al.* [72] comparing empirically the performance of different instance-based methods, as well as several state-of-the-art mapping systems, such as GLUE by Doan *et al.* [39], FCA-MERGE by Stumme and Mädche [144], the CAIMAN tool by Lacher and Groh [88] or the "mapping as classification" approach by Wang *et al.* [160]. Some of the most commonly applied similarity measures used by the cited approaches (including the Jaccard coefficient [39] and standard statistical measures [168]) have been described and compared in Chapter 3. Of course, instance-based mapping has also found an important place in the book on ontology matching by Euzenat and Shvaiko [43].

The instance-based approaches that we are about to present have several advantages, compared to other related techniques.

In contrast to most of the existing instance-based mapping procedures, the presented approaches do not rely on instance sets intersections and can be applied directly on ontologies populated with entirely different document sets. Using the descriptive statistical methods, or combining instance-based methods with structural information, the evaluation of the concepts pairwise similarity is done at once by the help of an easy to interpret geometrical representation. This prevents us from having to evaluate m times n concept pairs for two ontologies – one with n and another with m concepts.

Additionally, allowing the sets of ontology instances to be different for the two source ontologies makes the expensive step of extracting instances for the source ontologies from text (as done in [144]) unnecessary. By using the variable selection based measure of similarity, the relevant variables are determined for each ontology independently, and the matching itself is an inexpensive computation.

The concept similarity values are computed on the sets of input variables which, in case of text, correspond to actual words. Thus, the most important words that discriminate between a pair of similar concepts and the rest of the pairs of concepts can be readily made available in contrast to related methods (e.g. CAIMAN or GLUE). This information is useful to evaluate the quality and coherence of the matching results.

We suggest applying machine learning techniques (precisely SVMs) to select the characteristic variables. The fact that the assessment of the similarity of two concepts is entirely accomplished at the training-phase[5] of the learning task is a serious advantage of the method compared to state-of-the-art approaches relying on machine learning techniques (like Doan's GLUE [39] or Lambrix' literature-based alignment [89]).

Finally, the method is stable in multi-linguistic environments since documents from both ontologies need not be in the same natural language. It suffices that the documents TF/IDF vectors are translated into a single target language and not even all their features, but only the selected ones.

The suggested advantages significantly reduce the computational complexity of the method and increase its time efficiency.

4.3.1 An Instance-based Mapping Scenario

We recall the reader that the mapping problem in our overall setting consists in identifying semantic similarities between two heterogeneous hierarchical input ontologies, each equipped with a set of instances. In an instance-based mapping perspective, we focus on aligning pairs of cross-ontology concepts by degree of semantic similarity, measured on the basis of their extensions by the help of *machine learning* techniques.

[5]Recall that an automatic classification task is typically accomplished in two main steps: *training (or test) phase*, when available data is "learned" by the machine algorithm and *classification phase* when the learned rule is applied on unseen instances.

Machine learning classification algorithms require a specific kind of input. Usually training examples are represented as real valued or categorical vectors and their corresponding classes are coded via integer numbers.

In our instance-based ontology matching scenario, we consider two source ontologies $O_1 = \{C_1, \texttt{is\_a}\}$ and $O_2 = \{C_2, \texttt{is\_a}\}$ together with their corresponding sets of documents $D_1 = \{\mathbf{d}_1^1, ..., \mathbf{d}_{m_1}^1\}$ and $D_2 = \{\mathbf{d}_1^2, ..., \mathbf{d}_{m_2}^2\}$, where each document is represented as an n-dimensional TF/IDF vector (as described in Chapter 2 (Section 2.4) and in [74]) and m_1 and m_2 are positive integers. Let, additionally, subsets of the document sets D_1 and D_2 be assigned to the nodes of O_1 and O_2 through the map g (see (4.2)).

An instance-based mapping will be defined in two steps. First, we consider the concept-to-concept mapping:

$$sim : C_1 \times C_2 \to \mathbb{R}, \tag{4.9}$$

where sim is a properly chosen measure of similarity. Further, the instance sets of two mapped concepts are mapped to their union, i.e. $\forall A \in C_1$, $\forall B \in C_2$: $sim(A, B) > l \Rightarrow \{g(A), g(B)\} \to g(A) \cup g(B)$, where $g(A)$ is the subset of the document set D_1 which corresponds to the concept A, as defined in (4.2), and l is a parameter in the range of sim. Plausible choices of this parameter are discussed further in the chapter and in the experimental results of the thesis.

We define an instance-based or extensional mapping in the following manner:

$$M_{ib} : \{C_1 \times C_2, g(C_1), g(C_2)\} \to \{sim(C_1 \times C_2), 2^{D_1} \bigcup 2^{D_2}\}, \tag{4.10}$$

where $2^{D_1} \bigcup 2^{D_2}$ denotes all possible pair-wise unions of the elements of 2^{D_1} and 2^{D_2}. The above mapping will be equally represented as

$$M_{ib}(\{C_1, C_2\}, \{g(C_1), g(C_2)\}).$$

Finally, we remark that the documents in both sets D_1 and D_2 are based on the same set of attributes, which can be assumed without loss of generality. It is also assumed that O_1 and O_2 share a significant extensional overlap, i.e. their corresponding document sets intersect as it has been specified in Section 4.1.3 and all the documents are in the same natural language.

We comment on the last two assumptions. We will see that for what it concerns our approaches to instance-based concept mapping (presented below), the document sets intersection assumption is irrelevant – the concepts to be compared may be constituted of non-intersecting sets of documents. The intersection requirement is important, however, for the performance of one of the two suggested overall approaches, described in Section 4.5 of this chapter.

Regarding the second assumption, we do not discard the importance of ontology mapping in multi-lingual environments. Documents or the set of their frequently occurring terms can be translated automatically into a chosen target language. Only for the sake of simplicity this step is omitted from the formal description of the suggested mapping procedure. The problem is addressed in the concluding chapter of the thesis as a subject of future work (Section 6.2).

4.3.2 From Data Analysis to Concept Mapping

A major problem for most measures which provide concept similarity evaluation components to a global ontology matching system is that the number of pairs of concepts to be compared can become fairly large. Even if the measure performs excellently on a small number of concepts it might fail to be of any use on a larger scale for reasons of poor time efficiency. In the sequel, we will discuss the possibility of detecting potential concept mappings by the help of descriptive statistics in order to reduce the number of concept pairs to be used as an input for a similarity measure. We rely on the fact that the structure of the instance sets of two ontologies which is important for their similarity can be revealed by the help of statistical analysis methods which capture and expose information on the class separation of the ontology instances. We introduce a geometrical interpretation technique for detecting mappings among the concepts of two hierarchical ontologies by the help of a principal components analysis (PCA) and discriminant analysis (DA).

Principal Components Analysis

Principal component analysis (PCA) is one of the most general data analytical methods known from descriptive statistics. It helps to extract the most essential structural information contained in a dataset and serves as a basis for different methods of discrimination, classification or regression [77]. It is based on constructing new features, or principal components, by solving an Eigenvalue problem. The principal components are linear combinations of the original input variables[6] and are the new coordinates by which we represent the data. They approximate the data in the best possible way by recurrently capturing the directions of the biggest dispersion. Thus, PCA allows the representation of a multivariate data table containing thousands of variables in a lower-dimensional space (2 to 5 dimensions) by preserving and revealing the essential structural information contained in the data. In result, PCA shows what was not explicitly seen before: outliers or groups of instances are revealed; important information about the relations between variables and instances on one hand, and in-between variable relations, on the other hand is made available. These properties of the method can be extremely useful to obtain a first big picture over a dataset, in our case – the sets of instances of two source ontologies. Projecting our ontology data onto the first two principal components allows us to visualize and study basic dependencies and relations among sets of individuals or features.

The approach that we suggest consists in the following. We insist that the ontologies have been populated following a non-hierarchical instantiation (i.e. a document is assigned to one concept only and not to all of its predecessors as well). Let D be a set of documents assigned to an ontology O with a set of concepts C. We define $l : D \mapsto C$ to be the injection which assigns to each document the label of the concept of which this document is an instance. Through l, every document is identified by its class only.

Let $O_1 := (C_1, \texttt{is\_a})$ and $O_2 := (C_2, \texttt{is\_a})$ be two hierarchical ontologies and let D_1 and D_2 be their corresponding document sets of cardinalities m_1 and m_2, respectively. Let each element of the document sets has been labeled by the function l. Our goal is to find the mapping M_{ib} for every pair of concepts (A, B) such that $A \in C_1$ and $B \in C_2$. We produce a new dataset by taking the union of both document sets and the labels of their elements and let $D_{1,2}$ be

[6]The term "variable" in statistics stands for the commonly used terms "attribute" or "feature" in computer science. It denotes the original input variables while the term "feature" denotes the variables that have been created out of the inputs.

Var./Obs.	Var_1	Var_2	$\cdots$	Var_n	Concept
$\mathbf{d_1^1}$			$\cdots$		A_1
$\mathbf{d_2^1}$			$\cdots$		A_2
$\vdots$			$\vdots$		$\vdots$
$\mathbf{d_{m_1}^1}$			$\cdots$		A_1
$\mathbf{d_1^2}$			$\cdots$		B_1
$\mathbf{d_2^2}$			$\cdots$		B_1
$\vdots$			$\vdots$		$\vdots$
$\mathbf{d_{m_2}^2}$			$\cdot$		B_2

Table 4.1: Data-table with documents from two ontologies.

that set. Thus we come up with a multivariate data table that contains n real variables – the dimensions of the TF/IDF vectors, one categorical variable (the class) and m_1+m_2 observations[7] – the labeled documents from the two ontology document sets. Table 4.1 shows an example of such a table (the labels A_i and B_j $(i, j \in \{1, 2\})$ stand for names of concepts from two different ontologies).

We proceed to carry out a Principal Components Analysis on the set $D_{1,2}$. Since all our observations now live in one single space, PCA will project all documents from both ontologies in a single principal components space. As we already noted, PCA shows how observations are regrouped and thus identifies the existence of classes and their relations. Naturally, all documents belonging to one single concept (no matter from which ontology) will appear grouped together[8]. What is more, documents that belong to two or more different concepts from two different ontologies will also appear to be grouped together if they are instances of similar concepts. What remains is to take the labels of the documents which form one single group in the principal components projection and identify a mapping between the corresponding concepts.

An example illustrating the procedure is given in Figure 4.7. We see documents from concepts A1, A2, A3 and A4 from one ontology and documents from concepts B1, B2 and B3 from another. PCA shows that there are four main groups of observations. What our procedure suggests is that the documents that are grouped together in the PCA plot are instances of concepts which are to be mapped, i.e. A1 is mapped to B2, A2 – to B3, A4 – to B1, and A3 has no match in the second source ontology.

One straightforward problem with the proposed procedure is that the principal components analysis relies on a list of normality and linearity assumptions, which in many cases appear to be too strong restrictions. Applying a non-linear version of the PCA, like the one introduced in [15] or [134], based on using dot products in a feature space in terms of kernels in the input space could help overcome some of these assumptions. However, distribution related assumptions will still be on the way.

[7]The term "observations" is common for denoting the examples (or instances) in a dataset.

[8]As a matter of fact, the data *is* grouped together in its original representation (in a space of hundreds or thousands of dimensions). As stated above, the groups only become visible via the PCA-technique.

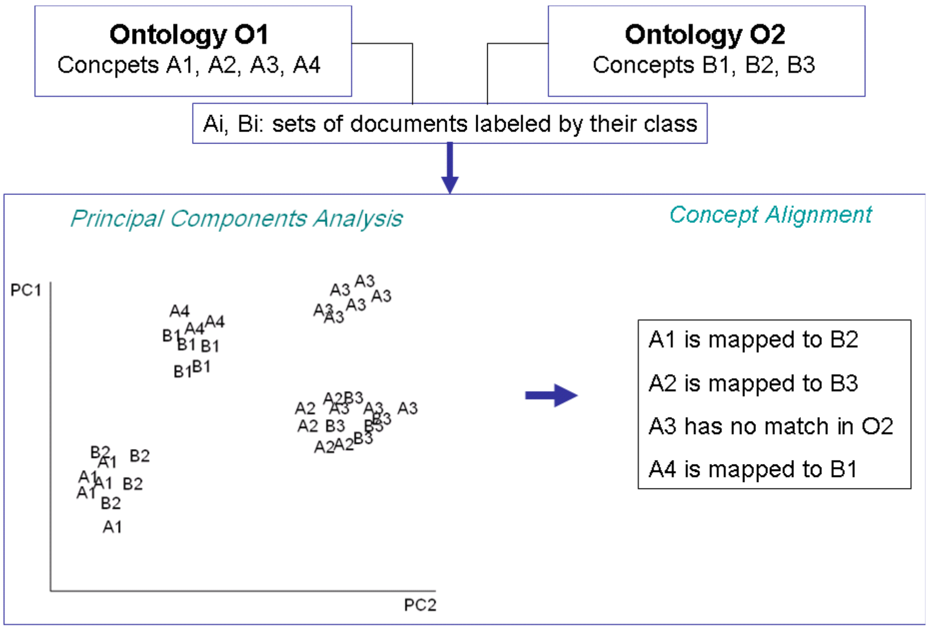


Figure 4.7: Example: PCA-based concept mappings.

Discriminant Analysis

PCA finds principal components by describing as much variance of the data as possible. However, in practice the first components may not (and often will not) reveal the class structure that we need. Discriminant analysis, originally introduced by Fisher [48], comes in to compensate for that drawback.

Similarly to PCA, discriminant analysis is also based on constructing principal (discriminant) axes but with the explicit objective to capture the separation of the classes by minimizing their in-class variation and maximizing the distances between their means. The class information has to be included in the input data from the start. The resulting discriminant axes are again linear combinations of the input variables. The variables with greatest weights for the construction of a given axis are the most important ones for the class separation projected on this axis. This gives rise to one of the most popular applications of DA analysis as a variable selection tool in class discrimination problems.

For an illustration, see Figure 4.8: represented is a population of two groups of observations (the blue and the red clusters) projected onto the first two discriminant axes (DA1 and DA2). DA1 provides a good separation of the two classes, unlike DA2 and the variables which are most significant for the construction of the axis DA1 are the ones which are most important for the separation of the blue and the red classes. We will come back to the ability of the method to score variables with respect to their discriminative qualities later on in next section and in our experimental studies.

In order to identify corresponding concepts, we apply a geometrical approach similar to the PCA case. We take an input dataset $D_{1,2}$, as introduced above and by the help of a discriminant analysis, we identify overlaps of groups of observations. Our argumentation goes as before:

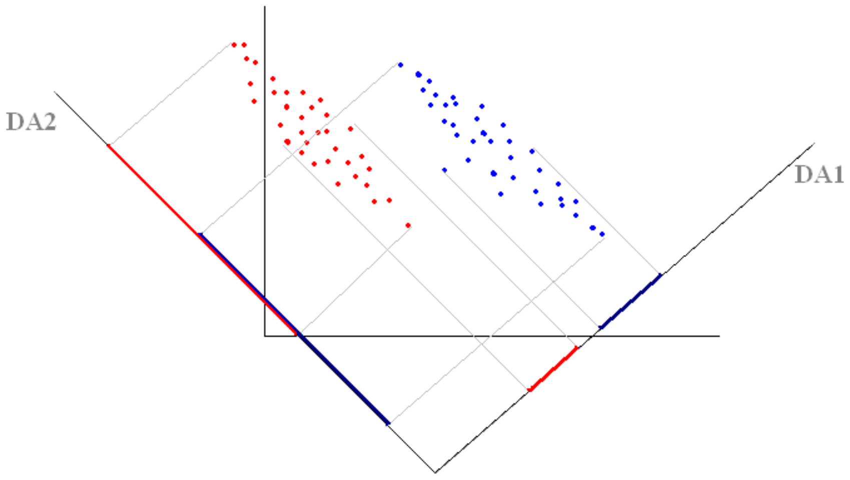


Figure 4.8: Example: discriminant analysis.

if two (or more) classes of observations that belong to two different input ontologies appear to overlap when projected on the DA discriminant axes, the concepts of which they are instances are assigned a mapping of a similarity degree according to the size of the overlap or the distance between the classes.

Since the basic motivation of DA is to provide a proper separation of previously given classes, it is a reasonable suggestion that those classes between which DA cannot properly discriminate are similar (see Figure 4.9 for an illustration).

Finally, we note that a kernel version of the discriminant analysis handling nonlinear cases has been elaborated in [105].

4.3.3 Variable Selection as a Measure of Concept Similarity

We proceed to propose a novel measure of instance-based concept similarity using variable selection for class discrimination. The instances in our study, we recall, are natural text documents assigned to the nodes of each ontology and coded as TF/IDF vectors [74]. Variable selection mechanisms are used to find variables (terms in the TF/IDF vectors), which are most characteristic for a given concept and play the most important role for separating its instances from the rest of the instances of the same ontology. The proposed measure of similarity is based on coinciding most important variables for two concepts taken from different ontologies.

The choice of a variable selection procedure within this setting is left to the user. However, we propose a novel selection criterion elaborated for Support Vector Machines (SVMs) (Section 4.3.4), arguing that it potentially outperforms standard selection techniques. The viability of the proposed concept similarity measure is demonstrated by experiments on real world textual data carried out by the help of variable selection procedures based on discriminant analysis and other standard selection techniques, such as mutual information, document fre-

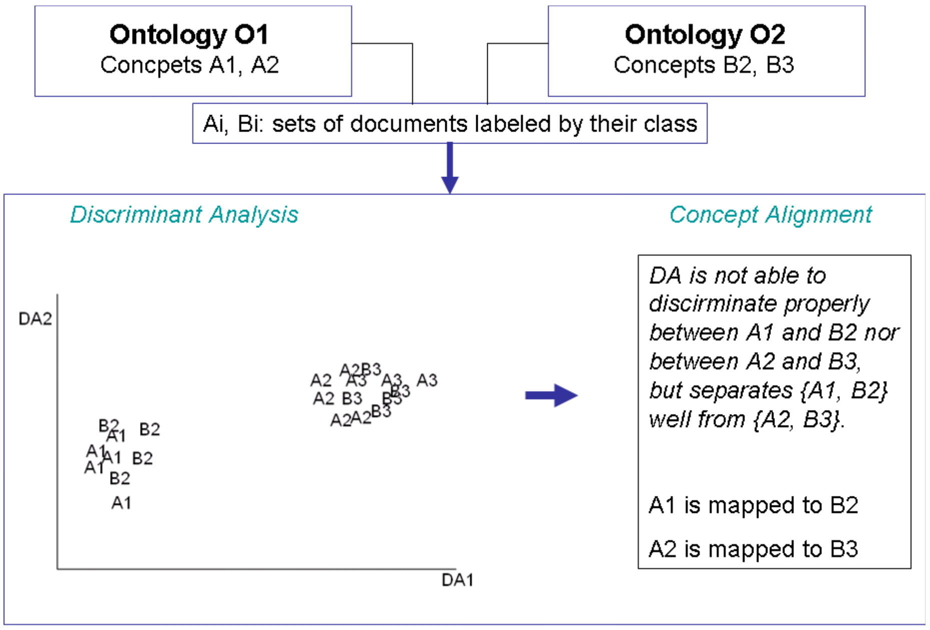

Ontology O1
Concpets A1, A2
Ontology O2
Concepts B2, B3
Ai, Bi: sets of documents labeled by their class
Discriminant Analysis
Concept Alignment
DA2
DA1
DA is not able to discriminate properly between A1 and B2 nor between A2 and B3, but separates {A1, B2} well from {A2, B3}.
A1 is mapped to B2
A2 is mapped to B3

Figure 4.9: Example: DA-based concept mappings.

quency thresholding and Chi-square statistics-based feature selection (Chapter 5).

Concept Similarity via Variable Selection

Variable (or feature) selection has been discussed in detail in Section 2.3. We recall that in our study the emphasis falls on the application of variable selection as a method which helps to find out more about the input – output relation in a given data set by pointing out the input variables, which most strongly affect the response.

We will introduce the type of datasets resulting from a specific instance-based representation of concepts within an ontology. The suggested machine learning procedure will be applied on sets of this kind. Let A be a concept from ontology O_1. We define a training data set $S^A = \{(\mathbf{d}_i^1, y_i^A)\}$, where $\mathbf{d}_i^1 \in \mathbb{R}^n$, $i = 1, ..., m_1$ and y_i^A are labels taking values $+1$ when the corresponding document $\mathbf{d}_i^1$ is assigned to A and -1 otherwise. The labels separate the documents in ontology O_1 into such that belong to the concept A (positive instances) and such that do not (negative instances).

The same representation and training data set can be acquired analogously for any given concept in both input ontologies O_1 and O_2. The similarity between two concepts A and B which belong to two different ontologies will be assessed by the help of their corresponding datasets S^A and S^B.

In our application scenario, for a given data set of the type S^A variable selection would indicate which of the TF/IDF vector dimensions are most important for the separation of the documents into such that belong to the concept A and such that do not.

We will see how a variable selection procedure can be applied to the task

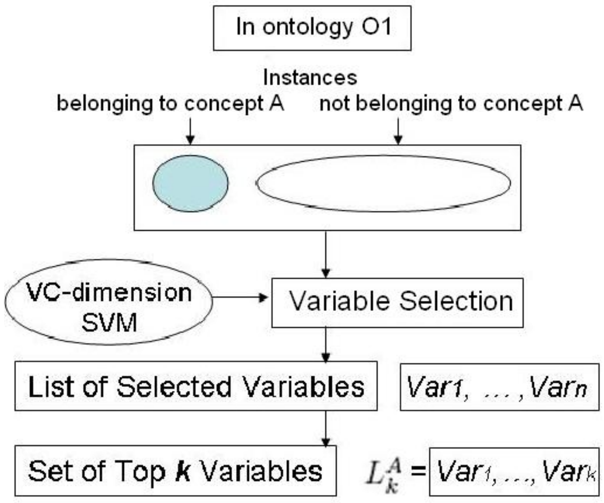


Figure 4.10: Variable selection for a concept A in ontology O_1.

of discovering concept similarities. We take as an input two concepts $A \in C_{O_1}$ and $B \in C_{O_2}$ together with their corresponding datasets $S^A = \{(\mathbf{d}_i^1, y_i^A)\}$, $i = 1, ..., m_1$ and $S^B = \{(\mathbf{d}_j^2, y_j^B)\}$, $j = 1, ..., m_2$ as introduced in Section 4.3.1. Our goal is to identify the degree of similarity between these two concepts. We carry out a variable selection procedure on each of the sets and order the variables from each of the sets by their importance for the class separation. Let

$$L^A = \{Var_{\sigma(1)}, Var_{\sigma(2)}, ..., Var_{\sigma(n)}\}$$

and

$$L^B = \{Var_{\delta(1)}, Var_{\delta(2)}, ..., Var_{\delta(n)}\}$$

be the ordered lists of variables for concepts A and B, respectively, where σ and δ are two permutations on the sets of variable indexes. We take from each of the lists a subset of the first k top ordered elements, where k is to be set by the user ($k < n$), and define the subsets $L_k^A = \{Var_{\sigma(i_1)}, Var_{\sigma(i_2)}, ..., Var_{\sigma(i_k)}\}$ and $L_k^B = \{Var_{\delta(j_1)}, Var_{\delta(j_2)}, ..., Var_{\delta(j_k)}\}$, $i, j \in (1, n)$, each of which contains the k most important variables for the separation of the instances in each corresponding ontology into such that belong to concept A, respectively B, and such that do not (Figure 5.6). The similarity of concepts A and B is given as

$$sim(A, B) = \frac{|L_k^A \cap L_k^B|}{k}, \tag{4.11}$$

with $sim(A, B) \in (0, 1)$. The defined measure of similarity fulfills the requirements for a similarity function (minimality, positiveness and symmetry), as introduced in definition 15 from Section 2.1. We will name this measure k-TF, from ***k* *T*op *F*eatures.**

In the experimental part of the thesis (Chapter 5), we will discuss in some detail the importance of the parameter k and its possible values. We mention that different values of k are appropriate with respect to different variable selection procedures used to form the sets L_k^A and L_k^B. However, a rule of thumb in choosing k's value is that it should be kept significantly lower than the number of variables in the dataset; otherwise one is running the risk of diluting the sets of selected variables with noise.

The concept or its complement?

Due to the nature of the introduced concept similarity criterion, there appears a certain ambiguity in the final similarity judgment. If a subset of variables is important for the separation of a given data set into classes B and $\overline{B}$ so is the same subset when we swap the two labels. The end result is that whenever our similarity measure $sim(A, B)$ yields 1 or a number close to 1 the following disjunction holds: {concept A is similar to concept B } or {concept A is similar to concept $\overline{B}$ } (the second possible disjunction, { $\overline{A}$ similar to $\overline{B}$ } or { $\overline{A}$ similar to B }, is complementary to the first one).

This undesired effect has been observed experimentally. We tested a VC-dimension-based similarity measure on the concepts of two small ontologies, O_1 and O_2, each containing two classes, such that class A_1 of O_1 was similar to class B_1 of O_2 and class A_2 of O_1 was similar to class B_2 of O_2. The selected most important variables for the class A_1 and the class A_2 were the same, so were the ones for the classes B_1 and B_2. In that the similarity measure yielded identical similarity values for each of the four pairs of classes.

However, this effect is most likely to be observed when there are only two or too few concepts from one of the ontologies to be mapped (for example, this can happen when mapping concepts from a high level of the ontology, where there is little granularity). If there are many concepts, which is often the case in real-life problems, this undesired effect is likely to disappear, for the complement of a concept will be a set of many other concepts; the characteristic variables of the complement of the concept will coincide only with the characteristic variables of the concept itself and no other concept in the dataset, for the complement of the concept is not an independent concept itself.

We will illustrate the former claim formally by a small example. Let us consider the two sets of concepts $O_1 = \{A_1, A_2, A_3\}$ and $O_2 = \{B_1, B_2, B_3\}$ taken from two different ontologies and let every set contain non-empty concepts, viewed as sets of instances, from a single level of the respective ontology (see Figure 4.11). In that the concepts are assumed to be pair-wise disjoint[9]: $A_1 \cap A_2 = \emptyset$, $A_1 \cap A_3 = \emptyset$, $A_2 \cap A_3 = \emptyset$, and $B_1 \cap B_2 = \emptyset$, $B_1 \cap B_3 = \emptyset$, $B_2 \cap B_3 = \emptyset$. We assume that a concept is entirely defined and exhaustive within a single ontology, i.e. that the concept's complement is contained entirely in the ontology, and not only partially. This is plausible, since it is difficult to talk about what a concept is and how it is defined out of the formal context of an ontology.

Now let us assume that the similarity measure (4.11) has identified the matches $\{A_1, B_1\}$, $\{A_2, B_2\}$, $\{A_3, B_3\}$ and thus the following disjunctions hold:

[9]The disjointness assumption is valid for tree-like ontologies and ontologies containing a hierarchical backbone, like the ones assumed in our model. However, this assumption might not hold for ontologies of more general kind.

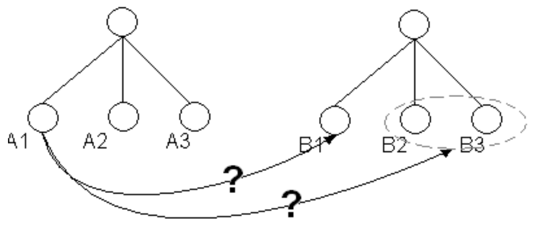


Figure 4.11: Example: the concept or its compliment?

$$\{A_1 = B_1\} \cup \{A_1 = \overline{B_1}\} \tag{4.12}$$

$$\{A_2 = B_2\} \cup \{A_2 = \overline{B_2}\} \tag{4.13}$$

$$\{A_3 = B_3\} \cup \{A_3 = \overline{B_3}\} \tag{4.14}$$

The complement of a concept B_1 is expressed as $\overline{B_1} = B_2 \cup B_3$. In the sequel, we will show that only the left hand side parts of the disjunctions above hold (the ones explicitly found by the similarity measure).

1. Let $A_1 = B_1$.

   Assume that $A_2 = \overline{B_2}$ ($\Rightarrow A_2 = B_1 \cup B_3$). We are in the case $A_1 = B_1$, which implies $A_2 = A_1 \cup B_3$. But, by assumption, $A_1 \cap A_2 = \emptyset$, therefore $A_2 = \overline{B_2}$ cannot hold and $A_2 = B_2$.

   Analogously, it follows that $A_3 = B_3$.

2. Let $A_1 = \overline{B_1}$ ($\Rightarrow A_1 = B_2 \cup B_3$).

   If $A_2 = B_2$ then it follows that $A_1 \cap A_2 \neq \emptyset$ – a contradiction to an assumption $\Rightarrow A_2 \neq B_2$. If $A_2 = \overline{B_2}$, i.e. $A_2 = B_1 \cup B_3$, it follows again $A_1 \cap A_2 \neq \emptyset$, which contradicts the assumption $\Rightarrow A_2 \neq \overline{B_2}$. We have arrived at the inconsistent conjunction $\{A_2 \neq B_2\} \cap \{A_2 \neq \overline{B_2}\}$. Hence, the only possible case is $A_1 = B_1$.

The same result can be derived analogously for the rest of the disjunctions in (4.13).

To sum up, we state that if a concept A in O_1 is to be mapped to the complement of a concept B in O_2, then all remaining concepts in O_1 have to be mapped to B, which is only reasonable in the case of two concepts and otherwise is a contradiction to the assumption of disjointness. Hence, the sketch of a proof above is valid for cardinalities of the sets of concepts greater than two. It is easy to verify that if there are only two concepts in one of the two sets of concepts the reasoning above does not hold and the effect of misjudging a concept for its complement (the second concept in the set) appears. In this case, we suggest to address the problem by the help of an approach which computes the statistical correlation between an attribute and the corresponding binary output estimated over the training data in order to get the desired sign information of for which of the two possible categories the selected features are relevant. Such approach can utilize, for instance, Mutual Information (discussed in previous sections) which explicitly computes the informativeness of a feature with regard to a given class.

Another way to approach this problem, is to rely on the *polarity* of the terms with respect to the categories. Given two concepts A and B and a term t, we count the number of documents from A in which t occurs and the number of documents in which it does not. We do the same for the documents in B. If t is relevant to A the number of documents in which it occurs in A should be bigger than those in which it does not and vice-versa for the concept B.

Parameter-free similarity measures

The similarity measure (4.11) depends on one parameter, k. Setting this parameter is not problematic, when we apply standard variable selection techniques. As seen in the experiments, once we have an algorithm for computing the similarity measure, trying out different values of k is inexpensive. Besides, the results corresponding to different values of k within a certain range are very competitive, in the majority of cases (see the relevant discussion in Chapter 5).

However, we point out that a parameter-free measure can be applied as well in order to avoid the manual setting of the parameter k. This is particularly helpful when using an SVM-based variable selection criterion, for in this case computations are more expensive. We propose to use standard parameter-free measures of rank correlations for two random variables – in our case, the two concepts whose similarity we want to measure. A measure of correlation standardly takes values between -1 and 1, assuming strong correlations for high positive values, strong anti-correlations for high negative values and no dependence for values close to 0.

The measures can be computed on the variables *scores* or on the corresponding variable *ranks*. As a result of the variable selection procedures, each concept in our two ontologies can be represented as a list of variables scores and ranks can be computed over these scores. The variables ranks are integers, such that $rank(Var_i) = 1$ if the score of the variable Var_i is minimal, and so on. Table 4.2 provides an example: r_i^A, $i \in (1, ..., n)$ stands for the rank corresponding to variable Var_i computed from its score, x_i^A, in result of the variable selection procedure performed for concept A.

We refer to the book by Conover [29] for an overview of rank correlation-based tests. We will introduce three of the most popular choices for correlation measures of two random variables and indicate how they can serve as similarity measures for two classes of instances.

- Spearman's coefficient [142] provides a measure of the correlation of two variables represented as lists of statistical rankings. Formally, it is given by

$$\rho = 1 - 6\frac{\sum_i d_i^2}{n(n^2-1)}, \tag{4.15}$$

  where d_i is the difference of the ranks calculated for the i-th pair of two corresponding variables (see Table 4.2) and n is the total number of observations[10]. In that, ρ acts as a similarity function – greater values of ρ would indicate higher similarity of the concepts.

[10]In this case the observations are the different rank values for each of the n variables; recall that the two random variables that we measure the correlation of are the two classes A and B.

	Scores (A)	Ranks (A)	Scores (B)	Ranks (B)	d_i
Var_1	x_1^A	r_1^A	y_1^B	r_1^B	$r_1^A - r_1^B$
Var_2	x_2^A	r_2^A	y_2^B	r_2^B	$r_2^A - r_2^B$
Var_3	x_3^A	r_3^A	y_3^B	r_3^B	$r_3^A - r_3^B$
Var_4	x_4^A	r_4^A	y_4^B	r_4^B	$r_4^A - r_4^B$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Var_n	x_n^A	r_n^A	y_n^B	r_n^B	$r_n^A - r_n^B$

Table 4.2: Concepts, variables, scores and ranks.

- Pearson's product moment correlation coefficient can be applied directly on the variables scores without transforming them into integer ranks. It is given by

$$r = \frac{\sum_{i=1}^{n}(x_i^A - x_{mean}^A)(y_i^B - y_{mean}^B)}{\sqrt{\sum_{i=1}^{n}(x_i^A - x_{mean}^A)^2}\sqrt{\sum_{i=1}^{n}(y_i^B - y_{mean}^B)^2}}, \tag{4.16}$$

  where x_i^A and y_i^B are the scores corresponding to the variables for the classes A and B and x_{mean}^A and y_{mean}^A are their mean values calculated over all $i = 1, ..., n$, $r \in [-1, 1]$[11].

  Note that Pearson is the general formulation of Spearman when there are no tied ranks. In case our data contain tied ranks, Pearson's coefficient can be computed over the ranks, instead of the scores, by the formula

$$r = \frac{\sum_{i=1}^{n}(r_i^A - r_{mean}^A)(r_i^B - r_{mean}^B)}{\sqrt{\sum_{i=1}^{n}(r_i^A - r_{mean}^A)^2}\sqrt{\sum_{i=1}^{n}(r_i^B - r_{mean}^B)^2}}, \tag{4.17}$$

  where r_{mean}^A and r_{mean}^B are the mean values over all variables ranks for each of the two classes. It can be shown that in this case Pearson is identical to Spearman (see, for instance, [29]).

- Finally, we introduce Kendall's τ, which only makes use of ordering information, given by

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)}, \tag{4.18}$$

  where n_c and n_d are the number of concordant and discordant pairs among the observations for two given classes. Two observations are called concordant if both members of one observation are larger than their respective members of the other observation. A pair of observations is called discordant if the two numbers in one observation differ in different directions from their corresponding numbers in the second observation. For example, the pairs {(2.2, 3.4), (2.8, 4.1)} and {(5.2, 5.7) (2.1, 3.6)} are concordant, whereas the pair {(3.7, 1.5),(2.8, 3.2)} is discordant.

[11]Pearson's correlation usually and here is denoted by r and is not to be confused with the variables rankings.

As mentioned, the measures presented above are measures of correlations in the range $[-1, 1]$ and therefore are not properly measures of similarity. In our case, for two concepts A and B, a high positive correlation indicates that whenever a variable has a high score for concept A, so it does for the concept B (the same variables are important for the two concepts). A correlation close to 0 means that no such dependency exists, a high negative correlation indicates that whenever a variable reaches a high score for A, it has a low score for B (i.e. it is certainly not the same variables which characterize the two concepts). Since having the same variables characterizing two concepts is our basic similarity criterion, similar concepts are judged to be those with a high positive coefficient, and not those with a coefficient simply above 0. Therefore, as we will see in our experimental results, a similarity threshold of 0.5 is sensible. A properly defined similarity funtion based on either of the measures of correlation above would look like that:

$$sim(A, B) = \max\{0, corr(A, B)\}, \tag{4.19}$$

where $corr(A, B)$ stands for either of the measures ρ, r or τ applied on two copncepts A and B and $sim(A, B) \in [0, 1]$.

In the experimental part of the thesis (Chapter 5), we will discuss and compare the performance of the parameter-free coefficients introduced above and the k-TF similarity measure (4.11).

4.3.4 VC-dimension-based Variable Selection for SVMs

As we have seen in Section 2.2, the support vector machines have many attractive sides, particularly when dealing with text data - their performance does not depend on the distribution of the data (safe that they are i.i.d.), it does not demand a linear input-output relation and they are easy to implement. At least theoretically, the generalization properties of SVMs do not dependent on the size of the input space which makes variable selection little prominent for learning with SVMs. However, the listed properties turn them into a good candidate for a variable selection tool to be used self-dependently. In addition to that, some authors have shown that even though theoretically unnecessary, variable selection improves SVM learning in practice [126].

We have discussed different efforts in SVM-based variable selection during the last decade in Chapter 2. Most of the existing endeavors are based on the variations of the weight vector with respect to a variable of the original dataset. The variable selection criterion that we propose is based on the sensitivity of the VC dimension of the SVM classifiers with respect to a single variable or a block of variables. As we have seen in Section 4.3.1, for different values of the VC dimension h, different values of the VC confidence (describing the capacity of the classifier) will be computed and thus different bounds on the actual risk (2.19), where from the generalization power of the classifier will change. Our main heuristics can be formulated as

> *a less informative variable is one, which the VC confidence of the classifier is less sensitive to.*

For computational reasons, the evaluation function of our variable selection procedure will be formulated in terms of VC dimension directly, instead of

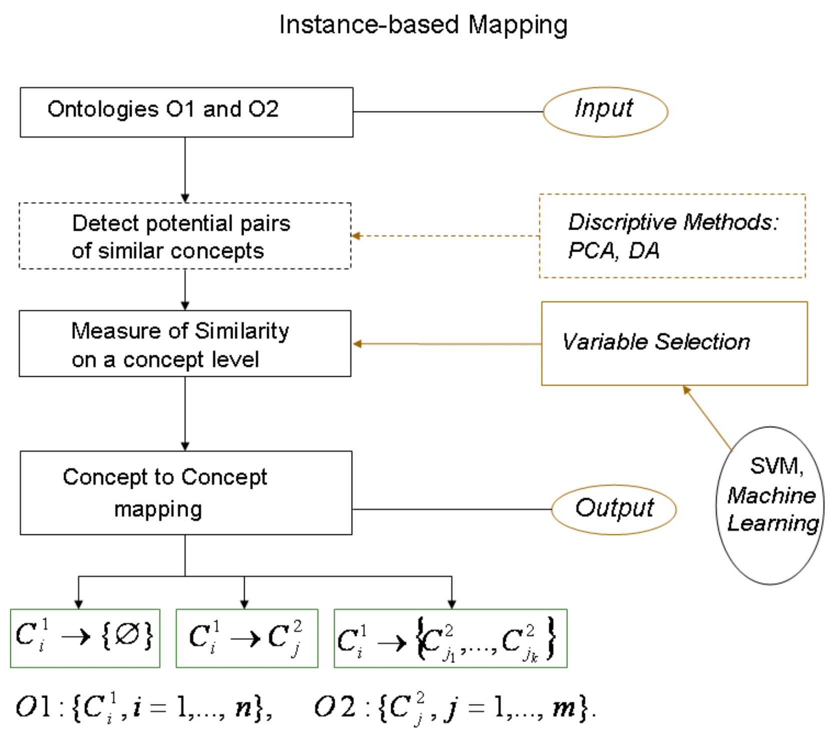

Instance-based Mapping
Ontologies O1 and O2
Input
Detect potential pairs of similar concepts
Discriptive Methods: PCA, DA
Measure of Similarity on a concept level
Variable Selection
SVM, Machine Learning
Concept to Concept mapping
Output
$C_i^1 \to \{\varnothing\}$
$C_i^1 \to C_j^2$
$C_i^1 \to \{C_{j_1}^2, \ldots, C_{j_k}^2\}$
$O1 : \{C_i^1, i = 1, \ldots, n\}, \quad O2 : \{C_j^2, j = 1, \ldots, m\}.$

Figure 4.12: A possible architecture of instance-based mapping.

in terms of the VC confidence. This is plausible since the VC confidence is monotonous in h. Thus, the i-th variable is evaluated by

$$eval_i = h(H) - h(H^{(i)}),\ i = 1, \ldots n, \tag{4.20}$$

where $h(H)$ is the VC dimension of a set of SVM hypotheses H constructed over the entire data set and $h(H^{(i)})$ is the same quantity computed after the removal of the i-th variable in the data set (this is the variable whose pertinence is to be evaluated).

As a final remark, we note that in general, it is difficult to compute the VC-dimension directly. But in the case of the SVMs, we can compute an upper bound for the VC-dimension depending on the resulting weight vector and on properties of the given data (Lemma 25 introduces this bound). In the evaluation of the proposed variable selection technique, presented in Section **??**, we have used SVM*light* [75] where the VC dimension is estimated based on the radius of the support vectors.

The components of instance-based ontology mapping, which have been introduced and elaborated theoretically in the sections above are depicted in Figure 4.12 combined together in a possible matching architecture.

4.4 An Overall Variable Selection-Based Ontology Matching Procedure

In the previous couple of sections, we have introduced and discussed concept-to-concept mappings achieved by the help of an instance-based similarity measure, which uses variable selection methods. We have not yet specified how this measure can be applied in an optimal way for the task of asserting similarities of all possible pairs of concepts of two source ontologies. Clearly, simply taking the two sets and measuring one-to-one similarity values would be not only time and complexity inefficient[12], but also conceptually unjustified, for in this case the structure of the classes in both ontologies would not be taken into account. Structure and rich semantics is what differentiates ontologies from other types of data representations and therefore simply considering flat sets of concepts is not a reliable method of ontology matching. In addition to that, there is a practical problem in doing so. In the section above, we have introduced a variable selection procedure based on binary support vector machines. Now let us imagine that in the first of our source ontologies we have the classes "Hardware-Mac" and "Hardware-PC" among other classes, such as "Religion" and "Politics". When an SVM-based variable selection procedure attempts to identify the characteristic variables for the class "Hardware-Mac", it will stumble upon a problem: the instances in the dataset will be labeled positive for that class and negative for all other classes, including "Hardware-PC", which is much closer semantically to "Hardware-Mac" than "Religion" or "Politics". In that, the instances of "Hardware-PC" will appear to the classifier as errors, rather than members of the negative class.

In order to account for this problem and to embody the structure of the two ontologies, we propose an overall procedure for ontology matching. We suggest that the similarity measure should be applied iteratively for the sets of concepts on corresponding levels of the two ontologies, descending down the tree structure. In a properly designed ontology the classes on a single level are homogeneous and non-intersecting and the undesired effect described above will not be produced. We will first describe the procedure for hierarchical ontologies (containing only `is_a` relations) with identical number of levels and identical level of detail - a fairly restricted case, but one which allows to explain the core of the algorithm. Building on that particular case, we will further present an extended version of this procedure accounting for non-hierarchical ontologies of general kind.

4.4.1 Matching Hierarchical Ontologies

Consider we have two ontologies both containing the same number of levels, equally granular and specific. In the first step, one takes only the classes of the first level of the two ontologies (the direct descendants of the top concepts) and measures their similarity[13]. Let the corresponding collection of concepts be labeled C_A and C_B. If a concept A from C_A is found to be similar to a concept B from C_B, the procedure is repeated for the two sub-trees which start in A

[12]Mapping an ontology with n concepts to an ontology with m concepts (performing all possible $n \times m$ mappings) can become prohibitive for high values of n and m.

[13]We assume that conceps are instantiated hierarchically (see Section 4.1.2).

and B, respectively.

We will illustrate this idea with one example. Let us consider the two ontologies depicted in Figure 4.13a. Our task is to identify which concepts of the ontology on the left hand side of the figure (let that be ontology O_A) are similar to which concepts of the right hand side ontology (ontology O_B). As suggested, we start by matching the set of concepts $\{A_1, A_2, A_3\}$ against the set $\{B_1, B_2, B_3, B_4\}$. Let the mappings identified by the help of the variable selection based similarity measure be

$$A_1 \to B_4$$
$$A_2 \to \emptyset$$
$$A_3 \to B_1.$$

We proceed to map the sets of the children of each pair of concepts that were mapped in the first step. That means to match the set $\{A_{11}, A_{12}, A_{13}\}$ against $\{B_{41}, B_{42}\}$ and the set $\{A_{31}, A_{32}, A_{33}\}$ against $\{B_{11}, B_{12}, B_{13}\}$. Let the mappings found in the first case be

$$A_{11} \to B_{42}$$
$$A_{12} \to B_{41}$$
$$A_{13} \to \emptyset.$$

The descendants from A_{11} and those from B_{42} are further mapped in the same manner. We proceed analogously for all other pairs of mapped concepts until reaching the leaves of the tree O_1. A sketch is given in Figure 4.13b.

4.4.2 Generalizations of the Procedure

If our source ontologies are more complex than the ones considered above, contain different number of levels and are of different granularities, we have to insure that at each step we are matching corresponding levels. To these ends, we suggest either using anchoring techniques (as described in [114]), or setting a threshold of the measured concept similarity: if the values found are under that threshold, the levels do not correspond and we should descend on the following level of O_2. As a result of this verification, the root of one of the ontologies can be mapped to a non-root node of the other one and start the procedure from there on. By using the similarity threshold technique, this can be achieved by mapping the most similar nodes from both ontologies and setting them as roots in the procedure, or by performing all possible mappings for the root node of the first ontology to the nodes of the second one and selecting the best result as a starting point of the procedure. In that way the described approach is generalized for ontologies of different *sizes*, i.e. which do not necessarily contain identical numbers of levels in their respective hierarchies (see Figure 4.14).

Finally, we suggest that the procedure can be generalized in one more direction by adapting it for ontologies that do not contain strictly hierarchical relations binding their concepts. As long as a hierarchical structure can be extracted out of the ontologies, i.e. as long as there is a hierarchy-like backbone underlying the structure of these ontologies, the suggested procedure can be applied in the following steps.

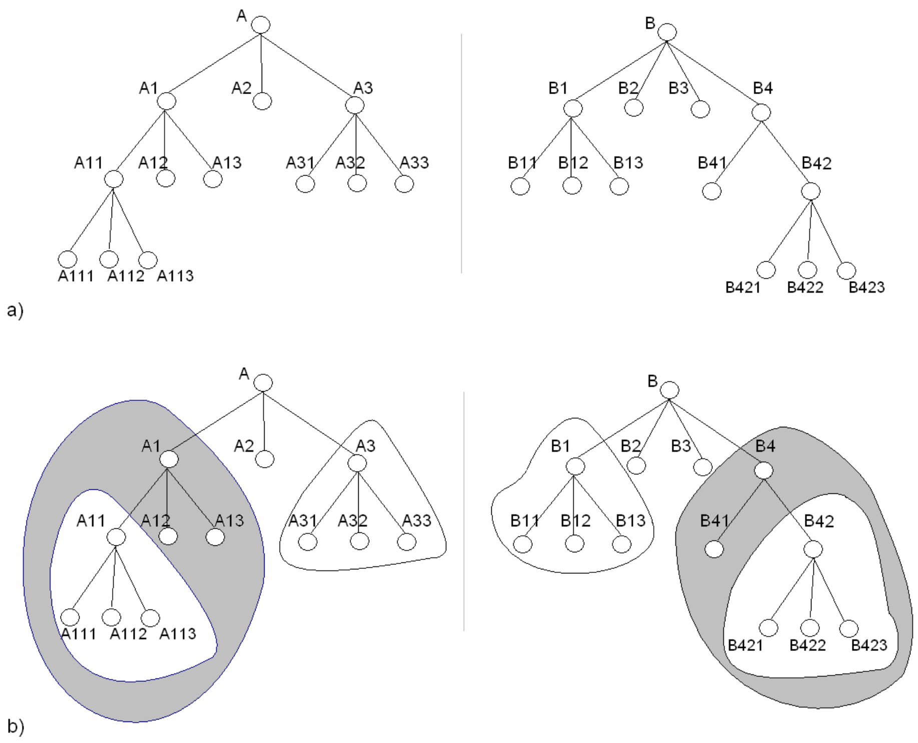


Figure 4.13: Instance-based mapping with variable selection.

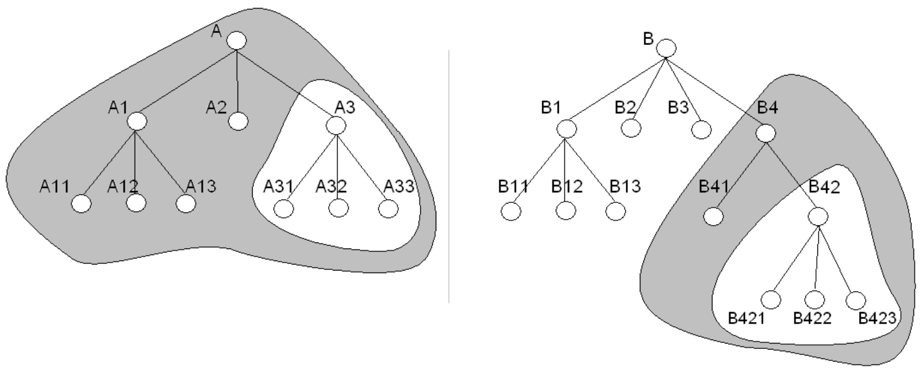


Figure 4.14: Mapping ontologies of different sizes.

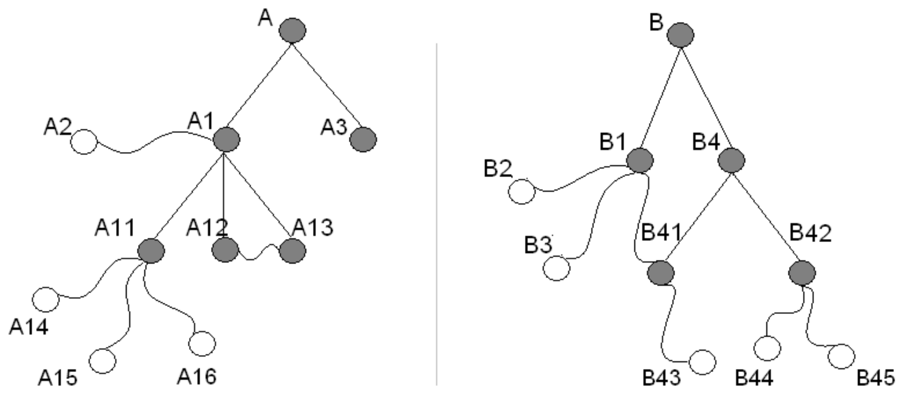


Figure 4.15: Mapping non-hierarchical ontologies.

1. Extract a hierarchy out of every input ontology. That means to "strip down" the ontologies to their sub-ontologies, which only contain subsumption relations and those concepts, which are connected by these relations. Let us call these structures "hierarchical backbones" of the ontologies.

2. Apply the overall variable-selection-based procedure described above on the hierarchical backbones and identify the similar pairs of concepts. Let these form the set of concept correspondences Σ_{hier}.

3. Back to the input ontologies: mark all concepts that are nodes in the ontologies backbones. Consider one after the other every pair of concepts which have been found to be similar in the preceding step and look for unmarked concepts, which are related to them (by a relation of some other kind). Proceed to check the similarity of these concepts by applying a variable selection-based similarity measure. Let the set of unmarked concepts, identified as similar at this step, be denoted Σ_{non}.

4. Add the new similarity assertions (if any are found) to the set of correspondences which came as an outcome of Step 2 and define the final set of concept mappings as the set $\Sigma = \Sigma_{non} \cup \Sigma_{hier}$.

For an example, take a look at the ontologies depicted in Figure 4.15. As usually, the circles represent concepts and the straight lines connecting them – `is_a` relations. The curved lines connecting concepts indicate relations of some other kind (these might be standard `part_of` relations or user defined more specific relations, such as `employed_by`, `graduated_at`, etc.). The shaded nodes represent the "marked nodes" – those, which form the ontologies backbones.

In the first two steps of the suggested procedure, the set of shaded nodes from the first ontology will be mapped to the set of shaded nodes of the second ontology, following the overall matching procedure described in the beginning of the section, yielding the set of pairs of concepts Σ_{hier} (containing pairs of concepts that have been asserted "similar"). Let the mapping $A_1 \to B_1$ be an element of this set, as well as the mapping $A_{11} \to B_{41}$.

In Step 3 of the procedure described above, one will proceed to map the sets of concepts related by non-sumbsumptional relations to A_1 and B_1 and to A_{11} and B_{41}. In other words, one will perform the matching $\{A_2\}$ against $\{B_2, B_3\}$ and the matching $\{A_{14}, A_{15}, A_{16}\}$ against $\{B_{43}\}$ (note that the node B_1, although related to B_{43} by a non-subsumptional relation will not be included in the similarity check at this step because it is marked and its similarity has already been accounted for in a previous step of the procedure). The pairs of similar concepts found at this step will form the set of correspondences Σ_{non}.

The final step of the procedure simply suggests to take the union of the two sets of mappings derived, respectively, at Step 2 and at Step 3 of the procedure.

4.5 Specificity and Granularity Judgments for Ontology Matching and Merging

In sections 4.2 and 4.3, we have discussed two separate aspects of measuring ontology similarity. The first one concerns considering purely structural similarity indications (Section 4.2) and makes use of a graph distance metric defined in terms of the maximal common sub-graph of the input ontology graphs. The second aspect describes a set of techniques, which look into the extensions of the ontologies in order to decide on the degree of their semantic similarity by employing an instance-based measure of concept similarity (Section 4.3).

The introduced approaches will be our basic instruments used to build a combined procedure for matching taxonomic ontologies populated with text documents. We note that a third important, although quite straightforward criterion for concept similarity is the presence of identical or similar concept names in the ontology concept sets. We will not include this similarity criterion in the study, assuming that there are no concepts with identical or similar names in our input ontologies for the sake of simplicity of the presented approach. In addition to that, we consider the lexical criteria to be in general weaker and less reliable indication of similarity than the instance-based and structural techniques, because of the many problems arising from polysemy and synonymy. Precisely for that reason lexical matching accounts are never to be used self-dependently, but rather in combination with prior knowledge or dictionaries. Nonetheless, we keep in mind that the name-based criterion can be of help to obtain initial information on which would be potential matches among two ontologies and therefore it can be used alongside the proposed techniques.

The procedure we are about to describe relies for a starting point on the instances populating the ontologies. We can formulate our main heuristics as: *"Intersecting instance sets*[14] *indicate intensional overlap."* The structural and conceptual similarity assessments come into play only for ontologies, which have been found to share a certain extensional overlap, i.e. the intersection of the instance sets comes as a measure of an overall ontology similarity and gives an idea whether the ontologies are intensionally close or remote (see an extended discussion on the topic of overall ontology similarity in [34]). Further on, when we find a common structure in both ontologies, provided that they are built on the same or similar sets of documents (or on two sets with a significant

[14]Intersection is understood again in the sense of the modified relaxed intersection introduced above.

generalized (semantical) intersection) we find the correspondences between the nodes by applying the instance-based node-to-node mapping. Assuming that the two trees are structurally different, after we map the root node of the smaller tree to a node of the bigger one, we already know a lot about where the smaller tree is situated in the bigger one. An interesting case is when the two root nodes match, and still one of the trees is smaller than the other. Provided that the two trees are structurally the same, a discrepancy might appear on an extensional level: one of the trees might have less instances than the other or the sets of instances of the two ontologies might only overlap partially.

All of these cases are treated separately in the remainder of the chapter. As a final outcome, the procedure yields assertions on how specific or granular one ontology is compared to the other and also how do both ontologies differ in terms of instantiation (which ontology is extensionally richer than the other). On these bases, the user is given propositions for ontology merging, if necessary or required. The different possible cases are ordered from most specific to most general in terms of the structure-extension relation.

4.5.1 Preliminaries

Let us take as an input again the ontologies O_1 and O_2 together with their corresponding document sets D_1 and D_2 as well as their graph representations G_1 and G_2. The procedure to be presented has the following main components:

- The structural distance between the two input ontologies given by $d(G_1, G_2)$;

- The relative size of the (modified relaxed) intersection of the sets D_1 and D_2 expressed by

$$r_{\Delta} = \frac{|D_1 \cap D_2|}{|D_1|}, \tag{4.21}$$

  where $|D_1|$ is the smaller of the two sets (or either of them, when they are of equal cardinalities).

- Instance-based mapping criteria

- Lexical matching component[15]

With respect to each of the components above, we distinguish between different cases and we will concisely describe and comment them separately, before we introduce them combined in the overall procedure in Section 4.5.2.

[15]The lexical matching component is mentioned for completeness; it is a matter of future work to make it an integral part of the presented procedure.

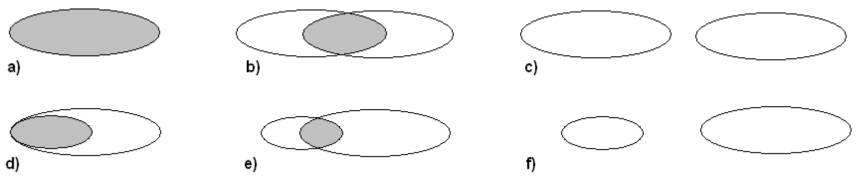


Figure 4.16: Intersections of document sets.

Structural distance

The structural measure of graph similarity that we apply is based on the size of the maximal common sub-graph of both graphs relative to the size of the bigger graph of the both. Let $G_0 = mcs(G_1, G_2)$. We distinguish between the following cases.

1. $d(G_1, G_2) = 0$. This implies that the maximal common subgraph of both graphs is identical to each of them ($G_0 = G_1 = G_2$).

2. $1 > d(G_1, G_2) > 0$ and $G_0 = G_1 \subset G_2$. The maximal common subgraph is identical to the smaller of the two trees (the choice of the smaller graph is conventional; choosing otherwise does not substantially influence the ensuing reasoning).

3. $1 > d(G_1, G_2) > 0$ and $G_0 \subset G_1$ and $G_0 \subset G_2$. This is the most general case when the maximal common subgraph of both trees is a subgraph of each of them[16].

The document sets intersection

We will distinguish between ontologies with respect to their document set intersection (again, in the sense of the generalized intersection) and sizes. Let us denote $D_1 \cap D_2 = D_0$.

As we have observed above, the relative size of intersection of the sets of documents is modeled by the quantity $r_\triangle$. Note that when the document sets are of similar cardinalities, we will have a situation similar to the one depicted in Figure 4.16a,b,c; when the cardinalities are different (i.e. the most probable case when one of the ontologies contains less documents than the other), the picture will be similar to the one depicted in Figure 4.16d,e,f. We list the possibilities which follow :

1. $r_\triangle = 1$. The two sets entirely overlap (Figure 4.16a or Figure 4.16d).

2. $0 < r_\triangle < 1$. The two sets do have a nonempty intersection which is smaller than each of the sets (Figure 4.16b or Figure 4.16e).

3. $r_\triangle = 0$. The intersection of both sets is the empty set (Figure 4.16c or Figure 4.16f).

[16]The fourth possible case $1 = d$ occurs only for two graphs one of which is the empty graph.

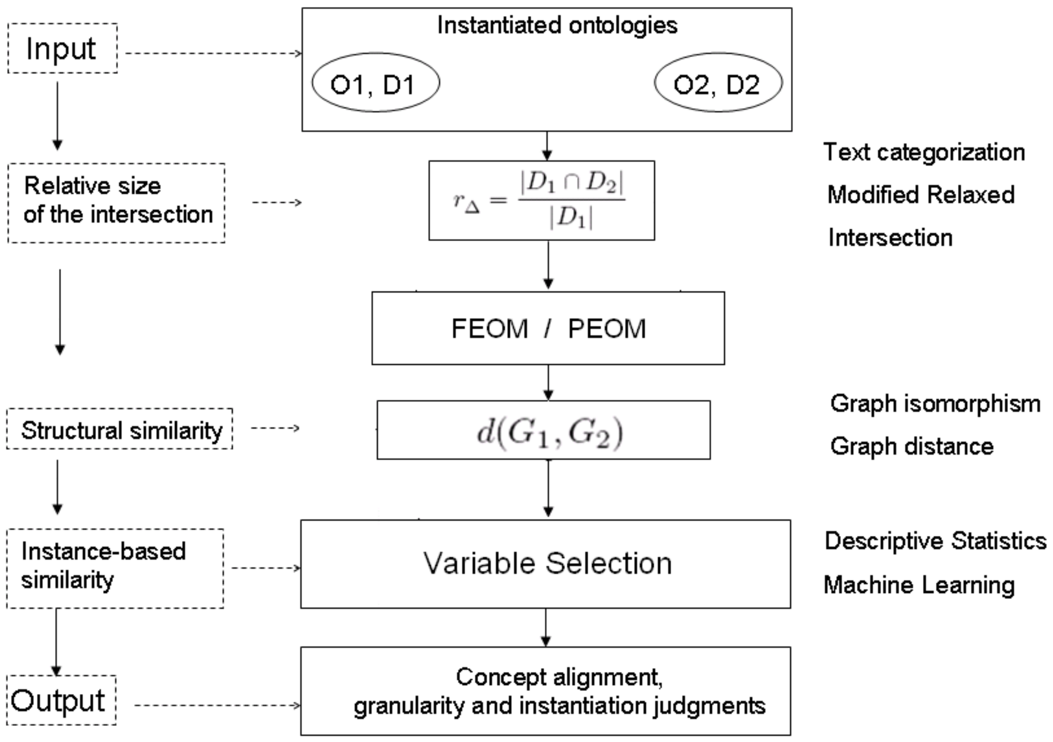


Figure 4.17: A procedure for ontology matching.

In Section 4.5.2, we will introduce the procedure FEOM (Full Extensional Overlap Matching), which covers the cases in point *one* from the list above and the procedure PEOM (Partial Extensional Overlap Matching), ruling out the cases from point *two*. If a case from point *three* is produced, the matching procedure is terminated at the very beginning, because a major assumption of our model is broken (see Section 4.1): the ontologies do not share any extensional overlap.

Instance-based concept mapping

An extensional approach to aligning concepts has been described in the preceding sections and a formal definition of the instance-based mapping M_{ib} is given in (4.10). We recall that an instance-based mapping is understood as a one-to-many mapping from the set of concepts of the first ontology to the set of concepts of the second ontology realized through a similarity function of some kind and an additional mapping of the sets of the documents corresponding to the mapped concepts to their union. Instance-based mapping, combined with structural and instance-sets intersection criteria, is an important component of the matching procedures to be described in the next sections.

Lexical criteria for concept mapping

The structural and extensional approaches described so far are our basic tools for ontology matching. A third important and quite straightforward criterion for concept similarity is the presence of identical or similar concept names in both ontology concept sets. Linguistic analysis approaches to this problem, relying on names and textual description of ontology elements, are used in [17] and [144]. A comparison of string distance metrics for matching names is found in [76, 158], as well as Section 2.1.3 and Section 2.4.4 of this thesis and the sources cited there.

4.5.2 Overcoming Granularity and Instantiation Discrepancies: A Procedure for Ontology Merging

We will present a procedure for ontology matching, which is based on exploring separately each of the cases resulting from combining the different possibiliies and matching components discussed in the section above, yielding as an output indications for a possible merging of the two source ontologies. The cases that are handled by the FEOM and the PEOM methods are described separately below; the cases of no extensional overlap, which are in conflict with our initial assumptions, have been omitted. Figure 4.17 provides an illustration of the procedure's architecture.

Full extensional overlap matching (FEOM)

As stated above, the FEOM procedure covers the cases when there is a complete overlap of the sets of instances of the two ontologies[17], i.e. $r_\Delta = 1$. The procedure starts by measuring the structural distance between the two input ontology graphs.

(A) **Case 1:** $d(G_1, G_2) = 0$. The first case contains taxonomies, which have identical structures. Depending on the relative degree of instantiation of the two ontologies, we distinguish between two further cases - either their extensions will be similar, as shown in Figure 4.18a, or on of the ontologies will have less instances than the other (Figure 4.18b). In both cases, we proceed to establish the precise concept-to-concept correspondences, or perform the mapping

$$M_{ib}(\{C_1, C_2\}, \{g(C_1), g(C_2)\}).$$

We recall that this means to match the set of concepts C_1 against the set of concepts C_2 by merging the instance-sets of all pairs of matched concepts, by the help of the *instance-based* concept aligning methods (introduced in Section 4.3). We note that if we are in the latter case ($D_1 \subset D_2$, Figure 4.18b), the merged ontology will result containing the bigger instance-set of the two.

(B) **Case 2:** $d(G_1, G_2) > 0$ and $G_0 = G_1 \subset G_2$. The second major case governs pairs of input ontologies which are structurally dissimilar in a way that one of the trees is a sub-tree of the other and is thus the maximal common sub-graph of both. Since the instance-sets are overlapping, we will perform again an instance-based mapping (by the help of the methods described in Section 4.3) from the concepts of ontology O_1 to the concepts

[17]The convention that "full" or "complete" overlap is to be understood under a fixed tolerance threshold holds.

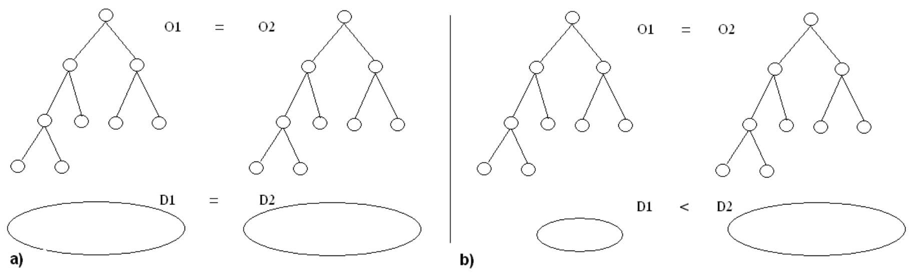


Figure 4.18: FEOM Case 1. Similar structures.

of ontology O_2. Note that with respect to how the two instance-sets are related, we can have one of the following sub-cases.

(a) *Case 2.1.* $D_1 = D_2$. One of the ontology trees is a subtree of the other, but the instance sets of the two are identical (Figure 4.19(a)): the ontology dissimilarity is entirely on the intensional level. We assert that O_2 is more specific than O_1 because it contains more concept nodes and branches in the tree.

(b) *Case 2.2.* $D_1 \subset D_2$. O_1 is both intensionally and extensionally a sub-taxonomy of O_2 (Figure 4.19b). O_1 will be, therefore, as a result of the instance-based matching entirely injected into O_2.

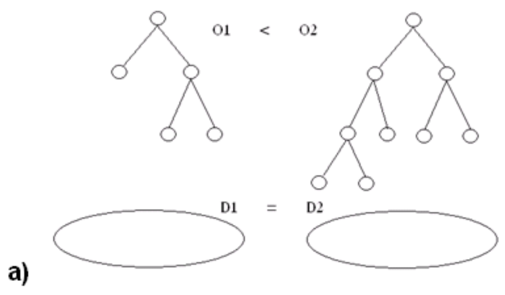
(c) *Case 2.3.* $D_2 \subset D_1$. O_2 is a bigger and more granular hierarchy, but less populated than O_1 (Figure 4.19c). The instance- based matching will result in taking instances from O_1 and assigning them to O_2.

(C) **Case 3:** $d(G_1, G_2) > 0$ and $G_0 \subset G_1$ and $G_0 \subset G_2$. This is the most general case of the maximal common subgraph being a subgraph of both input ontology graphs. Since we are working with trees, and not with graphs in general, the only possibility is that one of the trees is broader than the other (i.e. at at least one level it has more nodes than the other) and the other is deeper (has more levels) than the first. Let us assume that G_1 is the broader and G_2 – the deeper tree, as represented in Figure 4.20a.

We would like to map the ontologies in such a manner that both their breadth and depth is preserved in a resulting merged tree. Let G_0^1 denote the maximal common subgraph of G_1 and G_2 as a subgraph of G_1 and let D_1^0 denote its corresponding instance-set. We proceed to match the ontology O_0^1 (corresponding to the tree G_0^1) to the ontology O_2 - a task which has been solved in the case 2 above[18].

The concepts of O_1 which remain unmapped, i.e. the concepts corresponding to the nodes of the graph $G_1 - G_0^1$ (here "$-$" is used to denote a graph subtraction), are to be added, together with their corresponding instances, to G_2 as direct successors of the nodes which their parent nodes in G_1 have been mapped to. Figure 4.20b presents a rough sketch of this method.

[18]Precisely, the task is solved in cases 2.2. and 2.3., for the ontologies are of different structures and different, although overlapping extension sets.



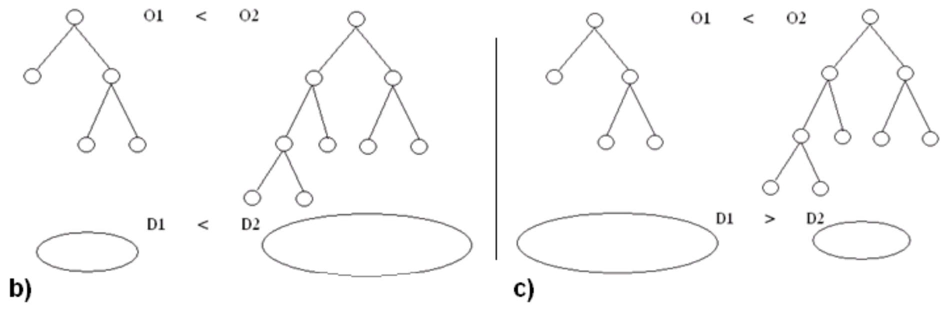


Figure 4.19: FEOM Case 2. One of both taxonomies is a sub-tree of the other.

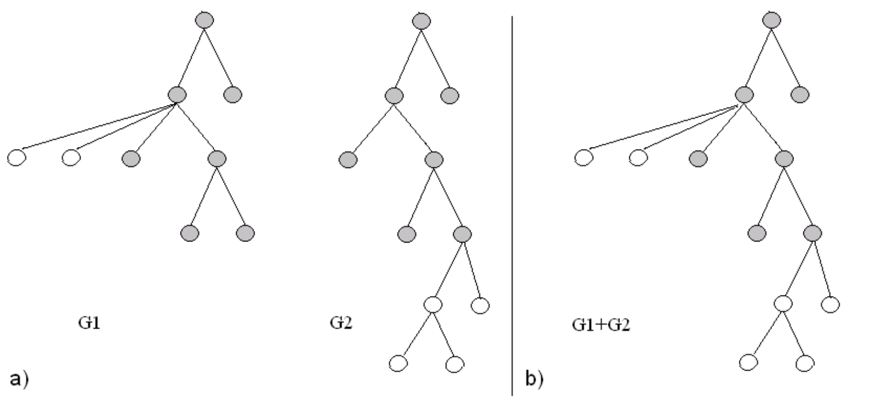


Figure 4.20: FEOM Case 3. a) One of the ontology trees is broader and the other one is deeper than their *mcs* (represented with shaded nodes). b) Merging O_1 and O_2.

Partial extensional overlap matching (PEOM)

The partial extensional overlap mapping procedure, which we are about to describe, rules out the cases when $0 < r_\Delta < 1$.

Recall the notation $D_0 = D_1 \cap D_2$. Let $G^1_{D_0}$ denote that part of the tree G_1 which contains only these concept nodes whose extension is some subset of D_0. Similarly, let $G^2_{D_0}$ denote the part of the tree G_2 which contains the concept nodes corresponding to subsets of the extensional subset D_0. In Figure 4.21a and 4.21b (above) the trees $G^1_{D_0}$ and $G^2_{D_0}$ are depicted with shaded nodes.

The main heuristics is that since both ontologies share a certain extensional overlap, so do they share a certain intensional one and the matching concepts of both ontologies are found in the trees $G^1_{D_0}$ and $G^2_{D_0}$. The task is, therefore, to find the precise correspondences between the nodes of these trees. This is a problem which has been already solved in Case 1.1. and Case 2.1. from the FEOM matching procedure (ontologies "built" on the same document sets).

The question which remains to be answered is whether a global merge of both ontologies is needed or just a partial mapping of the concepts in $G^1_{D_0}$ and $G^2_{D_0}$ would suffice. Clearly, apart from practical and application-related considerations, from a theoretical viewpoint the answer of this question depends on the size of the intersection D_0 relative to the set sizes. It would not be a worthwhile effort to merge two ontologies, if only very few of their instances are found to be similar. Therefore, we distinguish between two main further cases according to the size of the intersection D_0. We introduce a constant $c_{r_\Delta} \in (0, 1)$ (set by the user) according to the value of which we will either globally match and merge both ontologies, or match them partially by indicating which parts of the ontologies are correspondent in a set of concept similarity assertions. For a fixed value of c_{r_Δ} we sketch the following cases.

(A) $0 < r_\Delta < c_{r_\Delta}$. The size of the extensional overlap of both ontologies is judged by the user to be too small for a global ontology matching to take place. The procedure is terminated by matching the ontologies $O^1_{D_0}$ to the ontology $O^2_{D_0}$ by the instance-based mapping M_{ib} on the sets of concepts $C^1_{D_0}$ and $C^2_{D_0}$, where $O^1_{D_0}$ and $O^2_{D_0}$ are the ontologies corresponding to the graphs $G^1_{D_0}$ and $G^2_{D_0}$, respectively. The procedure yields a set of concept correspondences.

(B) $c_{r_\Delta} < r_\Delta < 1$. The size of the extensional overlap of both ontologies is judged by the user to be significant. The ontologies are to be globally matched in the following manner.

- Match the ontologies $O^1_{D_0}$ and $O^2_{D_0}$ by the instance-based mapping M_{ib} performed on the concept sets $C^1_{D_0}$ and $C^2_{D_0}$.
- Let $G^1_{minus} = G_1 - G^1_{D_0}$ denote the remainder of the graph G_1 after subtracting the tree $G^1_{D_0}$ from it. Concatenate the tree G^1_{minus} to the tree G_2 in such a manner that in every level of G_2 are found nodes which are also found on the same level in the tree G_1.
- Form an ontology graph which conforms with the hierarchical ontology definition 39. That means that if after the concatenation there exist two concept nodes which do not have a supremum, introduce a top concept to play that role (Figure 4.21a, below).

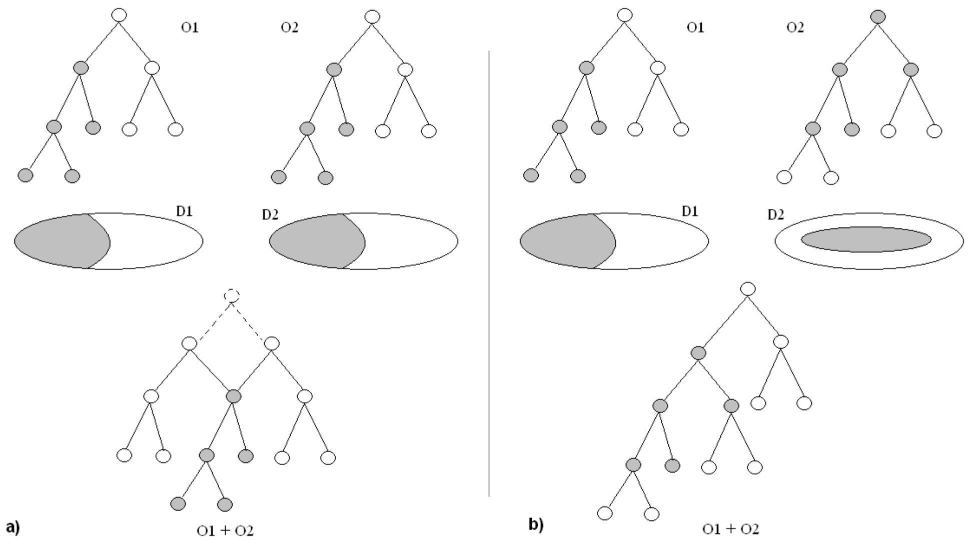


Figure 4.21: The PEOM ontology mapping.

Remark 46 *In spite of this being the intuitively more plausible case, there is no ground to believe that the graphs $G^1_{D_0}$ and $G^2_{D_0}$ are connected. However, since they are sub-graphs of trees, the missing edges between their nodes can be reconstructed with respect to the hierarchical structures of G_1 and G_2.*

4.6 Summary: Combined Matching Approaches

The current chapter contains the basic theoretical contribution of the thesis, which can be summarized concisely in the following set of claims:

- The VC-dimension of SVMs can provide a reliable variable selection / variable scoring criterion in a classification task.
- A concept in an ontology can be represented as a list of variables scored with respect to their discriminative power, or the role they play for separating the instances in the ontology into those that belong to the concept in question and those that do not. In view of that finding, a concept can be represented by just a small set of variables – those, which are most characteristic for this concept and distinguish it best from the rest of the concepts within its ontology.
- A measure of semantic proximity of two concepts, taken from two different ontologies, can be introduced based on the sets of characteristic variables corresponding to each of the concepts, or based on the whole lists of discriminative (scored) variables.
- An overall procedure for ontology matching can be built on the suggested above measure of concept similarity, by optimizing the search of similar pairs of concepts, using the structural relation of the concepts within

each ontology and recursively applying the similarity measure on sets of homogeneous classes.

- Combining graph representation of ontology with the described instance-based techniques and applying structural similarity measures together with the instance-based similarity helps yield assertions on the specificity, granularity and instantiation mismatches between two input ontologies. These provide guidelines for an ontology engineer through a process of ontology merging, when ontology merging is required for the needs of the concrete application.

- Structure-related ontology matching techniques are not sufficient and reliable indicator of the semantic similarity of two ontologies or parts of them and they need to be used in combination with approaches of other nature. Instance-based matching, on the other hand, can be improved significantly in terms of reducing time complexity and optimizing search by the help of structural accounts.

The findings listed above result in two separate combined (instance-based and structural) procedures for ontology matching: one relying heavily on extensional approaches (described in Section 4.4), and another taking into account to a greater degree the structural properties of the input ontologies (introduced in Section 4.5).

An empirical evaluation and discussion of the main results introduced above is presented in the following chapter of the thesis.

Chapter 5

Experimental Results and Discussion

The theoretical findings from the preceding chapter will be proved empirically by experiments on real-world data in the following couple of sections. For reasons stated in the introduction, as well as throughout the thesis, the emphasis of our experimental evaluation falls on the suggested instance-based techniques for concept mapping. Precisely, the chapter provides an entire evaluation of the work contained in Sections 4.3 and 4.4, as well as a partial evaluation of the procedure suggested in Section 4.5.

The chapter closes with a summary of the experimental results and a discussion attempting to situate our approaches in the general framework of models for similarity-based category formation, which have been discussed in detail in Section 3.4.4.

In much of the following experimental work, we have used data from the publicly available "20 Newsgroups" dataset [92], which is a collection of approximately 20,000 news articles, partitioned evenly across 20 different topics. Among other, it contains classes like "Religion", "Politics", "Autos", "Computers", "Sports and Recreation", etc. The structure of the topics is not entirely flat; some of the topics are sub-topics of others. For example the class "Computers" contains sub-classes, such as "Hardware-PC", "Hardware-Mac", "Graphics" and other.

We have set up several different experiments in which the listed classes have been used in order to mimic document populated ontology concepts or small taxonomies subject to similarity evaluation. The documents from the dataset are represented as TF/IDF vectors, except when explicitly stated otherwise (features were extracted by the help of the *RapidMiner* toolkit [104]).

5.1 Detecting Potential Concept Mappings with Discriminant Analysis

In the following two experiments, our goal has been to show that by the help of methods from descriptive statistics, such as Principal Components Analysis (PCA) and Discriminant Analysis (DA) we are able to extract from our dataset

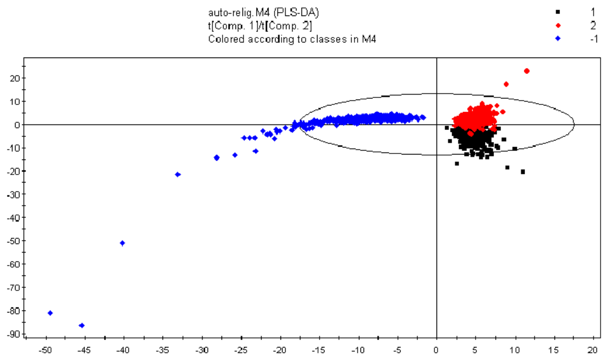


Figure 5.1: A DA plot of the population of the documents from three classes.

initial information concerning potentially similar concepts from two different ontologies. The results of the experiments are in support of the theoretical findings of Section 4.3.2.

5.1.1 Experiment 1

We started with the topics `Autos` and `Religion` and split the documents in `Autos` in two – `Autos1` and `Autos2`, producing sets of instances of two similar pseudo concepts. The documents in `Religion` were used as instances from a third (dissimilar) concept. Each class contains approximately 500 distinct documents, no two classes overlap.

We carried out a DA on the data set consisting of the three categories of documents, introduced so far. Figure 5.1 shows the results of the analysis on the first two discriminant axes. As we can see, the two autos classes appear very close to one another, sharing a big overlap and clearly separated from the religion class. The interpretation of that observation is that DA is not able to discriminate good between the two similar classes, but separates them well from the dissimilar class. DA shows to provide reliable indication of the similarity or dissimilarity of classes.

5.1.2 Experiment 2

We mimicked two ontologies by taking three classes from the 20 Newsgroups dataset – `Autos`, `Religion` and `Politics` and splitting the instances in each class in two even parts. In that manner we constructed the pseudo ontologies

$$O_1 = \{\texttt{Autos1}, \texttt{Religion1}, \texttt{Politics1}\},$$

$$O_2 = \{\texttt{Autos2}, \texttt{Religion2}, \texttt{Politics2}\}.$$

As in the previous experiment, each class contains distinct documents and classes do not intersect. We carried out a discriminant analysis and the results can be seen in Figure 5.2. On the plot, the positive labels correspond to the concepts of O1 and the negative ones – to the concepts in O2. In conformity with our claims and with the semantics of the classes used in the study, the six

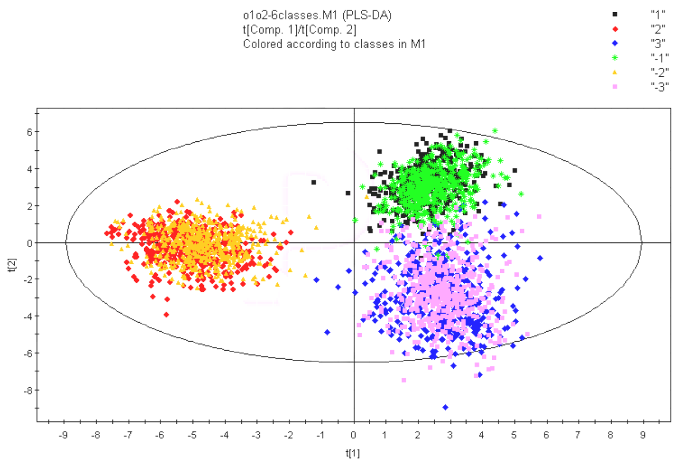


Figure 5.2: A DA plot of 6 pair-wise similar classes.

classes appeared to form three clear groups, where `Autos1` from O1 overlaps with `Autos2` from O2, `Religion1` overlaps with `Religion2` and `Politics1` overlaps with `Politics2`.

5.1.3 Experiment 3

We mimicked two ontologies, each containing four concepts, defined as it follows (the abbreviation “HW” stands for “Hardware”):

$$O_1 = \{\texttt{HW:PC}, \texttt{HW:Mac}, \texttt{Religion1}, \texttt{Politics1}\}$$

$$O_2 = \{\texttt{HW:Mixed}, \texttt{Autos}, \texttt{Religion2}, \texttt{Politics2}\}.$$

We have chosen the concepts and the documents for our task in such a manner that there are pairs of concepts which are clearly similar (e.g. `Religion1` and `Religion2`) and pairs of concepts which are clearly dissimilar (e.g. `Religion1` and `Autos`). In addition, there is one concept from O_2 which is in a way the union of two concepts of O_1 (the concepts `HW:Mixed` and the concepts `HW:PC` (containing documents about PC hardware) and `HW:Mac` (containing documents about Mac hardware)). Finally, as before, each of the classes contains approximately 500 distinct documents on the corresponding topic, non of the classes contains documents that are contained in another class.

We have carried out a discriminant analysis with the objective to detect potential concept mappings by plotting our instances onto a lower dimensional DA-space. As a first step, we observed the projection of the classes of O_1. In Figure 5.3, we see that the concepts `HW:PC` and `HW:Mac` overlap and are well separated from `Religion1` and `Politics1`, which, on their turn are well separated from each other. Figure 5.4 shows the plot of all eight concepts of O_1 and O_2. The classes are separated and regrouped according to their extensions. There are clearly four groups to be seen – one containing the instances of `HW:PC`, `HW:Mac` and `HW:Mixed`, one containing the instances of the classes `Religion1` and `Religion2`, one containing the instances of `Politics1` and `Politics2` and

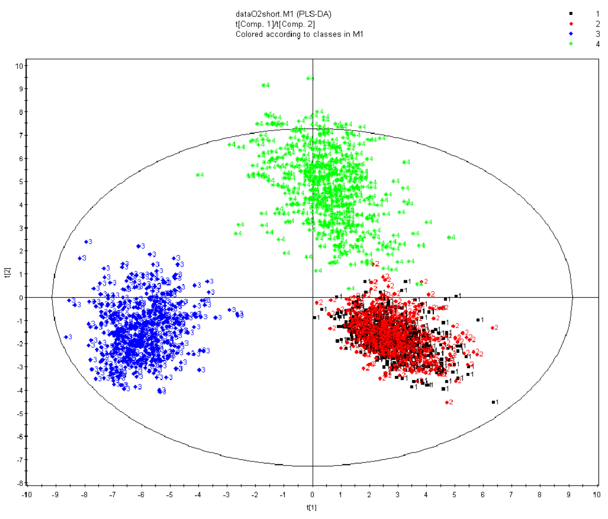


Figure 5.3: A DA plot of 4 classes of one ontology.

one containing the instances of the only class without a match, `Autos`. (On the plots, the classes are labeled by 1, 2, 3 and 4 for the classes in O_1 (in the respective order) and -1, -2, -3 and -4 for the classes in O_2 (in the respective order).)

We note that for this method to work properly and in order to be able to draw the correct conclusions, we need to have predefined labels of all our instances in each of the two respective ontologies.

5.2 Concept Similarity based on Variable Selection

Below, we present the results of a couple of experiments showing the viability of our finding that the similarity of two concepts can be measured by comparing the most characteristic for each of the concepts variables (Section 4.3.3). The results will be used to evaluate the instance-based concept similarity measure k-TF and compare it to Pearson's, Spearman's and Kendall's parameter-free coefficients (see Section 4.3.3 for definitions). The k-TF measure, we recall, is given by

$$sim(A,B) = \frac{|L_k^A \cap L_k^B|}{k} \tag{5.1}$$

for two concepts A and B taken from two different ontologies, where L_k^A denotes the set of the k most significant variables for class A. The empirical findings in this section equally provide support to the proposal of using an SVM-based variable selection criterion by observing variations of the VC dimension of the classifier, as proposed and discussed in Section 4.3.4.

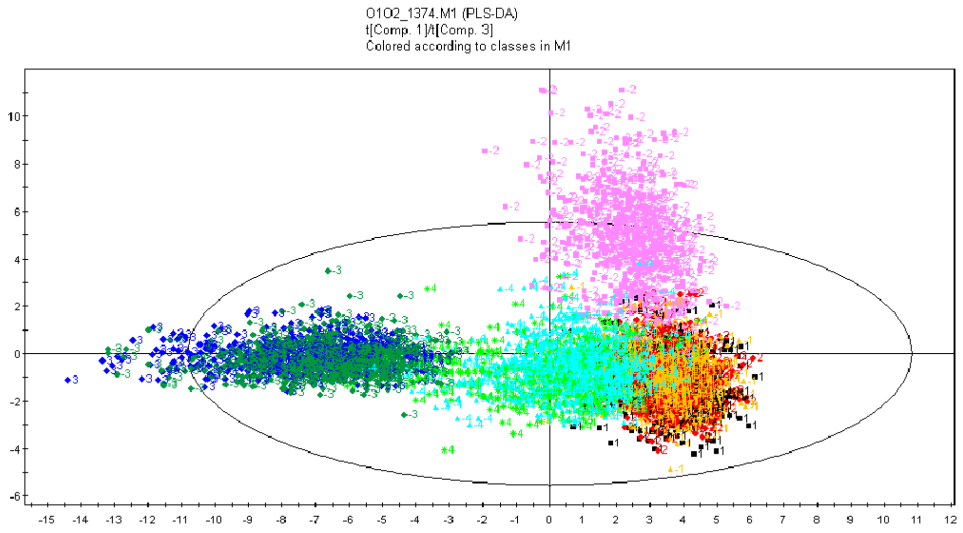


Figure 5.4: A DA plot of the classes of two ontologies.

5.2.1 Experiments 4: Concept Similarity via DA

In the current experimental setting, we have carried out discriminant analyses (DA) to obtain information about the variables importance with respect to a class. The study uses the same experimental setting as described in Section 5.1.2. We have again the following two collections of concepts:

$O_1 = \{\texttt{Autos1}, \texttt{Religion1}, \texttt{Politics1}\}$

$O_2 = \{\texttt{Autos2}, \texttt{Religion2}, \texttt{Politics2}\}$.

Let us recall our main argumentation: *for separating similar classes we need similar attributes, while for separating dissimilar classes we need dissimilar ones.* We evaluated the similarity of concepts `Autos1` and `Autos2` and the dissimilarity of concepts `Autos1` and `Politics2` and `Autos1` and `Religion2` by selecting the respective characteristic variables for each concept and applying the measure 5.1. To that end, we carried out the four following discriminant analyses:

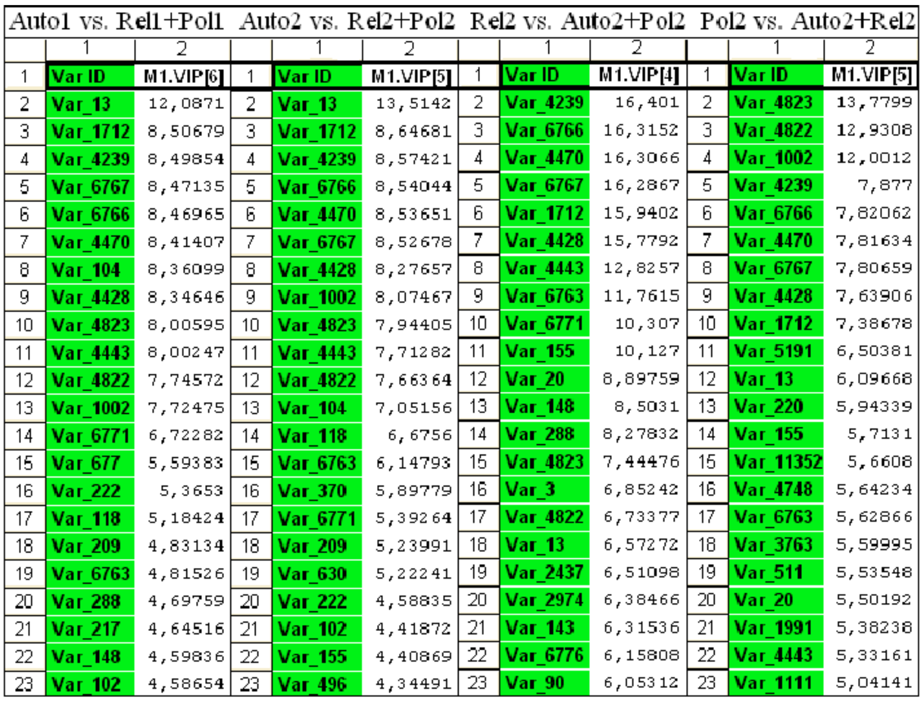
(DA1) `Autos1` vs. (`Religion1` + `Politics1`) – find the important variables that separate `Autos1` from all other concepts in O1;

(DA2) `Autos2` vs. (`Religion2` + `Politics1`) – find the important variables that separate `Autos2` from all other concepts in O2;

(DA3) `Religion2` vs. (`Autos2` + `Politics2`) – find the important variables that separate `Religion2` (a dissimilar concept) from all other concepts in O2;

(DA4) `Politics2` vs. (`Autos2` + `Religion2`) – find the important variables that separate `Politics2` (a dissimilar concept) from all other concepts in O2;

Figure 5.5 shows the lists of the top 23 most important variables for the class separation in the 4 different DA analyses. (VIP stands for a score coefficient calculated on the basis of the contribution of a single variable to the construction of the discriminant axes.) The results show that the lists of the important variables separating the classes in analyses (DA1) and (DA2) are very similar,



Auto1 vs. Rel1+Pol1			Auto2 vs. Rel2+Pol2			Rel2 vs. Auto2+Pol2			Pol2 vs. Auto2+Rel2		
	1	2		1	2		1	2		1	2
1	Var ID	M1.VIP[6]	1	Var ID	M1.VIP[5]	1	Var ID	M1.VIP[4]	1	Var ID	M1.VIP[5]
2	Var_13	12,0871	2	Var_13	13,5142	2	Var_4239	16,401	2	Var_4823	13,7799
3	Var_1712	8,50679	3	Var_1712	8,64681	3	Var_6766	16,3152	3	Var_4822	12,9308
4	Var_4239	8,49854	4	Var_4239	8,57421	4	Var_4470	16,3066	4	Var_1002	12,0012
5	Var_6767	8,47135	5	Var_6766	8,54044	5	Var_6767	16,2867	5	Var_4239	7,877
6	Var_6766	8,46965	6	Var_4470	8,53651	6	Var_1712	15,9402	6	Var_6766	7,82062
7	Var_4470	8,41407	7	Var_6767	8,52678	7	Var_4428	15,7792	7	Var_4470	7,81634
8	Var_104	8,36099	8	Var_4428	8,27657	8	Var_4443	12,8257	8	Var_6767	7,80659
9	Var_4428	8,34646	9	Var_1002	8,07467	9	Var_6763	11,7615	9	Var_4428	7,63906
10	Var_4823	8,00595	10	Var_4823	7,94405	10	Var_6771	10,307	10	Var_1712	7,38678
11	Var_4443	8,00247	11	Var_4443	7,71282	11	Var_155	10,127	11	Var_5191	6,50381
12	Var_4822	7,74572	12	Var_4822	7,66364	12	Var_20	8,89759	12	Var_13	6,09668
13	Var_1002	7,72475	13	Var_104	7,05156	13	Var_148	8,5031	13	Var_220	5,94339
14	Var_6771	6,72282	14	Var_118	6,6756	14	Var_288	8,27832	14	Var_155	5,7131
15	Var_677	5,59383	15	Var_6763	6,14793	15	Var_4823	7,44476	15	Var_11352	5,6608
16	Var_222	5,3653	16	Var_370	5,89779	16	Var_3	6,85242	16	Var_4748	5,64234
17	Var_118	5,18424	17	Var_6771	5,39264	17	Var_4822	6,73377	17	Var_6763	5,62866
18	Var_209	4,83134	18	Var_209	5,23991	18	Var_13	6,57272	18	Var_3763	5,59995
19	Var_6763	4,81526	19	Var_630	5,22241	19	Var_2437	6,51098	19	Var_511	5,53548
20	Var_288	4,69759	20	Var_222	4,58835	20	Var_2974	6,38466	20	Var_20	5,50192
21	Var_217	4,64516	21	Var_102	4,41872	21	Var_143	6,31536	21	Var_1991	5,38238
22	Var_148	4,59836	22	Var_155	4,40869	22	Var_6776	6,15808	22	Var_4443	5,33161
23	Var_102	4,58654	23	Var_496	4,34491	23	Var_90	6,05312	23	Var_1111	5,04141

Figure 5.5: Lists of characteristic variables in 4 DA analyses.

almost identical, whereas the variables separating the dissimilar concepts in analyses (DA3) and (DA4) differ from the lists obtained in the first two analysis. (We note that they do share a small overlap, for the concepts are not totally dissimilar, but rather.) By applying our variable-selection-based measure of similarity, we conclude that the concept `Autos1` from O1 is similar to the concept `Autos2` from O2 and dissimilar to the concepts `Religion2` and `Politics2` from O2 which is in line with the semantics of the selected classes.

5.2.2 Experiments 5: Concept Similarity via DF, MI, Chi^2

In the next set of experiments, we used several standard feature selection techniques, often applied for text categorization, to select the sets of characteristic variables of the concepts. Document Frequency Thresholding (DF), Mutual Information (MI) and Chi^2 statistics (Chi^2) are introduced in Section 2.4 of the second chapter of the thesis.

We took again the ontologies from Section 5.1.3:

$O_1 = \{\texttt{HW:PC}, \texttt{HW:Mac}, \texttt{Religion1}, \texttt{Politics1}\}$,

$O_2 = \{\texttt{HW:Mixed}, \texttt{Autos}, \texttt{Religion2}, \texttt{Politics2}\}$.

The task is to evaluate the pairwise similarity of the concepts from O_1 and O_2 by the help of the similarity measure 5.1.

Table 5.1: Heuristics similarity matrix

Concept Names	HW:Mixed	Autos	Religion2	Politics2
HW:PC	**sim** $> \mathbf{l}$	$sim < l$	$sim < l$	$sim < l$
HW:Mac	**sim** $> \mathbf{l}$	$sim < l$	$sim < l$	$sim < l$
Religion1	$sim < l$	$sim < l$	**sim** $> \mathbf{l}$	$sim < l$
Politics1	$sim < l$	$sim < l$	$sim < l$	**sim** $> \mathbf{l}$

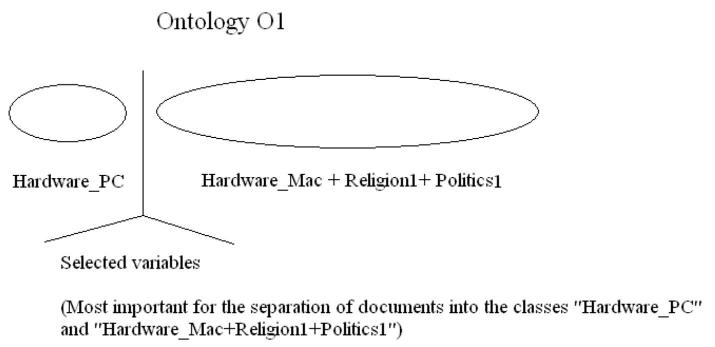

Ontology O1
Hardware_PC
Hardware_Mac + Religion1+ Politics1
Selected variables
(Most important for the separation of documents into the classes "Hardware_PC" and "Hardware_Mac+Religion1+Politics1")

Figure 5.6: Variable selection for a concept "Hardware PC" in ontology O1.

We have drawn a so called heuristics similarity matrix, represented on Table 5.1, containing the expected pairwise concept similarity values, which will serve as an evaluation of the test similarity values to be obtained in the experimental studies. In the table, *sim* is a number between 0 and 1 corresponding to the similarity of the respective pair of concepts. A similarity value greater than a fixed threshold, l, indicates expected semantic proximity of the concepts; otherwise does a similarity value smaller than l.

Setting the similarity threshold l. The values of l vary with respect to the variable selection technique applied. As we shall see, using Mutual Information requires setting l quite small, while working with Chi square statistics or document frequency thresholding demands a higher value of the parameter. Therefore, we suggest that l should be set (independently for every variable selection method) to the lowest value computed over all pairs of similar concepts taken from a training document dataset (on which we have heuristics about the semantic closeness of the concepts) before applying the similarity measure on unseen documents.

The task consists in the following steps.

- Carry out a variable selection procedure for four different cases (one for each concept) in each of the two ontologies (all in all that makes 8 different variable selection tasks). The variable selection procedure used is up to

Table 5.2: Performance using Mutual Information

Concept Names	HW:Mixed	Autos	Religion2	Politics2
HW:PC	**0,033**	0	0	0
HW:Mac	**0,067**	0	0	0
Religion1	0	0	**0,3**	0
Politics1	0	0	0	**0,3**

Table 5.3: Performance using Chi^2 with 30 features

Concept Names	HW:Mixed	Autos	Religion2	Politics2
HW:PC	**0,700**	0,433	0,400	0,367
HW:Mac	**0,500**	0,467	0,433	0,367
Religion1	0,400	0,400	**0,700**	0,300
Politics1	0,333	0,367	0,333	**0,633**

the users choice; experiments have been made with three different selection techniques (see below).

- Select 8 sets of variables which are most characteristic for each of the 8 considered concepts. An illustration of that step is given in Figure 5.6 for the concept `HW:PC` in Ontology 1.
- Calculate the pairwise similarities of each pair of concepts from O_1 and O_2, by using the measure of similarity (5.1), where A is a concept from O_1 (`HW:PC`, `HW:Mac`, `Religion1` or `Politics1`) and B is a concept from O_2 (`HW:Mixed`, `Autos`, `Religion2` or `Politics2`).
- Calculate a 4×4 similarity matrix, where each element of the matrix is the similarity between two concepts (each belonging to a distinct ontology).
- Finally, compare the values of the experimentally obtained similarity matrix with the values of the heuristics similarity matrix (Table 5.1).

We present the results that we have obtained by the help of the standard variable selection techniques Mutual Information, Chi-square and Document Frequency Thresholding in the similarity matrices presented, correspondingly, on tables 5.2, 5.3, 5.4, 5.5.

Table 5.4: Performance using Chi^2 with 200 features

Concept Names	HW:Mixed	Autos	Religion2	Politics2
HW:PC	**0,500**	0,230	0,275	0,235
HW:Mac	**0,475**	0,200	0,315	0,275
Religion1	0,230	0,175	**0,545**	0,145
Politics1	0,240	0,165	0,125	**0,590**

Table 5.5: Performance using Document Frequency Thresholding

Concept Names	HW:Mixed	Autos	Religion2	Politics2
HW:PC	**0,722**	0,556	0,440	0,485
HW:Mac	**0,726**	0,545	0,437	0,489
Religion1	0,431	0,444	**0,753**	0,541
Politics1	0,479	0,526	0,550	**0,772**

Setting the parameter k. We have varied the parameter k (the number of selected features on the basis of which the similarity of the classes is computed). Tables 5.3 and 5.4 show the similarity matrices calculated for the Chi-square selection procedure by using 30 features in the first case and 200 in the second. As we would have expected, the similarity values for the similar concepts decrease by adding more features in the computation (more "noisy" features are present in a bigger set). However, so do the similarity values for the dissimilar concepts. This is due to the fact that the method takes into account both frequent and rare terms in the selection of characteristic features.

The observed variations show that both values of k are appropriate, for the interval between the similarity values of similar and dissimilar concepts has not shrunk, but has rather shifted. Of course, we note that, despite the fact that the difference between the two values of k is quite big, they are both still relatively small numbers compared to the overall number of features (about 12 000). The similarity measure that we work with will show no significant results if k is set too large relative to the overall number of features (all concepts will show to be similar).

5.2.3 Experiments 6: Concept Similarity via VC-dimension-based Variable Selection

This section presents an evaluation of the theoretical finding that the VC dimension of support vector machines can provide a criterion for identifying a set of characteristic variables for a concept, which in turn can be used for measuring the similarity of two concepts. The variable selection criterion was introduced theoretically in Section 4.3.4. For its empirical evaluation for the task of ontology matching, we will use again the k-TF measure of similarity (5.1), as well as, alternatively, Spearman's coefficient (calculated on the ranks), Pearson's coefficient (calculated on the scores) and Kendall's measure of correlation (see Section 4.3.3 and [29]).

Table 5.6: VCdim-based similarity: k-TF measure (left) and Pearson's coefficient (right)

Concept Names	HW	Religion	Politics	HW	Religion	Politics
HW	**1**	0,20	0,35	**1**	0,24	0,87
Religion	–	**1**	0,10	–	**1**	0,21
Politics	–	–	**1**	–	–	**1**

Table 5.7: VCdim-based similarity: Spearman's coefficient (left) Kendall's coefficient (right)

Concept Names	HW	Religion	Politics	HW	Religion	Politics
HW	**1**	0,37	0,62	**1**	0,25	0,45
Religion	–	**1**	0,34	–	**1**	0,24
Politics	–	–	**1**	–	–	**1**

In the first preliminary experiment we started with a small set of three classes

$$O_1 = \{\texttt{HW}, \texttt{Religion1}, \texttt{Politics1}\},$$

duplicated it and tested how the k-TF measure of similarity and the three correlation coefficients will perform to evaluate the pairwise similarity of the concepts in O_1 and its copy (or in other words, measure the pair-wise similarity of the concepts within a single ontology). The results are shown on the similarity matrices on tables 5.6 and 5.7. (The redundant values are omitted, k was set to 20 for the k-TF measure.) We note that in all four similarity matrices, `Religion`

Table 5.8: VCdim on non-intersecting classes with k-TF 100 Features (left); optimized (right)

Concept Names	Autos	Religion2	Politics2	Autos	Religion2	Politics2
HW	0,40	0,37	0,41	0,40	0,15	0,35
Religion1	0,33	**0,52**	0,40	0,40	**0,60**	0,50
Politics1	0,44	0,41	**0,57**	0,50	0,40	**0,65**

Table 5.9: VCdim on non-intersecting classes with Pearson's coefficient (left); optimized (right)

Concept Names	Autos	Religion2	Politics2	Autos	Religion2	Politics2
HW	0,29	0,02	0,31	0,35	0,06	0,35
Religion1	0,17	**0,55**	0,15	0,29	**0,92**	0,17
Politics1	0,5	0,18	**0,68**	0,6	0,23	**0,75**

appears to be more or less equally dissimilar to `Hardware` and to `Politics`; `Politics`, on the other hand, shares more commonalities with `Hardware` than with the two other classes. These appear to be intuitively plausible findings. In addition, they conform to the fact that (5.1) is not a distance function since it does not fulfill the triangle inequality.

In the next of the VC-dimension-related experiments that we carried out, we have tested the similarity of the members of the following two collections of pairwise disjoint sets of documents (the indexes 1 and 2 after the identical class names indicate that they contain different sets of documents):

$O_1 = \{\texttt{HW}, \texttt{Religion1}, \texttt{Politics1}\}$

$O_2 = \{\texttt{Autos}, \texttt{Religion2}, \texttt{Politics2}\}$.

The results by using the k-TF measure with $k = 100$ are shown on table 5.8 (left). So far the similarity values are not significantly better or worse than the ones observed by measuring the similarity with a standard text-classification feature selection technique. However, the new variable selection technique performed much better in identifying the first most important variable for each of the classes. In the many trials we had, this variable was always identical for the expectedly similar classes and never for any other pair of concepts from the two sets. This effect was confirmed by setting the number of selected features k to 10 and doing the test again – the similarity coefficients for the Religion and

Politics classes rose, and dropped proportionally for the other pairs of concepts. After some optimization of the parameters of the support vector machines we obtained even more fine-grained results (Table 5.8 (right)).

Encouraged by the results achieved by using Pearson's correlation described in the beginning, we tested the measure on disjoint sets of instances. The results are shown on the tables 5.9 (right (without optimization) and left (with optimization)). We see that the method shows to perform well in distinguishing clearly between similar and dissimilar classes. (We notice that the class `Politics1` is judged to be quite similar to the class `Autos`, but nevertheless, its highest similarity coefficient is for the class `Politics2`.)

5.2.4 Experiments 7: Comparison of the Performance of the Similarity Coefficients (*k*-TF vs. Correlation Measures)

So far, the performance of the different similarity coefficients shows to be competitive with an advantage for the k-TF measure and the Pearson's correlation coefficient. In order to support or disprove this finding we tested the performance of the four similarity coefficients in a series of experiments on larger sets of concepts containing non-intersecting sets of documents. We recall the reader that using some of these measures does not influence the concept of similarity that we consider, described by the heuristics: *similar concepts are characterized by similar sets of characteristic variables; otherwise are dissimilar ones.* The selection or the ranking of variables is always with respect to the variations of the VC dimension of the SVMs (see equation (4.20)). The eventual computation of the concept similarity coefficients can be done in different manners, four of them being the k-TF measure, Pearson's, Spearman's and Kendall's measures of correlation.

We measured the performance of the four similarity measures in terms of precision in judgments and average similarity coefficient found for similar and dissimilar concepts; see Table 5.10. The precision in judgments is computed with respect to the lowest value calculated for a pair of similar concepts on the data. That means that we set the similarity threshold l to the lowest similarity value computed on all pairs of similar concepts. In that, naturally, the precision on similar concepts is always 100% and the indicator of the overall precision of performance is the number of errors in judging dissimilar ones. The average similarity values for similar and dissimilar concepts (abbreviated in the table as *avg. sim. coeff.*) are meant to indicate how reliable the respective measure is (the larger the interval between the average value for similar and the average value for dissimilar concepts, the more reliable the measure).

The result of our tests have shown that, among the three tested similarity measures, Pearson performs best, achieving the same precision as the k-TF measure, but showing to have a significantly larger interval between the average similarity and dissimilarity judgments – an indicator for its reliability. Kendall's coefficient demonstrated worst precision and tightest border between similarity and dissimilarity judgments. These results provided support to our finding that the k-TF measure and the Pearson's correlation coefficient provide reliable similarity judgments by using the SVM-based criterion.

In the next section, we will evaluate the procedure for ontology matching

Table 5.10: Performance of the similarity measures

	Similar concepts Avg. sim. coeff.	Dissimilar concepts Avg. sim. coeff.	Precision
100-TF	0.55	0.40	92%
10-TF	0.62	0.38	92%
Pearson's r	**0.87**	**0.28**	92%
Spearman's rho	0.36	0.19	78%
Kendall's tau	0.25	0.10	84%

based on the variable selection similarity criterion, by using the SVM-based techniques. To compute the similarity coefficients we have used Pearson's correlation coefficient and the k-TF measure. The presented experiments only feature the results obtained by the help of Pearson's measure.

5.2.5 Experiments 8: Comparison of the Performance of the Variable Selection Techniques

Using the same setting and criteria as for the experiments described in the preceding section, we compared the performance of the introduced variable selection techniques: the standard MI, DF and Chi^2 Statistics against the novel VC-dimension-based method (VC-dim). The results are shown in Table 5.11. Although all tested techniques are entirely accurate in judging concept similarity (understood as a binary operation: two concepts are asserted similar or dissimilar without accounting for degrees of similarity), the novel SVM-based method shows to be a more reliable technique, since the interval between the value assigned to dissimilar concepts and the value assigned to similar ones is much larger than the same interval for the rest of the tested techniques.

Table 5.11: Performance of the variable selection techniques

	Similar concepts Avg. sim. coeff.	Dissimilar concepts Avg. sim. coeff.	Precision
MI	0.175	0.0	100%
DF	0.743	0.495	100%
Chi^2	0.527	0.234	100%
VC-dim	**0.905**	**0.23**	100%

5.3 Evaluation of the Overall Variable Selection-based Ontology Matching Procedure

In Section 4.4 we have observed that the proposed instance-based measure of similarity should not be applied directly on the two sets of concepts of the two source ontologies. Instead, we suggested a procedure, which takes into account

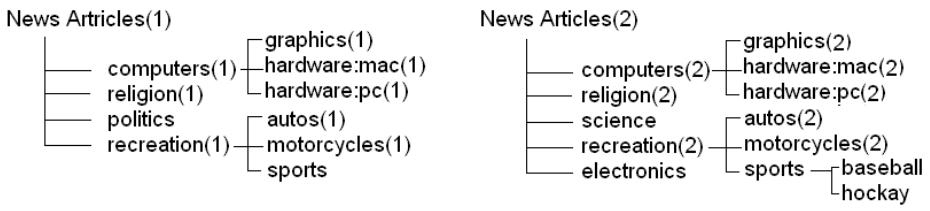


Figure 5.7: Two News-Ontologies.

the structure of the ontologies and maps sub-sets of concepts in an iterative manner descending down the hierarchies' levels. In that way, in addition to including the structure into the mapping process and narrowing down the set of concept pairs to be checked for similarity, we avoid an undesired effect of matching a concept from one ontology to the concept's match subsumer in another ontology. We have shown that this can be problematic for a variable selection procedure based on binary classification, such as the VC-dimension criterion for SVMs that we suggest.

5.3.1 Experiments 9: Matching the News-Ontologies

In the current set of experiments, we will match two small ontologies that we have created manually in order to show the viability of the suggested overall variable selection-based mapping procedure discussed in Section 4.4. Our source ontologies, which we call the News-Ontologies are shown in Figure 5.7.

Before testing the suggested VC-dimension-based similarity criterion, we will start by evaluating the performance of two standard concept similarity measures: the cosine similarity for class prototypes, $sim_{proto}(A, B)$ which underlies the Caiman tool and the Jaccard similarities $sim_{jaccard}$ which is in the core of the Glue mapper. For these methods, we will only present the results for the top-level categories of the news ontologies. For the similarity measure based on the VC-dimension, we additionally evaluate the application of the entire matching procedure.

In the test ontologies, leaf nodes were assigned about 500 documents on the related topic. Parent nodes were assigned the union of the documents assigned to their children plus some additional documents on their topic, in order to account for documents annotated directly by the parent concept. Each top-level has between 1900 and 2500 instances. Each instance is described by 329 features corresponding to a selection of the words occurring in the original news articles. Note that we reduced the initial number of terms substantially by removing stop words, applying stemming, and deleting high and low frequency words. The results, as so far, are presented in the form of similarity matrices, where underlined entries mark the pairs of concepts that are supposed to be mapped onto each other, like *Computers(1)* and *Computers(2)*. We have decided to use a natural value of the threshold l of 0.5. Numbers in italics mark values above this threshold. If we establish a mapping for a pair of classes that are not supposed to be mapped, the mapping might be considered as incorrect.

Table 5.12: Matching the news ontologies: sim_{proto}

Concept Names	Comp(2)	Religion(2)	Science(2)	Recr(2)	Electr(2)
Computers(1)	*0.924*	0.188	0.457	*0.514*	*0.6*
Religion(1)	0.175	*0.972*	0.154	0.201	0.191
Politics(1)	0.414	0.246	0.441	*0.522*	0.47
Recreation(1)	*0.539*	0.218	0.369	*0.843*	*0.586*

Table 5.13: Matching the news ontologies: $sim_{jaccard}$

Concept Names	Comp(2)	Religion(2)	Science(2)	Recr(2)	Electr(2)
Computers(1)	*0.597*	0.03	0.14	0.02	0.16
Religion(1)	0.0	*0.99*	0.0	0.0	0.0
Politics(1)	0.008	0.003	0.12	0.02	0.005
Recreation(1)	0.02	0.04	0.007	*0.86*	0.06

5.3.2 Prototype Method

The prototype method based on the Caiman idea simply consists in first computing the class (concept) means in the usual manner, and then applying the instance based similarity measure introduced in equation (3.2) to it. Since all vectors have non-negative feature values, the similarity lies always between 0 and 1.

The similarities for the concepts in the two news ontologies can be found in Table 5.12. It can be determined that the prototype approach is able to detect similar pairs of concepts. However, it fails to properly detect dissimilar pairs, provided a natural threshold of 0.5 (see e.g. the pair *Politics(1)/Recreation(2)*). This makes it difficult to use this measure for the recursive matching procedure, which relies on being able to make such decisions. We assume that the relatively high similarity values for dissimilar concept pairs result from the averaging process that is used for computing the means: compared to the more "extremal" vectors in each class, the means tend to be more similar.

5.3.3 Jaccard Similarity-Based Method

For completeness of our proof of concept, we tested the similarities of the first levels of the two news ontologies by the help of one of the most popular measures found in the extensional OM literature, the Jaccard similarity which is in the core of the Glue matcher, given in equation (3.3) in Section 3.5.2. The quantities $P(A \cap B)$ and $P(A \cup B)$ were estimated by learning an SVM on the instances of O_1 and testing it on the instances of O_2 and vice versa, as explained in [39]. The concept similarities are found in Table 5.13. The results by using the *corrected* Jaccard coefficient (equation (3.4)), suggested by [72], were similar to the ones presented here. Below, these results will be compared with the results achieved with the proposed approaches.

5.3.4 Similarity Based on VC Dimension (VC-VS)

We present an evaluation of the instance-based matching procedure suggested in Section 4.4 as well as of the finding that the VC dimension of SVMs can

Table 5.14: Matching the news ontologies by selecting variables with MI

Concept Names	Computers(2)	Religion(2)	Science(2)	Recr(2)	Electr(2)
Computers(1)	*0.548*	−0.405	0.119	0.146	0.464
Religion(1)	−0.242	*0.659*	−0.105	−0.201	−0.271
Politics(1)	−0.105	0.019	0.276	0.138	−0.015
Recreation(1)	0.318	−0.301	0.001	***0.528***	0.356

Table 5.15: Matching the news-ontologies by selecting variables with VC-SVM: first levels vs. mixed levels

Concept Names	Comp(2)	Religion(2)	Science(2)	Recr(2)	Elec(2)
Computers(1)	*0.78*	0.08	0.04	0.40	0.06
Religion(1)	0.07	*0.94*	0.02	0.10	0.01
Politics(1)	0.08	0.12	0.08	0.14	0.01
Recreation(1)	0.29	0.09	0.06	*0.60*	0.06
Graphics(1)	0.30	-0.01	-0.1	-0.01	-0.002
HW:PC(1)	0.18	-0.03	-0.01	-0.02	-0.03
HW:Mac(1)	0.03	-0.02	-0.01	-0.02	-0.09

provide a criterion for selecting variables in a classification task. As a measure of similarity we have used Pearson's measure of correlation (wherefrom the negative numbers). The measure indicates high similarity for positive values and low similarity for non-positive values. The results achieved by using the novel SVM-based variable selection technique (Table 5.15) proved to be better than those achieved by a standard point-wise mutual information-based criterion (Table 5.14). The results presented below come from using the former method.

After the root nodes of O_1 and O_2 have been matched, we proceeded to match the sets of their direct descendants. The results are shown in the upper similarity matrix on Table 5.15. Our similarity criterion identified successfully the three pairs of similar concepts (the Computers, the Religion, and the Recreation pairs). Following the matching procedure, as a second step we matched the sets of the descendants of the concepts that were found to be similar in the first step. The obtained similarity values for the children of the computer- and recreation-classes are shown in Table 5.16. Finally, in order to show the effect of not respecting the rule of hierarchical matching, we have matched the first level of O_2 with the descendants of the computer class in O_1. The obtained similarity values are shown in the lower matrix in Table 5.15: although the potentially similar classes are accorded higher similarity values, the similarity coefficients are much lower than in the previous cases and much closer to the values for the dissimilar classes.

The similarity values are computed on the sets of input variables which, in case of text, correspond to actual words. Thus, the most important words that discriminate between a pair of similar concepts and the rest of the pairs of concepts can be readily made available in contrast to related methods (e.g. Caiman or Glue). For example, our selection procedure found out that among the most important tokens that characterize the concept *Computers* are *comp*, *chip*, *graphic*, *card*, *devic*, *file*. In contrast, the features with highest scores for *Religion* were *christ*, *church*, *faith*, *bibl*, *jesu*, for *Politics* – *polit*, *govern*, *legal*,

Table 5.16: Similarities of the descendants of the computer- and the recreation-classes

Concept Names	Graphics(2)	HW-Mac(2)	HW-PC(2)	
Graphics(1)	*0.954*	-0.475	-0.219	
HW-Mac(1)	-0.266	*0.501*	-0.073	
HW-PC(1)	-0.577	0.304	*0.556*	
Concept Names	Autos(2)	Motorcycles(2)	Baseball	Hockey
Autos(1)	*0.978*	0.478	0.117	0.095
Motorcycles(1)	*0.560*	*0.989*	0.121	0.452
Sports	0.452	0.491	*0.754*	*0.698*

talk and for *Recreation* – *motorcycl*, *auto*, *speed* and *engin*. This information is useful to verify the quality and coherence of the matching results.

5.3.5 Matching the News-Ontologies: Discussion

In this matching task, three similarity criteria have been tested on two source ontologies populated with textual instances: one based on variable selection with VC-dimension (VC-VS), the Jaccard similarity used in the Glue tool and one, prototype method based on the Caiman system. Our results showed that the VC-VS method outperforms the other considered measures, yielding a clearcut difference between the similarity values obtained for pairs of similar and pairs of dissimilar concepts. The method shows to respond properly to a natural similarity threshold of 0.5 on a 0-1 scale. Although the results achieved with the Jaccard similarity are competitive, the VC-VS approach makes a step further in terms of similarity verification, since the discriminant features (words) for each class are readily made available.

5.4 Summary and Discussion

The experimental work that we have just presented intends to convince the reader in the practical value of a set of technical solutions to the problem of overcoming ontology heterogeneity, which have been introduced theoretically in the previous chapters of the thesis. The results provide support to our hypotheses that the suggested approaches perform efficiently and effectively. The basic outcome of our empirical tests can be summarized in the following points.

5.4.1 Summary and discussion of the experimental results

- Every instances-populated ontology concept can be described by the variables, which describe its instances, scored by their importance for separating the instances within an ontology into those that belong to the concept and those that do not;
- Similarity of concepts can be measured in an inexpensive way by comparing their corresponding sets of scored variables. To score the variables,

various methods can be applied, such as descriptive statistical techniques and variable selection;

- Variable selection techniques provide a more reliable ground for measuring concept similarity, compared to descriptive statistical methods. The latter are useful for extracting the most important initial information out of a large dataset (many instances belonging to many concepts), for getting the big picture. The former, in turn, yield correct results in a concept-to-concept mapping task;

- Among the tested variable selection techniques, the novel SVMs-based selection criterion performs better than the standard variable selection techniques for text categorization (MI, DF, Chi-square);

- Among the four tested measures of concept similarity, based on variable selection procedures, the Pearson's correlation coefficient showed to perform best, competing closely with the suggested k-TF similarity measure.

- In order to optimize the search of the potential concept mappings among all possible pairs of cross-ontology concepts, the structure of the two source ontologies should be taken into account. The suggested overall variable-selection-based procedure for ontology matching is an efficient and reliable matching tool on larger ontology datasets, performing series of similarity checks only on sets containing potentially similar concepts;

The main advantages of the proposed approaches compared to state-of-the-art methods have been outlined in the introduction of Section 4.3. Summing them up, we conform to our initial hypothesis that the efficiency of the suggested approaches is due to their following properties.

- The presented approaches do not rely on instance sets intersections and therefore can be applied on source ontologies populated with entirely different documents;

- The relevant variables, needed for the similarity computation, are determined independently for each ontology, and the matching itself is an inexpensive computation. Once the variables are scored, a similarity measure of any kind can be computed based on this scores;

- The assessment of the similarity of two concepts is entirely accomplished at the training-phase of the learning task;

- The discriminant features (words in the case of textual instances) for each class are readily made available;

- The concept similarity measurement is performed only on sets containing potentially similar concepts, which prevents us from having to evaluate m times n concept pairs for two ontologies (one with n, another with m concepts).

A shortcoming in terms of time complexity of our method, is that applying the VC-dimension SVM-based technique can be also fairly time demanding, because of the many learning tasks performed for every concept in order to score the variables with respect to their importance for this concept. In order to increase the time efficiency, we need to diminish the number of variables describing the documents in our sets, by using another method for dimensionality reduction in a preprocessing phase. Our results showed that using a variable selection technique of other kind (like, for instance Mutual Information) instead of the SVM-based approach will speed up the process, leading to an overall running time shorter than the time needed for computing the Jaccard coefficient or the VC-dimension-based similarity. However, the price for the saved time is the decreased quality of the similarity judgments.

In the following subsection, we will leave the sphere of technical details and continue the discussion by focusing on the relation of our approach to concept similarity to other (standard and recent) similarity models from cognitive psychology, artificial intelligence and computational linguistics.

5.4.2 Situating our approaches in the discussion about similarity and categorization

We have observed that the origins of the problem of ontology matching, to which this work is dedicated, should be sought in the nature of human concept formation and the ambiguities which stem out of the decentralized and distributed nature of this process. We pointed out that this process can be explained as the understanding of the relations that hold among a set of building components: words, concepts and real world objects (most generally these are percepts and actions (situations, properties, locations, times, etc.)) [99]. Griffiths *et al.* define these relations in the following way [60]. **Word-concept** relations [165, 59] concern the knowledge that a word, say, *dog*, refers to a concepts, "dog". The **concept-concept** relations [28, 83] are described by the knowledge of how different concepts are related, i.e. the knowledge of facts, such as, for instance, "dogs are animals", "animals have tails". Finally, the **concept-percept** and **concept-action** relations consist in the knowledge of what things look like, how can they be distinguished from one another (for example in what respects is a dog different from a cat), what actions can one perform on these things (how can one handle a dog or a cat). Additionally, the authors mention in a separate group the **word-word** relation [90], that is the knowledge how words are related between each other (i.e. the knowledge that the word *dog* is more likely to co-occur together with the word *tail*, than with the expression *mp3 player*).

Ambiguity in the human understanding of these relations and the way they are modeled can appear on any of the observed four levels. There is no generic model of semantic representation, which can account for all of these aspects simultaneously and unambiguously. In result, categorization and concept formation are heterogeneous as processes and phenomena among communities, societies or even individuals (remember the example of the Chinese taxonomy of the animal kingdom from the introduction [130]).

Similarity, as a concept which (as argued by many [67, 140]) is in the core of human categorization, can be introduced to restore the links between the different understandings of the entities discussed above and cope with heterogeneity of the semantic representation on each of the suggested levels (see Section 3.4.4 for a discussion of different computational models of semantic similarity).

In our study, we propose a formal framework for overcoming heterogeneity between ontologies populated with text instances. The notions of similarity that we define and apply on concepts and on structures are strict mathematical notions, which rely on a set of parameters. The actual choice of these parameters with respect to a particular matching task or method guarantees the needed and expected flexibility of the similarity judgments in view of the fact that ontology concepts are not strictly defined mathematical notions and the way they are modeled and organized together to form an ontology is strongly biased by human fuzziness. However, the judgments of semantic closeness remain an outcome of a similarity measure and similarity remains central to these judgments. In that respect, we conform to the tradition set by, among others, Rosch [129, 131] and pursued by Sloutsky [140] and Hampton [67] according to which similarity is in the core of human categorization – a proposition, which implies the plausibility of using similarity as an indication of concepts semantic proximity. In other words, if it is a claim that similar entities are put together in order to form the extension of a concept, similar concepts will have similar extensions.

In the preceding chapter, we have presented fully the theoretical grounds of several new approaches to ontology matching which, as it has been outlined, consist of two main similarity assessment components – (1) instance-based concept-to-concept similarity and (2) a measure of structural similarity of taxonomies. The current chapter has provided empirical evidence for the viability of the theoretical findings.

The proposed approach to concept similarity measurement is based on variable selection techniques from machine and text learning and shares commonalities with basic *spatial* and *feature models* of similarity. It is based on selecting a small set of distinctive features which characterize a concept and measuring the closeness of two concepts on the basis of their corresponding sets of distinctive features. The measures of similarity that we apply to that ends are not distance functions, therefore the model does not conform as a whole with the spatial models. However, we use, as a part of the instance-based mapping procedure methods for lower dimensional representation of observations where all metric conditions are fulfilled.

We note that findings in cognitive psychology support the idea that some features are more important for determining class membership than others (see experiments with HSE[1] patients by Taylor *et al.* [146]). A central claim is that for every category some features are more informative than others. These features that are shared by many concepts are indicative for the subsuming category of these concpets, but not helpful for identifying the objects belonging to these concepts.

The overall approaches to ontology matching that we presented combine structural information (how entities are related) with instance information (what is the statistical nature of a collection of observations taken from the real world). In view of that, the proposed methods share similarities with generative approaches [84] for they combine statistical learning with structure. More precisely, the proposed procedure to ontology matching, based on variable selection techniques, discussed in Section 4.4, is intuitively based on *hierarchical models*

[1]Herpes Simplex Encephalitis, a disease which has an effect of semantic impairment for certain categories (e.g. "living things") – patients would not assign all characteristic features to objects of a certain category, but only the category distinctive ones – sufficient to recognize the category, but not the objects.

of similarity of structured representations, whose basic claim is that the similarity of two entities is assessed on the basis of both similar / dissimilar features *and* the structure of these entities (relations of their components). The second suggested approach to ontology matching, described in Section 4.5, falls rather in the group of *transformational models*, for the dissimilarity of two ontologies is accounted for by the parts of their corresponding ontological graphs, which remain out of their maximal common subgraphs (*mcs*); the similarity of the two ontologies, under assumptions specified in Section 4.5, can be measured by the number of operations required to transform each of the ontologies into their *mcs*.

Chapter 6

Conclusion and Future Work

The final chapter of this thesis aims to summarize the main outcomes and possible future developments of our research efforts to provide an efficient solution to the problem of finding correspondences of the elements of two heterogeneous ontologies. In the next section, we state how and to what extent our initial goals have been achieved, summarizing the main ideas behind our approaches and outlining their most important advantages, compared to similar techniques (Section 6.1). The second section of the chapter closes the thesis by pointing out several theoretical and practical questions which have remained unsolved and which will direct future work (Section 6.2).

6.1 Concluding Remarks

Ontologies are becoming more and more important in a long list of application fields. We have observed that, partly for that reason, the process of ontology acquisition, creation and development is widely distributed, decentralized and dynamical, mostly manual (or semi-automatic) and strongly human-influenced. There is no common framework to provide general rules defining the structure, vocabulary, level of detail and perspective that should be considered when building an ontology.

Many of the existing ontologies inherit the consequences of these features of the process of ontology development. Although they might have been created with similar intentions, their interoperability is impaired by many mismatches on terminological, structural, conceptual and other levels – a phenomenon that we have called *ontology heterogeneity.* And as ontologies application finds broader demand and acceptance, more research is required in direction of enabling the compatibility of different ontologies, created by different people or communities in different places.

During the past 10-15 years, many research groups in various fields of AI, computer science, computational linguistics and cognitive science have put efforts in providing the knowledge management practitioners community with approaches to deal with the problem of ontology heterogeneity by developing methods for ontology matching or merging, based on various aspects of ontology similarity (structural, terminological, semantic, extensional) and with different "coordinates" on the scale *formal – applied* [119].

The work presented in this thesis is directed towards the development of computational models for reducing ontology heterogeneity. The proposed solutions account for different aspects of similarity of ontologies populated with natural language text documents. Our main scientific interest has been to provide theoretically well-grounded and practically efficient methods for *instance-based* ontology matching, lead by our belief that the most reliable sources of information about the semantic proximity of domain specific ontologies are exactly their extensions, or sets of instances.

However, we have also claimed that instance-based similarity judgments can be improved and optimized by taking the structure of the source ontologies into account. It has been often argued by different authors [43, 115] that structure alone is not sufficient and reliable ground to judge ontology similarity and structural techniques have to be applied in combination with other approaches, such as lexical, semantic or extensional matching techniques. While still considering this statement as being fully sensible, we have inverted its claim by placing the extensional methods in the center and claiming that structure-originating techniques should only come as a support to these methods, as means to improve their performance, guaranty their consistency and derive bits of additional similarity or dissimilarity assertions.

A central theoretical finding of this study is the fact that similarity of concepts can be measured on the basis of the features that characterize their instances, scored by their importance with respect to the concepts in question. We have proposed different variable selection techniques, which have the task of providing these scores. Concept similarity measures have been introduced based on the scored features and two resulting procedures for overall ontology matching using these similarity measures and combining them with structure have been developed.

The experimental part of our work contains evaluation and comparison of the suggested similarity measures, variable selection techniques and one of the overall ontology matching procedures. We have seen that the suggested procedures compete closely with or overpass the performance of standard approaches, for reasons we have explained previously. Summarized, the advantages of our contribution are that, in the first place, the proposed approaches do not build on the assumption that the two source ontologies are populated with the same sets of instances. The computation of the similarity of concepts is an inexpensive task once the variables are scored with respect to every concept. The latter is an outcome of a variable selection procedure, which is entirely accomplished at the training-phase of a binary classification task (in contrast to the rather time-costly training and classifying procedure performed in many approaches ([39, 89]). The most important features (words, when working with textual instances) are made available to the user by the selection procedure, which is helpful to immediately evaluate the quality of the similarity judgement. Finally, the structural relation of the concepts taken into account, the process is optimized by measuring similarity only on sets of potentially similar concepts, increasing the time efficiency of the method.

An extended account of the major theoretical results and contribution of the thesis have been given in the concluding section of Chapter 4. A summary of the experimental results and a list of advantages of our approaches compared to related techniques are found in the conclusion of Chapter 5. We will, therefore, not go into further detail here, but proceed to outline the open questions and directions for future work, which have been shaped during the work on this thesis.

6.2 Loose Ends

Usually, writing a PhD thesis takes longer time than planned in the beginning, and still, at the end, not sufficient to answer fully all the initial questions and the ones, which have occurred during the work. This thesis is no exception from this general rule that shapes the unfriendly terms, on which writing and time are.

In the current section, we will outline a set of problems and open questions that we had to leave out for time limitations with the hope to come back to them at a later point in the near future. We will roughly sketch directions for possible solutions of some of these problems, but most of them will be left simply as a list of related, unsolved issues, organized in three groups. We will first discuss a couple of theoretical questions. Later, we will list a few possible improvements of our approaches with respect to technical aspects and efficiency. Finally, we will draw future directions for more and better empirical evaluation of our results.

6.2.1 Theoretical questions

There are a couple of theoretical improvements, which, unfortunately, remained unaddressed within the frames of this dissertation work. We will spot out three of the most important ones. They will be subject of near future work.

Generalization

Section 4.2 of the thesis is dedicated to modeling ontologies as graphs and aspects of possible ways of measuring their similarity by solving a graph isomorphism problem. Although the discussed results and provided definitions are general and hold for all kinds of ontologies, the final procedure, which makes use of these results (Section 4.5) is defined on hierarchical, tree-like ontologies only. We have discussed a possible generalization of the suggested approach by applying a Cartesian product on trees, which leads us to more general constructs, such as DAGs and semi-lattices. However, more work needs to be done in direction of generalization in order to account for ontologies, which contain other than subsumption and `part_of` relations and to integrate the generalization component in the proposed procedure.

Ontological spaces

We have defined a set of ontologies presented as graphs and a metric on this set (see again Section 4.2). These definitions naturally give rise to a metric space and we suggest to explore the benefits of formalizing ontologies as elements of a metric space. If Ω is a set whose elements are ontologies, we can trivially define an ontological metric space (or ontological space) as the metric space (Ω, d), where d is a distance function of some kind (if we consider strictly ontology graphs, it can be the distance function from definition 43).

We can think of defining classes of similar ontologies by fixing a point O_0 in (Ω, d) and considering the open ball

$$B_{O_0} = (O | O \in \Omega, d(O_0, O) < c), \tag{6.1}$$

where c is in the range of d.

Modifying the radius c will enlarge or restrain the notion of similarity. In other words there can be found a non trivial *maximal common sub-ontology*[1] for all the ontologies contained in B. Larger the radius c, smaller the maximal common sub-ontology. This is a fact of particular interest with respect to the model selection procedures discussed in the following section. The notion of "reducibility" to a common sub-ontology should be explored on that bases by providing necessary and sufficient conditions for two ontologies to be reduced to a maximal common sub-ontology.

Model selection

Model selection is defined as the task of selecting the most appropriate mathematical model for a real life problem out of a set of potentially applicable models [22, 9]. This task often comes down to selecting the best for our goals values for a certain set of parameters. For this reason parameter selection is considered as a synonym of model selection by some authors.

In our case, the notion of ontology similarity is defined by and is dependent on a set of parameters. These are the number k of selected variables for the k-TF similarity measure, the similarity threshold l on the basis of which we judge of two concepts as similar or dissimilar; the parameters involved in the computation of the relative size of the intersection of the sets of instances, the SVM parameters, and so on. The different values of these parameters will define different similarity measures with diverse levels of restriction in their definitions. Therefore, a model selection mechanism should be available to (semi-)automatically choose the most appropriate values for the parameters for two given ontologies. In consideration should be taken the trade-off between too broad a notion of similarity, which will better guarantee finding faster similar ontologies, on the one hand, and, on the other hand, a too restrictive similarity measure which will better disambiguate ontologies and provide a more precise notion of similarity, but in the same time will fail to recognize the similarity between ontologies sharing smaller commonalities.

In order to provide an intuition about the problem, an insight to what different models to choose from can be will give us the following example. Let O_0 be the ontology whose similarity with a set of other ontologies we would like to identify and compare. Let us define the family of embedded closed balls B_{O_0} in the following manner:

$$\{B^i_{O_0} = (O | O \in \Omega, d(O_0, O) \leq c_i)\}, i = 1, 2, ... \; and \; c_i \leq c_{i+1}. \tag{6.2}$$

[1]For the time being, think of the term *maximal common sub-ontology* of a set of ontologies in an analogous way as of a maximal common sub-graph of a set of graphs.

Intuitively, our ideal model corresponds to that value c_{i*}, which defines a ball containing a set of ontologies that is in the same time rich enough (possessing a large enough cardinality) and the common sub-ontology of the ontologies contained in it is not of a too small order.

6.2.2 Technical improvements

From a technical viewpoint, with regard to the better performance of the matching approaches and similarity measures proposed in the thesis, we suggest that there is room for improvement in several directions, listed below.

Multi-linguistic ontologies

We have argued (see Section 4.3) that the proposed instance-based approaches to measure concept similarity are stable in multi-linguistic environments, i.e. that one can easily map a concept from one ontology containing documents written in one natural language, onto a concept in another ontology containing documents written in another natural language. The defined similarity criteria hold; the only technical modification of the procedure would be a preprocessing step, in which the document vectors (containing the document features) have to be rendered into a mutual language. And although this is a trivial enterprise, it needs to be elaborated on in the future, mainly because multi-language ontology matching is an important part of the field.

In connection to that, an interesting question arises: would the target language be chosen to be one of the languages already used in the document sets (e.g. the language, in which most of the documents appear), or should it be a neutral language; would two documents in two different languages, translated into a third language appear to be easier to compare than a document in its original language and a document translated into that language? Especially in the case of automatic translation this question might appear to be very important, because of the quality of automatic translation.

Automatic selection of parameters

The performance of the suggested procedures will be increased significantly if a model selection algorithm is implemented as an integral part of the matching procedures. The main objective of this algorithm should be to solve the task of setting the model parameters (as discussed in the section above) automatically or with as little human assistance as possible.

Lexical concept and relation matching

Several times, we made the point that a verification of similar or identical concept and relation names among the source ontologies is the most straightforward (although not self-sufficient) criterion for concept similarity or similarity of the relation of concepts. At the current state of affairs, we have basically assumed that the ontology graphs are unlabeled, name information is practically not used, for the sake of simplifying the suggested approaches, which focus on other aspects of ontology similarity. An important technical improvement of the ontology matching procedures, proposed in this thesis, would be to "equip" them with a name similarity checking criterion, as an integral component of their architectures.

An optimal similarity measure

A possible direction of future work in terms of improvement of the accuracy of concept similarity judgments is to apply multiple similarity criteria on the same mapping task before coming out with a final decision of the pair-wise similarity of the concepts. Within the framework of the results of the thesis, this can be done by combining the judgments of all the measures of concept similarity that we have elaborated (all based on variable selection, but using different variable selection techniques). On a broader scale, we would suggest to implement other concept similarity measures used in other matching tools (for instance the Jaccard coefficient [39]) and apply them simultaneously with the measures we propose. The ultimate goal would be to arrive at more precise and reliable judgments on the concept similarities by combining the performance of multiple similarity measures and building an optimal measure from all of them. Certainly, the cost to pay for gaining this bit of accuracy would be to increase the complexity and decrease the time efficiency of the mapping process.

Extending the concept similarity measure

We have defined a SVM-based variable selection measure for comparing concepts. We consider that a prominent direction of future work would be to extend the definition of this measure to using structural information via convolution kernels, as well.

6.2.3 Evaluation

Finally, due to time limitations we have left some empirical work unaccomplished. Below are several ideas about what is to be explored in terms of experimental evaluation in the near future.

Ontology merging

Section 4.5 of the thesis suggested a procedure, which combines instance-based concept similarity with graph-theoretic judgments of overall ontology similarity in order to match ontologies and lead an ontology engineer through a process of ontology merging, if merging is required for the particular application. For reasons stated in the introduction, the main focus of our work within the frames of this thesis falls on instance-based similarity approaches. Therefore the thesis contains only a partial evaluation of the suggested procedure, covering the extension-related concept mapping. A subject of future work will be the entire empirical evaluation of the procedure, including the graph-isomorphism-based measure of structural similarity of ontologies and the proposed techniques for combining and merging of intensionally overlapping parts of the input ontologies.

The human judgment

We suggest that human judgments on concept and ontology similarity can provide an important ground for evaluation of the performance of the proposed approaches. In many systems, the process of ontology matching is human mediated and in the best case semi-automatic. Therefore, a comparison of the results we achieve in mapping concepts by the help of our instance-based procedures with the guesses of ontology engineers and domain experts and an assertion about how these two types of judgment compete will be an useful argument, hopefully, in favor of our approaches.

Further, in a deeper attempt to study the intensional ambiguities stemming from the differences in conceptualization among people, we would suggest to carry out a psychological test on the human perception of structure of entities. In a rough sketch of the experimental set up, we would provide subjects with a set of concepts and several example taxonomies organizing these concepts and ask them to choose, which among the presented taxonomies represents best their idea of the structure of the concepts from the set. This can be a first step to study what the dissimilarities between ontologies created to represent similar domains are due to and provide ground for addressing the problem of ontology matching from its origins.

As a final observation, we note that it is a continuous practice among the ontology matching community to attempt to solve only pieces of the big problem of ontology heterogeneity; to provide partial solutions and local approaches in a just as decentralized and ununified manner as the heterogeneous ontologies that these approaches attempt to reconcile have been created. We admit that the work presented in this thesis is no exception from this "tradition". We would also claim that a global solution of the problem of ontology heterogeneity need not exist for ontology matching to be an useful support for engineers in development, maintaining and integration of ontologies.

However, an effort to bring ontology matching approaches into a single common framework is a valuable theoretical and practical enterprise. There are already some encouraging initiatives in that direction, providing a benchmark for comparing and evaluating the performance of ontology matching approaches with the objective to identify the areas in which each matching procedure is strong or weak (see more about the Ontology Alignment Evaluation Initiative[2] [23]). A further step would be to attempt to set a common formal framework for measuring ontology and concept similarity by integrating and optimizing the broad variety of theoretical and practical approaches, already available out there. In our view, intensive work in these directions is an urgent task for both researchers and practitioners in the field of ontology matching.

[2]http://oaei.ontologymatching.org/2009/

Bibliography

[1] K. Aas and L. Eikvil. Text categorization: A survey. In *Technical Report of the Norwegian Computing Center Num. 941*, 1999.

[2] A Agresti. *Categorical Data Analysis.* Hoboken: John Wiley and Sons, 2002.

[3] G. Antoniou and F. van Harmelen. Web ontology language: Owl. In S. Staab and R. Studer, editors, *Handbook on Ontologies in Information Systems*, pages 76–92. Springer-Verlag, 2003.

[4] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68, 1950.

[5] S. Auwatanamongkol. Inexact graph matching using a genetic algorithm for image recognition. *Pattern Recogn. Lett.*, 28(12):1428–1437, 2007.

[6] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In *Festschrift in honor of Jo"rg Siekmann, Lecture Notes in Artificial Intelligence*, pages 228–248. Springer-Verlag, 2003.

[7] F. Baader and W. Nutt. Basic description logics. *The description logic handbook: theory, implementation, and applications*, pages 43–95, 2003.

[8] L. W. Barsalou. Situated conceptualization. In H. Cohen and C. Lefebvre, editors, *Handbook of Categorization in Cognitive Science.* Elsevier, 2005.

[9] P. L. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 286–297. Morgan Kaufmann, San Francisco, 2000.

[10] C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[11] E. Bengoetxea. *Inexact Graph Matching Using Estimation of Distribution Algorithms.* PhD thesis, Ecole Nationale Supérieure des Télécommunications, 2002.

[12] T. Berners-Lee and M. Fischetti. *Weaving the Web: The Past, Present and Future of the World Wide Web by its Inventor.* Britain: Orion Business, 1999.

[13] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *J. Mach. Learn. Res.*, 3:1229–1243, 2003.

[14] G. Blanchard, O. Bousquet, and P. Massart. Statistical performance of support vector machines. *Annals Of Statistis*, 36:489, 2008.

[15] G. Blanchard, P. Massart, R. Vert, and L. Zwald. Kernel projection machine: a new tool for pattern recognition. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 1649–1656. MIT Press, 2005.

[16] J. L. Borges. *Other inquisitions 1937-1952.* New York: Washington Square Press, 1966.

[17] P. Buitelaar, P. Cimiano, and B. Magnini. Ontology learning from text: An overview. In P. Buitelaar, P. Cimiano, and B. Magnini, editors, *Ontology Learning from Text: Methods, Applications and Evaluation*, pages 3–12. IOS Press, 2005.

[18] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, August 1997.

[19] H. Bunke. Graph matching: Theoretical foundations, algorithms, and applications. In *Proc. Vision Interface 2000*, pages 82–88, 2000.

[20] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Lett.*, 19(3-4):255–259, 1998.

[21] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[22] K. P. Burnham and D. R. Anderson. *Model Selection and Multimodel Inference: A Practical-Theoretic Approach, 2nd ed.* Springer-Verlag, 2002.

[23] C. Caracciolo, J. Euzenat, J. Hollink, R. Ichise, A. Isaac, V. Malaisé, C. Meilicke, J. Pane, P. Shvaiko, H. Stuckenschmidt, O. Svab-Zamazal, and V. Svátek. Results of the ontology alignment evaluation initiative 2008. In *Ontology Matching Workshop, ISWC08*, 2008.

[24] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data.* Morgan Kaufmann, Amsterdam, 2003.

[25] C. Chemudugunta, A. Holloway, P. Smyth, and M. Steyvers. Modeling documents by combining semantic concepts with unsupervised statistical learning. *The Semantic Web - ISWC 2008*, pages 229–244, 2008.

[26] K. Church and P. Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 76–83, Morristown, NJ, USA, 1989. Association for Computational Linguistics.

[27] P. Cimiano, A. Hotho, G. Stumme, and J. Tane. Conceptual knowledge processing with formal concept analysis and ontologies. In *Proceedings of the The Second International Conference on Formal Concept Analysis (ICFCA 04)*, volume 2961 of *LNCS*. Springer, 2004.

[28] A. Collins and M. Quillian. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8(2):240–247, April 1969.

[29] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, December 1998.

[30] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, 2000.

[31] N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent semantic kernels. *J. Intell. Inf. Syst.*, 18(2-3):127–152, 2002.

[32] A. D. J. Cross, R. C. Wilson, and E. R. Hancock. Genetic search for structural matching. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 514–525, London, UK, 1996. Springer-Verlag.

[33] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, April 2002.

[34] J. David and J. Euzenat. Comparison between ontology distances (preliminary results). *The Semantic Web - ISWC 2008*, pages 245–260, 2008.

[35] S. Dennis. A memory-based theory of verbal cognition. *Cognitive Science: A Multidisciplinary Journal*, 29:145–193, 2005.

[36] S. Dennis and M. Harrington. The syntagmatic paradigmatic model: A distributed instance-based model of sentence processing. In M. Isahara and Q. Ma, editors, *proceedings of the 2nd Workshop on Natural Language Processing and Neural Networks*, pages 38–45, November 2001.

[37] M. Deza and E. Deza. *Encyclopedia of Distances*. Springer-Verlag, 2009.

[38] R. Diestel. *Graph Theory*. Springer-Verlag, 2007.

[39] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web, 2002.

[40] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA, 1998. ACM.

[41] M. Ehrig and J. Euzenat. State of the art on ontology alignment. *Knowledge Web Deliverable 2.2.3, University of Karlsruhe*, 2004.

[42] D. Estival, C. Nowak, and A. Zschorn. Towards ontology-based natural language processing. In *In RDF/RDFS and OWL in Language Technology: 4th Workshop on NLP and XML. ACL*, pages 59–66. Markowski A, 2004.

[43] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag Gmbh, 1 edition, 2007.

[44] N. Fanizzi and C. D'Amato. A similarity measure for the aln description logic. In *Atti di CILC 2006, Convegno Italiano di Logica Computazionale*, June 2006.

[45] A Faro, D Giordano, and Maiorana. F. Discovering complex regularities from tree to semi lattice classification. *International journal of computational intelligence*, 2(1), 2005.

[46] A. Ferrara, D. Lorusso, S. Montanelli, and G. Varese. Towards a benchmark for instance matching. In P. Shvaiko, J. Euzenat, F. Giunchiglia, and H. Stuckenschmidt, editors, *OM*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[47] R. Ferrer i Cancho and R. V. Sole. The small world of human language. *Proceedings of The Royal Society of London. Series B, Biological Sciences*, 268:2261–2266, 2001.

[48] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[49] P. Gaitanou. Ontology semantics and applications. In *2nd International Conference on Metadata and Semantics Research*, 2007.

[50] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, R. Oltramari, and L. Schneider. Sweetening ontologies with dolce. In *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 166–181. Springer, 2002.

[51] B. Ganter and R. Wille. *Formale Begriffsanalyse: Mathematische Grundlagen.* Springer-Verlag Gmbh, 1 edition, 1996.

[52] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences).* W. H. Freeman, January 1979.

[53] T. Gärtner, J. W. Lloyd, and P. A. Flach. Kernels for structured data. *SIGKDD Explorations*, 5:49–58, 2002.

[54] P. Geibel, H. Gust, and K.-U. Kühnberger. Variants of tree kernels for xml documents. In *MICAI*, pages 850–860, 2007.

[55] P. Geibel, O. Pustylnikov, A. Mehler, H. Gust, and K.-U. Kühnberger. Classification of documents based on the structure of their dom trees. *Neural Information Processing: 14th International Conference, ICONIP 2007, Kitakyushu, Japan, Revised Selected Papers, Part II*, pages 779–788, 2008.

[56] A.M. Glenberg and D.A. Robertson. Symbol grounding and meaning: A comparison of high-dimensional and embodied theories of meaning. *Journal of Memory and Language*, 43:379–401, 2000.

[57] G. Goller, J. Loning, T. Will, and W. Wolff. Automatic document classification: A thorough evaluation of various methods. *Intern. Symposiums für Informationswissenschaft*, pages 145–162, 2000.

[58] G. H. Golub and C. F. Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)*. The Johns Hopkins University Press, October 1996.

[59] H. Goodglass, A. Wingfield, and S. E. Ward. Judgments of concept similarity by normal and aphasic subjects: Relation to naming and comprehension,. *Brain and Language*, 56(1):138 – 158, 1997.

[60] T. L. Griffiths, M. Steyvers, and J. B. Tenenbaum. Topics in semantic representation. *Psychological review*, 114(2):211–244, April 2007.

[61] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, 1995.

[62] N. Guarino and C. Welty. Ontological analysis of taxonomic relationships. *Data and Knowledge Engineering*, 39:51–74, 2000.

[63] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(1):1157–1182, 2003.

[64] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, 2002.

[65] U. Hahn, N. Chater, and L. B. Richardson. Similarity as transformation. *Cognition*, 87(1):1–32, February 2003.

[66] A. Hameed, A. Preece, and D. Sleeman. Ontology reconciliation. In *Handbook of ontologies, International handbooks on information systems, chapter 12*, pages 231–250. Springer Verlag, 2004.

[67] J.A. Hampton. The role of similarity in natural categorization. In M. Ramscar, U. Hahn, E. Cambouropolos, and H. Pain, editors, *Similarity and categorization*. Cambridge University Press, 2000.

[68] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, corrected edition, August 2003.

[69] D. Haussler. Convolution kernels on discrete structures. Technical report, University of Santa Cruz, 1999.

[70] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 172–184, New York, NY, USA, 1974. ACM.

[71] B. Hu, Y. Kalfoglou, H. Alani, D. Dupplaw, P. H. Lewis, and N. Shadbolt. Semantic metrics. In *EKAW*, pages 166–181, 2006.

[72] A. Isaac, L. van der Meij, S. Schlobach, and S. Wang. An empirical study of instance-based ontology matching. *The Semantic Web*, pages 253–266, 2008.

[73] B.-J. Jain, P. Geibel, and F. Wysotzki. Svm learning with the schur-hadamard inner product for graphs. In *The 12th European Symposium on Artificial Neural Networks, ESANN'04*, pages 299–304, 2004.

[74] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, pages 137–142. Springer Verlag, 1998.

[75] T. Joachims. Svm light, http://svmlight.joachims.org, 2002.

[76] H. L. Johnson, K. B. Cohen, W. A. Baumgartner, Z. Lu, M. Bada, T. Kester, H. Kim, and L. Hunter. Evaluation of lexical methods for detecting relationships between concepts from multiple ontologies. *Pac Symp Biocomput*, pages 28–39, 2006.

[77] I. T. Jolliffe. *Principal Component Analysis.* Springer, second edition, October 2002.

[78] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (Prentice Hall Series in Artificial Intelligence).* Prentice Hall, 1 edition, February 2000.

[79] Y. Kalfoglou and M. Schorlemmer. If-map: An ontology-mapping method based on information-flow theory. *Journal on Data Semantics*, 2800/2003:98–127, 2003.

[80] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, January 2003.

[81] J. Kandola, J. Shawe-Taylor, and N. Cristianini. Learning semantic similarity. In *NIPS*, pages 657–664. MIT Press, 2003.

[82] G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Journal of Applied Statistics*, 29:119–127, 1980.

[83] F. C. Keil. *Semantic and conceptual development: An ontological perspective.* Cambridge: Harvard University Press, 1979.

[84] C. Kemp, A. Bernstein, and J. Tenenbaum. A generative theory of similarity. In *The Twenty-Seventh Annual Conference of the Cognitive Science Society*, 2005.

[85] M. Klein. Combining and relating ontologies: an analysis of problems and solutions. In Gomez A. Perez, M. Gruninger, H. Stuckenschmidt, and M. Uschold, editors, *Workshop on Ontologies and Information Sharing, IJCAI'01*, Seattle, USA, 2001.

[86] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273 – 324, 1997.

[87] R. R. Korfhage. *Information Storage and Retrieval.* John Wiley and Sons, 1997.

[88] M. S. Lacher and G. Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *In Proceedings of the 14th Int. FLAIRS Conference*, pages 305–309. AAAI Press, 2001.

[89] P. Lambrix, H. Tan, and W. Xu. Literature-based alignment of ontologies. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, editors, *OM*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[90] T. K. Landauer and S. T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, April 1997.

[91] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2):259–284, 1998.

[92] K. Lang. 20 newsgroups data set.

[93] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9(4):341–352, 1973.

[94] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins, and B. Scholkopf. Text classification using string kernels. *Journal of Machine Learning Research*, 2:563–569, 2002.

[95] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:563–569, 2000.

[96] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.

[97] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[98] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[99] E. Margolis and S. Laurence. *Concepts*. MIT Press, 1999.

[100] A. B. Markman and D. Gentner. Structural alignment during similarity comparisons. *Cognitive Psychology*, 25(4):431–67, 1993.

[101] J. J. Mcgregor. Backtrack Search Algorithms and the Maximal Common Subgraph Problem. *Software-pract. and exper.*, 12(1):23–34, 1982.

[102] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proc. 17th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 483–493, Colorado,USA, April 2000.

[103] A. Mehler, P. Geibel, and O. Pustylnikov. Structural classifiers of text types: Towards a novel model of text representation. *LDV Forum*, 22(2):51–66, 2007.

[104] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, August 2006. ACM.

[105] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and K. R. Müllers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48, 1999.

[106] T. Mitchell. *Machine Learning.* McGraw-Hill Education (ISE Editions), October 1997.

[107] P. Mitra and G. Wiederhold. An ontology composition algebra. In S. Staab and R. Studer, editors, *Handbook on Ontologies.* Springer, 2004.

[108] R. Mizoguchi. Tutorial on ontological engineering part 2: Ontology development, tools and languages. *Journal New Generation Computing*, 22(1):61–96, 2004.

[109] D. Mladenic. Feature subset selection in text-learning. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 95–100, London, UK, 1998. Springer-Verlag.

[110] F. Mougin, A. Burgun, O. Bodenreider, J. Chabalier, O. Loréal, and P. Beux. Automatic methods for integrating biomedical data sources in a mediator-based system. In *DILS '08: Proceedings of the 5th international workshop on Data Integration in the Life Sciences*, pages 61–76, Berlin, Heidelberg, 2008. Springer-Verlag.

[111] G. L. Murphy. *The Big Book of Concepts (Bradford Books).* The MIT Press, March 2002.

[112] D. Nardi and R. J. Brachman. An introduction to description logics. *The description logic handbook: theory, implementation, and applications*, pages 1–40, 2003.

[113] M. Neuhaus and H. Bunke. A convolution edit kernel for error-tolerant graph matching. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 220–223, 2006.

[114] N. Noy and M. Musen. Anchor-prompt: Using non-local context for semantic matching. In *IJCA '01: Proceedings of the Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence*, pages 63–70, August 2001.

[115] N. Noy and M. Musen. Evaluating ontology-mapping tools: Requirements and experience. In *In Proceedings of OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management*, pages 1–14, 2002.

[116] K. Oflazer. Error-tolerant retrieval of trees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(12):1376–1380, 1997.

[117] C. K. Ogden and C. K. Richards. *The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism.* Harcourt, reissue (june 1989) edition, 1923.

[118] E. Ovchinnikova and K.-U. Kuehnberger. Automatic ontology extension: Resolving inconsistencies. *GLDV-Journal for Computational Linguistics and Language Technology*, 22(2):19–33, 2007.

[119] E. Ovchinnikova and K. Todorov. Handbook on onotlogies - review notes. Unpublished, 2008.

[120] E. Ovchinnikova, T Wandmacher, and K.-U. Kuehnberger. Solving terminological inconsistency problems in ontology design. *IBIS-Interoperability in Business Information Systems*, 2:65–80, 2007.

[121] A. Pease, I. Niles, and J. Li. The suggested upper merged ontology: A large ontology for the semantic web and its applications. In *In Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web*, 2002.

[122] M. Pelillo, K. Siddiqi, and S. W. Zucker. Matching hierarchical structures using association graphs. *Lecture Notes in Computer Science*, 1407:3–??, 1998.

[123] M. F. Porter. An algorithm for suffix stripping. *Program*, 14:313–316, 1997.

[124] J. Quesada. Human similarity theories for the semantic web. In Christophe Guéret, Pascal Hitzler, and Stefan Schlobach, editors, *NatuReS*, volume 419 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[125] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[126] A. Rakotomamonjy. Variable selection using svm based criteria. *J. Mach. Learn. Res.*, 3:1357–1370, 2003.

[127] A Rakotomamonjy and F. Suard. Sélection de variables par svm: application á la détection de piétons. In *RFIA04*, 2004.

[128] J. Rocchio. *Relevance Feedback in Information Retrieval*, pages 313–323. 1971.

[129] E. Rosch. Human categorization. *Advances in Cross-Cultural Psychology*, 1:1–72, 1977.

[130] E. Rosch. *Principles of Categorization*, pages 27–48. John Wiley & Sons Inc, 1978.

[131] E. Rosch and B. Lloyd. *Cognition and Categorization.* Hillsdale NJ: Lawrence Erlbaum Associates, 1978.

[132] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Neurocomputing: foundations of research*, pages 89–114, 1988.

[133] A. Sanfeliu and K-S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, 13(3):353–362, 1983.

[134] B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[135] A. Schwering. *Semantic Similarity Measurment Including Spatial Relations for Semantic Information Retrieval of Geo-Spatial Data.* PhD thesis, Universität Münster, 2005.

[136] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[137] L. Shapiro and R. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:504–519, 1981.

[138] R. Shepard. Stimulus and response generalization: Tests of a model relating generalization to distance in psychological space. *Journal of Experimental Psychology*, 55(6):509–523, 1958.

[139] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22:183–236, 1990.

[140] V. M. Sloutsky. The role of similarity in the development of categorization. *Trends in Cognitive Sciences*, 7(6):246 – 251, 2003.

[141] J. F. Sowa. *Knowledge representation: logical, philosophical and computational foundations.* Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 2000.

[142] C. Spearman. The proof and measurement of association between two things. *Amer. J. Psychol., 15*, pages 72–101, 1904.

[143] S. Staab and R. Studer. *Handbook on Ontologies.* International Handbooks on Information Systems. Springer, January 2004.

[144] G. Stumme and A. Maedche. Fca-merge: Bottom-up merging of ontologies. In *In IJCAI*, pages 225–230, 2001.

[145] X. Su. A text categorization perspective for ontology mapping. Technical report, 2002.

[146] K. Taylor, H. Moss, and L. Tylor. The conceptual structure account: A cognitive model of semantic memory and its neural instantiation. *The Natural Basis of Semantic Memory*, 2007.

[147] A. Thor, T. Kirsten, and E. Rahm. Instance-based matching of hierarchical ontologies. In *The 12th German Database Conference (BTW)*, 2007.

[148] K. Todorov. Combining structural and instance-based ontology similarities for mapping web directories. In *ICIW '08: Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services*, pages 596–601, Washington, DC, USA, 2008. IEEE Computer Society.

[149] K. Todorov. Detecting ontology mappings via descriptive statistical methods. In *ICIW '09: Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services*, pages 177–182, Washington, DC, USA, 2009. IEEE Computer Society.

[150] K. Todorov and P. Geibel. Ontology mapping via structural and instance-based similarity measures. In *Third International Workshop On Ontology Matching (OM2008), 7th International Conference on the Semantic Web*, volume 431, pages 224–229. CEUR-WS, October 2008.

[151] K. Todorov and P. Geibel. Variable selection as an instance-based ontology mapping strategy. In *International Conference on Semantic Web and Web Services, Las Vegas (SWWS09)*, 2009.

[152] W-H Tsai and K-S Fu. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 9:757–768, 1979.

[153] A. Tversky. Features of similarity. In *Psychological Review*, volume 84, pages 327–352, 1977.

[154] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, January 1976.

[155] G. Valiente. *Algorithms on Trees and Graphs.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[156] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.

[157] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[158] P. Velardi, R. Navigli, A. Cucchiarelli, and F. Neri. *Ontology Learning from Text: Methods, Evaluation and Applications*, chapter Evaluation of OntoLearn, a Methodology for Automatic Learning of Domain Ontologies. IOS Press, 2005.

[159] S. Vishwanathan and A. Smola. Fast kernels for string and tree matching. In *Advances in Neural Information Processing Systems 15*, pages 569–576. MIT Press, 2003.

[160] J. T. L. Wang, B. A. Shapiro, D. Shasha, K. Zhang, and K. M. Currey. An algorithm for finding the largest approximately common substructures of two trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:889–895, 1998.

[161] Sh. Wang, G. Englebienne, and St. Schlobach. Learning concept mappings from instance similarity. In *ISWC '08: Proceedings of the 7th International Conference on The Semantic Web*, pages 339–355, Berlin, Heidelberg, 2008. Springer-Verlag.

[162] C. Wartena and R. Brussee. Instanced-based mapping between thesauri and folksonomies. In *ISWC '08: Proceedings of the 7th International Conference on The Semantic Web*, pages 356–370, Berlin, Heidelberg, 2008. Springer-Verlag.

[163] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *Advances in Neural Information Processing Systems*, volume 13, pages 668–674. MIT Press, 2000.

[164] W. J. Wilbur and K. Sirotkin. The automatic identification of stop words. *J. Inf. Sci.*, 18(1):45–55, 1992.

[165] L. Wittgenstein. *Philosophical Investigations.* Blackwell Publishing, 1953.

[166] S. Wu and P. Flach. Feature selection with labelled and unlabelled data. In *ECML/PKDD'02 workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning*, pages 156–167, 2002.

[167] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1:67–88, 1997.

[168] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann Publishers Inc., 1997.

[169] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.