M A G I S T E R A R B E I T

Semantic Metadata Enrichment

A Semi-Automatic Approach for Linking Legacy Metadata with Knowledge Organization Systems.

ausgeführt am Institut für
Distributed and Multimedia Systems
Universität Wien

unter Anleitung von Univ.Prof. Dr. Wolfgang Klas
und unter Mitwirkung von Mag. Dipl.-Ing. Bernhard Haslhofer

durch

Bernd Moser, Bakk. rer. soc. oec.
Matr. Nr.: 0201183
A - 1120 Wien, Murlingengasse 52/2

Wien, im Juli 2007

Abstract

As with the amount of a rapidly growing digitised content, the need for advanced search mechanisms is also growing. Traditional search engines do not use new approaches like *Ontologies*, which leads to poor knowledge discovery. The following work combines aspects of several sciences in order to increase the discovery and search mechanisms of search engines. To do so, the concepts of machine learning and information science are used to enrich metadata records by making them ontology-aware. A lot of enterprises are in possession of metadata records, which are useless to new discovery services, because of the missing ontology-awareness.

As a manual enriching process would be too time consuming, the thesis shows a semi-automatic way of making a record ontology-aware, by using machine learning algorithms and text recognition. The practical use of this new approach will be substantiated by a prototypical implementation and its corresponding case-study.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

As with the amount of a rapidly growing digitised content, the need for advanced search mechanisms is also growing. New technologies, like Ontologies, can improve the organization of knowledge and increase the quality of search engines. The first part of this thesis describes how the ideas of 2000 year old storing mechanisms have found their way into the digital world and how they can be used to boost search capabilities. Instead of using simple string-match algorithms an ontology aware search engine can interpret a data's semantic, which enables it to provide more accurate information to the user. Ontologies have been used by librarians and are now finding their way into *Knowledge Organization Systems*. Ontologies can express different levels of semantics, which can be used by reasoners to answer simple questions.

To accelerate the whole development process, the W3C has built the *Semantic Web Initiative [40]* which provides a common framework that allows data to be machine processable. The most important technologies that were defined as part of the semantic web are the *Resource Description Framework (RDF)* and the *Web Ontology Language (OWL)*. RDF provides a flexible data model to describe resources of any kind in a flexible way. OWL allows the definition of ontologies which are needed to standardise meanings in a certain domain and to define concepts, which are required to classify knowledge.

1.1 Motivation

Some Digital Libraries already use standardised ontologies and legacy metadata records to structure their data. The problem here is that many different standards are in circulation and that different institutions use different frameworks. In order to profit from the semantic web approach the existing data have to be transformed into machine interpretable records, which can be done by using the Resource Description Framework.

A transformed record can than be enriched with a relation to an ontology which in turn increases the scope and possibilities of search and discovery services.

On the syntactical level the transformation from a proprietary format into RDF is not a big deal. But the transformation by oneself does not change anything, because the data would still not be machine-interpretable. To gain full power of the Semantic Web technology the transformed metadata records need to be aware of its ontology, what is done by inserting a new relation. This is also called as semantic metadata enrichment. As a manually inserting of such a relation by the end-user would be a too tedious process, the thesis presents a semi-automatic approach for handling this annoying procedure. Semi-automatic enrichment can be done by using the power of machine-learning techniques, which will be theme of the second part of this work. Machine-learning algorithms try to classify data on the basis of features. In the case of a metadata record, a feature would be its textual representation. Different textual representations, lead to different classification as it would also have been the case with an human end-user. A classifier can be trained by an user, so that the classifier knows which terms a data record must to relate it to a specific part of the ontology. New classifiers, like the *Support Vector Machine (SVM)*, have been proven to have a good performance. Even the use of logical rules can be applied to improve the quality of the classification result.

To show the practical impact of semi-automatic enrichment a prototypical implementation has been built and analyzed, which is the last part of the thesis.

1.2 Goals

The main goal of the underlying study was to show, that a new approach of knowledge organization can be used to increase the possibilities of search- and knowledge-discovery-mechanisms. Another important part was to introduce a new way of enriching metadata records. Other approaches, which use the semantic web technologies, try to solve a similar but slightly different problem. Their goal is to obtain semantic rich metadata descriptions based on a resource's semantic or structural features. Which is opposite to enriching existing metadata records by linking them to concepts in ontologies. To be of practical use, a crucial factor is the quality of the machine-learning algorithm. As the work will show the semi-automatic enrichment approach is really a good possibility for linking existing legacy metadata with KOS.

Chapter 2

Knowledge Organization Systems (KOS)

In July 1945 Vannevar Bush, director of the office of scientific research and development, published an article where he faced the difficulties of the ongoing information growth and the inappropriate way of how machines handle this information. In chapter six he wrote:

> ”Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing. They are filed alphabetically or numerically, and information is found (when it is) by tracing it down from subclass to subclass. ...but the human mind does not work that way. It operates by association [6].”

V. Bush’s at that time fundamental idea is now present in the *World Wide Web (WWW)* where pages are referenced via hyperlinks. A study which has been conducted at the School of Information Management and Systems at the University of California at Berkeley [24] shows that new stored information between 1999 and 2002 grew about 30 percent a year. This shows that no matter at which domain we look, the need for knowledge discovery and information sharing are of paramount importance.

Science has developed two disciplines to handle the questions concerning information. *Computer Science (CS)* and *Information Science (IS)*. According to a panel discussion a distinction between these two fields is not that easy.

> ”At first we have to see that Computer Science as a discipline has undergone many phases of refinement. 25 years ago, Computer Science had to be all things to all people. As we have a data-centric economy today it is important to know how to store and retrieve data [26].”

.

Dr. Talburt reduces the problem on two principal differentiators between IS and CS.
These are:

- data-centric vs. algorithmic centric
- solution focused vs. software development focused

The core aspect of CS is to find algorithms. Data is only provided for executing and testing them. But in IS the general focus lies on the data. In combination the two sciences provide mighty tools to handle the information overload. While this chapter gives more insights on the data-approach, the next chapter focuses on the algorithmic approach by introducing the ideas of machine learning.

2.1 Semantics and Machine Interpretation

Since we have millions of documents in digitized form, we need to change the way we think about data [9]. The first people who realized the importance of data were computing professionals. Object-oriented programming languages began to provide facilities to process data. Later on, the W3C proposed the *Extensible Markup Language (XML)* in 1998 which based on *SGML* which was the key for machine interpretable data.

> ”Machine interpretable means that the semantics of the model is semantically interpretable by the machine; in other words, the computer and it’s software can interpret the semantics of the model directly - without human involvement [9].”

The vision of Tim Berners Lee *Semantic Web* may never become reality but at the moment a lot of effort is made to integrate and develop new technologies. Wherever documents are stored, sooner or later the need for a convenient way to access and find information will be crucial. The key question is how to get the data machine interpretable. Therefore, several ideas and concepts came up which all have in common that it’s about on how to organize information. Which leads us back to Mister V. Bush and his complaint that the human mind does not work like a machine.

2.2 Categorization, Conceptualization, Classification

When organizing information, it is important to do that in an unambiguous way. Like it is demonstrated in the text “categories” from Aristotle’s Organon [2]. There he wrote that categories are discrete entities characterized by a set of properties which are shared by their members. In analytical philosophy, these properties are assumed to establish the conditions which are both necessary and sufficient to capture meaning. In an other

way, we can think of categorization as a process in which ideas and objects are differentiated, recognized and understood. A category is also clearly defined, mutually exclusive, and collectively exhaustive. In the context of *Information Science*, it is common to use categorization as a process that divides a particular part of the world (also called the domain of interest) into divisions usually called concepts (A categorization is therefore also a conceptualization). A clear definition on the creation of a concept is given by Immanuel Kant in [21].

> "The logical acts of the understanding by which concepts are generated as to their form are: (1.) comparison, i.e., the likening of mental images to one another in relation to the unity of consciousness; (2.) reflection, i.e., the going back over different mental images, how they can be comprehended in one consciousness; and finally (3.) abstraction or the segregation of everything else by which the mental images differ. In order to make our mental images into concepts, one must thus be able to compare, reflect, and abstract, for these three logical operations of the understanding are essential and general conditions of generating any concept whatever. For example, I see a fir, a willow, and a linden. In firstly comparing these objects, I notice that they are different from one another in respect of trunk, branches, leaves, and the like; further, however, I reflect only on what they have in common, the trunk, the branches, the leaves themselves, and abstract from their size, shape, and so forth; thus I gain a concept of a tree."

A concept act as a substitute for a real- or possible-world-object. Concepts are represented by terms which stand-in for the underlying-meaning that is the concept itself, attributes, attribute values, and relationships to other concepts that concept participates in [9]. So conceptualization is the basic idea for organizing information. A result of this idea was classification. Up to now, classification is used by libraries to structure their "knowledge". One of the most famous library classifications is the *Library Of Congress (LOC)*[1] classification.

The next section covers some of the actual methods for organizing digital information and shows how to profit from ancient techniques of organizing wisdom.

2.3 KOS Spectrum

The main goal in handling the information growth is to make data machine-interpretable and to organize information in an unambiguous way. As studies by Eleanor Rosch [30] in the 1970s have shown (Cognitive Science) and what already should have been clear from the works of Aristotle and Linnaeus, a possible solution to traditional storing mechanisms could be classification. The first approaches for building mechanisms

[1] See http://www.loc.gov/catdir/cpso/lcco/lcco.html

to digitally represent data in a way of classification were databases which are up to now widespread. Starting with flat models, nowadays most databases use the relational model which has been introduced by E. F. Codd [7] in the 1970's. Although relational databases are still a widespread technology, there are better ways for the global storing of information. Especially when it comes to distributed data, as it happens in the WWW other techniques are needed. This was when ontologies found their way to the web.

Nowadays, every approach to formally represent knowledge is based on a conceptualization which again is "objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them [15]". One mechanism to formally represent knowledge corresponding to Information Science is an Ontology. Origination from philosophy, an ontology in the field of IS describes a "specification of a conceptualization [17]". In other words, an ontology is a description of the concepts and relationships that can exist (that can be represented) for an agent or a community of agents [17]. An ontology can vary in its semantics. That means that different notions are available for representing a domain of interest, as depicted in figure 2.1. Every notion tries to structure/classify/represent the concepts and relate them pertaining to some subject matter of interest and has to be accepted by a community otherwise it is useless. So it could also be seen as an "agreement to use the same terms in the same way [17]". In its simplest form this could be a list of terms, where the people who use it agree on restricting to the terms on the list. With higher needs for stronger semantics the complexity of the notions is increasing. To be able to handle higher complexities, there are different forms of control-mechanisms. Starting with simple classes and relationships they also introduce possibilities to build logical rules which will be used when strong semantics are needed.

As Information Science has two different fields of interest and development - the younger field evolved from Computer Science and the traditional science is coming from the libraries domain - also the terminology is sometimes different. Nevertheless, the main goal of both disciplines is to structure information by using the models mentioned above.

2.4 Controlled Vocabulary

A Controlled Vocabulary (CV) starts with the fundamentals of structuring information and is part of every ontology. CVs have been used by librarians before they entered the digital world. There are a few standards around describing the purposes, construction and use of controlled vocabularies. One of them is the *ISO 2788:1986 standard [20]* where guidelines for the establishment and development of monolingual thesauri are documented. As it will be showed in section 2.8.4, a thesaurus uses all ideas of a CV to structure knowledge. Another very good, free and more on controlled vocabularies focused standard is the *ANSI/NISO Z39.19-2005 standard [28]* that provides guidelines for the construction, format, and management of mono-

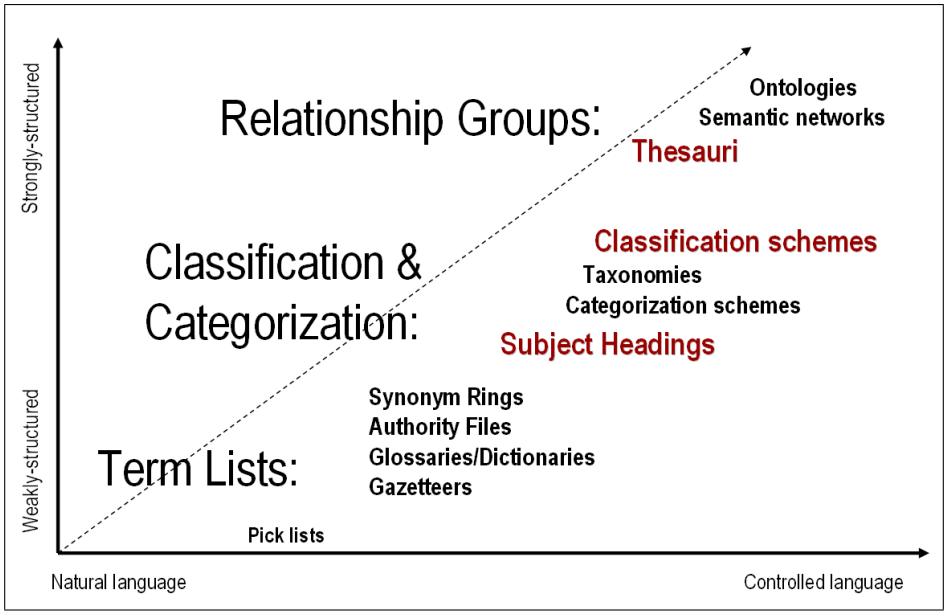

Figure 2.1: The Otology Spectrum [46]

lingual Controlled Vocabularies. According to the ANSI/NISO standard, the purpose of controlled vocabularies is to provide a means for organizing information. This is done through the process of assigning terms selected from controlled vocabularies to describe documents and other types of content objects. “It may have no meaning specified and it could be just a set of terms, that people agree to use [36]”. In natural language the human being often uses different terms for the same meaning, which enables it to vary, specify and enrich its sentences. But when organizing information, this freedom leads to confusion and reduces the results retrieved by a search engine.

> ”People search in the same language they speak, natural language, so a more advanced controlled vocabulary needs to take the concepts of the users (natural language) and match them to the concepts expressed in the language of your documents (controlled vocabulary) [36]”.

The converted natural language can be used for indexing and retrieval. This process is also known as *translation*.

In an information architecture there are usually two places where controlling labels and terms is useful [43]:

- a navigation scheme, which should use unambiguous labels and where the primary organization is usually hierarchical

- a search system, where search terms are selected and organized for tagging content, now usually through a content management system and a search engine.

One possibility to improve the searchability of a document or web page is to use consistency which already is a simple form of a controlled vocabulary. A consistent labeling system encourages uniformity in the term format and in the assignment of terms.

Basically, there are four levels of control. Depending on the chosen levels, a controlled vocabulary has either weak or strong semantics. While the first two levels of control deal with the terms itself, the latter two deal with the more complex concepts. Hence ambiguity and synonym control are also subsumed under *terminology control* while equivalence, hierarchical and associative relationships are also known under *semantic linking*. From simplest to most complex they are:

- Ambiguity control
- Synonym control
- Equivalence relationships
- Hierarchical relationships
- Associative relationships

2.4.1 Ambiguity Control

The distinction of ambiguous terms is the first step when building a controlled vocabulary. Ambiguity occurs when a word or phrase has more than one meaning. Acronyms and abbreviations often stand for more than one word or phrases. Therefore the full form of the term should be selected. Ambiguous terms could be homographs and polysems.

A polyseme is...

> ”a word with multiple meanings. In spoken language, polysems are called homonyms, while in written language they are called homographs [28].”

Multiple meanings come from a regional or historical evolution. The use of homonyms in controlled vocabularies should be avoided as far as possible . If used notwithstanding, a qualifier is needed to clarify the meaning of the term. Such a qualifier specifies the domain of meaning to which the term belongs.

Example for the word bank: (see [45])

- depository financial institution
- sloping land (especially the slope beside a body of water)
- an arrangement of similar objects in a row or in tiers
- the funds held by a gambling house
- a flight maneuver

2.4.2 Synonymy Control

Synonymy control varies in its graduations. But no matter which graduation we are looking at, it always groups terms with the same or roughly the same meaning together.

- Normally, real synonyms are very rare and only occur if the writing of the term is different.
  specialised - specialized.
- Also using an abbreviation instead of the long form is a synonym.
  UN - UNO - United Nations Organisation
- Terms that have different connotations according to their period of time or their region of use are synonyms too.
  pound: sexual; to fornicate forcefully, financial; british form of currency, culinary; a type of cake, weights and measures; an american weight measurement, physical; to hit forcefully, imprisonment; a holding area for animals[2]
- There is only a slight different in the meaning so that it could be hardly recognized.
  play - spectacle

2.5 Semantic Linking - Classification

Ontologies have a hierarchic structure. This means when navigating through concepts (i.e. when going from top to bottom) each concept has a more specific meaning. When building hierarchical structures it is important to know how deep or how specific the concepts should get. Deep hierarchies may get bulky and fuzzy or concepts will only have a few documents associated what makes it hard to retrieve them. Therefore it is a good idea to think about how deep or specific the structure should be. Once this is done, the next step

[2] See http://myconnotation.org/

is to build concepts out of the terms gained from the earlier steps. As already written concepts stand in for the real-world item and describe only the idea of it defined by the underlying terms. More details on building concepts out of terms are given in section 2.8.4.

2.6 Equivalence Classes

By using control mechanisms, the original vocabulary has been given a structure, which already could be used to improve navigation of the stored information. Ambiguity control separated terms with multiple meanings to different concepts, and synonymy control marked distinct words or phrases that have nearly the same meaning. The result were unambiguous terms describing the universe of discourse. In *Information Science (IS)* these resulting concepts are also called equivalence classes because each term in one specific class is equivalent to an other of this class. To work with those classes an identifier is needed. An identifier could either be a preferred term (aka descriptor or label) that represents the class or it could be a number. Using a number as identifier has the advantage that all terms in a concept can be used by a retrieval or indexing machine when looking for documents. Some systems offer the possibility to look for documents containing the given search term regardless of the other terms in the equivalence class. This may decrease the duration of the search. Another advantage is that equivalence classes can be easily changed which might not be the case when using preferred terms. The big disadvantage using numbers as identifiers is the fact that we loose control. Control in the way of a standardized language. And loosing the control in a controlled vocabulary should not happen. Therefore a very common approach is to define preferred terms or descriptors for the equivalence classes. The remaining terms are called variant terms (non-descriptor) and point to the preferred term. Because of its importance a descriptor should have the following properties:

- represent the equivalence class unambiguous, complete and clear
- language oriented
- easy to remember

Depending on the speech area, labels are either substantives (english: plural, german: singular) or adjectives or verbs.

2.7 Relationships

Now that each concept is represented by a descriptor and the underlying terms, it is possible to relate those classes. This again increases control but also complexity. The related classes form a semantic net which makes it easy to browse through the concepts (e.g. from more general to more specific terms). The linking is done via standardized abbreviations defined in *ISO-2788 standard [20]*. There are a few kinds of relationships:

- Equivalence Relationship
- Hierarchical Relationship
- Associative Relationship

2.7.1 Equivalence Relationship

Synonymy control has shown that terms with a same meaning can build an equivalence class. Equivalence classes are represented by their descriptor to which non-descriptor terms should point. This relationship is known as an equivalence relation. The below example shows an equivalence relationship between the more common word “zucchini” and “courgette” which is also sometimes used. When using a search engine, the common syntax would tell the computer that a search for courgette is equivalent as for Zucchini. Here the more frequent word Zucchini would be the descriptor term [43].

Courgette USE Zucchini

This example also shows that equivalence relations should always be reciprocal.

2.7.2 Hierarchical Relationship

The results of terminological control are equivalence classes which, in a first step, have been synonymically related. The next step is to bring those classes in a hierarchy which should correspond to our human understanding of generalization/specification. A hierarchical relation is either generic or partitive.

Generic Relationship

The concept of inheritance from object-oriented programming follows the same idea as generic relationship. Classes are put in a hierarchy where the properties of one class are used as a basis for a new class that is to be defined. The new class or sibling must at least have one distinguishing property that makes it unique of

it's parent class. A "vehicle" for example is a more general term for "car". The "car"-class should therefore inherit the properties of the *"vehicles"-class* plus a distinguishing property. A common method for building a generic relation is the bottom-up approach. This means that first the individual classes are specified in great detail and then linked to classes with a more general meaning. Thomas Aquinas already suggested this method in 1255:

> "Since we ought to acquire knowledge of simple things from composite ones and come to know the prior from the posterior, in instructing beginners we should begin with what is easier, and so we shall begin with the signification of being and proceed from there to the signification of essence [1]."

Partitive Relationship

A partitive relation is a relation between two concepts, where one of the concepts constitutes the whole and the other concept a part of the whole. A tree for example represents the whole while a tree trunk only represents a part of it. Though the generic and the partitive relation are different from its definitions, ontologies do not distinguish between them.

Example:

Tree NT Fruit Tree (NT = narrower term)
Fruit Tree BT Tree (BT = broader term)

As with equivalence relations it is suggested to relate the terms reciprocal. An ontology known for its use of hierarchic relations is a taxonomy as section 2.8.3 shows.

2.7.3 Associative Relationship

> "This relationship covers associations between terms that are neither equivalent nor hierarchical, yet the terms are semantically or conceptually associated to such an extent that the link between them should be made explicit in the controlled vocabulary, on the grounds that it may suggest additional terms for use in indexing or retrieval. The relationship is symmetrical, and is generally indicated by the abbreviation RT (related term) [47]."

Example:

cells RT cytology
cytology RT cells

The associative relationship is the most problematic one. It tends to be the reservoir for all relations that have in some way a connection with the classes. An associative relationship should form a cross-relationship to possibly find a better descriptor, and therefore only be used if one term implies the other term. Associative relationships have no direction and could therefore cause redundancies (one term is referred by a hierarchic relationship and also by an associative relationship). For a more detailed view see section 2.8.4.

2.8 KOS - Examples

Figure 2.2 shows examples of control mechanisms

> "...by the requirements of the types of relationships each must accommodate. It also shows that the more complex vocabularies (taxonomies, thesauri) include the simpler structures (lists, synonym rings). For example, a Thesaurus includes explicit devices for controlling synonyms, arranging hierarchies, and creating associative relationships while a List is a simple set of terms containing no relationships of any kind [28]."

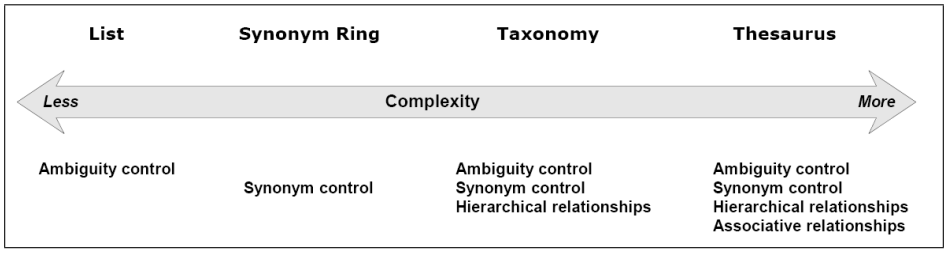

Figure 2.2: Increasing structural complexity of controlled Vocabularies
[28]

2.8.1 List

A list simply contains terms that describe entities. The number of terms is limited and is an agreement of users to use them. Lists are arranged in a logically way (alphabetical, by size, etc). Examples are geography, language or format lists.

Example (List of austrian states in alphabetical order):

Burgenland

Carinthia

Lower Austria

Upper Austria

Salzburg

Styria

Tirol

Vorarlberg

Vienna

2.8.2 Synonym Ring

When using search engines a user wants to retrieve all documents including his search term. Documents using a synonym instead of the search-term will not be retrieved though the user may like it. Synonym rings ensure that a concept that can be described by multiple synonymous or equivalent terms will be retrieved if any one of the terms is used in a search.

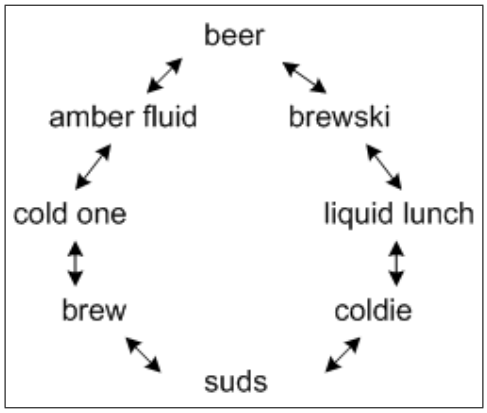

Figure 2.3: A synonym ring for beer

2.8.3 Taxonomies

In the middle of the 18th century Carolus Linnaeus, a swedish scientist, began to classify nature within a hierarchy. For two centuries his taxonomy enabled people to identify, name and describe plants. Now the Linnaean taxonomy is known by many botanists. The first person who tried to classify the human knowledge was Aristotle. In our life we are surrounded by classifactions. Classification makes our life easier because it organizes knowledge. Think of a supermarket. How would you find your goods if they are not in any order. Or think of going to the library looking for a book without knowing where to look for it. A simple classification only separates things and group them with respect to their properties. Things could be concrete

(e.g. Computer, Tables, ...) or abstract (e.g. Mathematics, Physics, ...). A taxonomy is a special form of a classification with special properties. The definition of the information technology field for it is:

> The classification of information entities in the form of a hierarchy, according to the presumed relationships of the real-world entities that they represent.

The first thing is, that we have a hierarchy. Think of it as a tree, that means it has a root and branches which are also called nodes. Each node stands for a real-world entity. Nodes are differentiated by a distinguishing property that makes it unique. Connections between nodes can represent the relation (also called link). As written above, a hierarchic relationship relates its classes from generalized to specialized concepts. Sibling are also known as the "is subclassification of"-relation (the link's arrow is pointing up at the child node) and the parents as the "is superclassification of"-relation (the link's arrow is pointing down at the child node). According to the terminology above the "is superclassification of"-relation is known as *Broader Term (BT)* and the "is subclassification of"-relation as *Narrower Term (NT)*-relation. When going downward, entities will become more specific, as in the other way they will become more general (Generalization/Specification). Figure 2.4 gives an example of a Taxonomy for classifying pants.

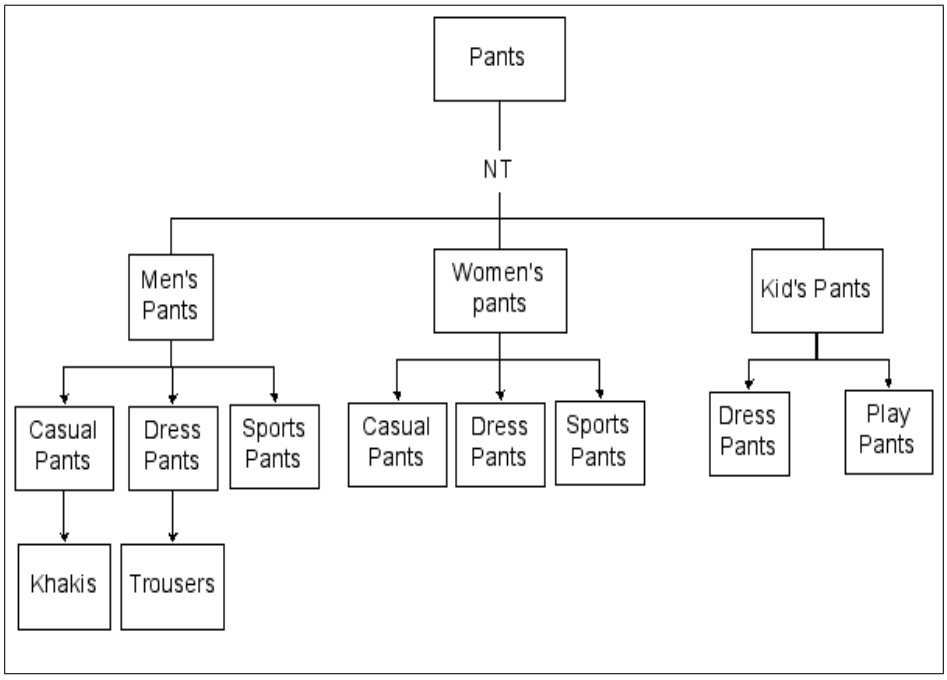

Figure 2.4: A simple Taxonomy

Today many taxonomies exist and especially librarians are familiar with them. Well established taxonomies are the Dewey Decimal Classification (DDC)[3] which is the world's most widely used library classification system, the Linnaean taxonomy which we have already heard of and which is used in the "tree of life-project[4]", the United Nations Standard Products and Services Code (UNSPSC) which provides an open, global multi-sector standard for efficient, accurate classification of products and services[5] and many more.

The Taxonomy is the basic structure for the information space. With an increasing need of more specific search-tasks we need stronger semantics. How to express stronger semantics in a machine understandable way will be discussed in the next section.

2.8.4 Thesauri

> "A thesaurus is a controlled vocabulary arranged in a known order and structured so that the various relationships among terms are displayed clearly and identified by standardized relationship indicators. Relationship indicators should be employed reciprocally [28]."

The primary purpose of a thesaurus is to facilitate the retrieval of documents and to achieve consistency in the indexing of written or otherwise recorded documents. A Thesaurus uses all levels of control and has the most relevance of all ontologies because of its use in libraries. When creating a Thesaurus the following steps have to be considered:

- Controlling the universe of discourse
- Term Collection

2.8.5 Controlling the universe of discourse

While natural language basically provides a vocabulary for all situations, a thesaurus only fulfills its use if its universe of discourse is clearly defined. As a matter of complexity, all attempts to build a universal thesaurus failed. When starting to build a thesaurus some things should be considered:

- The topic of the thesaurus (emphasis, fringe)
- Deepness of hierarchy
- Language style of the thesaurus (scientific, common)

[3] See http://www.oclc.org/dewey/
[4] See http://tolweb.org/tree/
[5] See http://www.unspsc.org/

- size (vocabulary, relationships, ...)

Because every thesaurus differs in its use there is no unique way to set the above parameters.

2.8.6 Term Collection

After knowing what the thesaurus should represent, the next step is to define its labels or terms. There are several options for obtaining terms:

- potential users and experts
- specialist dictionaries and norms
- technical literature
- existing thesauri or classification systems
- standard works, manuals
- Internet

Each source has its own level of detail, so a good idea is to collect terms from different sources. In some cases, if the thesaurus is not very big for example, it is probably easier to start from scratch instead of going through all sources finding the right terms. Sometimes databases that store search requests are also a good source when looking for terms. The Web offers also a lot of sites that collect links to electronic vocabularies. When looking through the collected terms it is advisable to do a rough classification. Burkart [5] recommends to use 50-150 terms per class. It is suggested to annotate the terms (notation, source, allocation to classification, existing relations). The terms should have a unitary form (singular/plural, no abbreviations). As a thesaurus is a controlled vocabulary, the next thing is to go through the different levels of control and to apply each level to the thesaurus. Thesauri do not differ between polysems and homonys. There is also only one type of relation no matter if it is generic or partitive.

Designing a thesaurus on paper may seems useless in the digital age. Nevertheless it the fastest way to organize terms and get a good overview of the originated structure. But there is still need for a digital representation. Possible rudiments to digitally represent thesauri or other controlled vocabularies are shown in section 2.9.

2.9 Metadata

With the growth of digital media the need for indexing the media grew. In 1967 most of the media was text documents. Obviously, the first field that had a demand for some indexing mechanism were digital libraries. Different libraries have different books. When searching for books, how does one library know about the books of the other? The answer is metadata. Exchanging the documents, so that every library has every document of the other libraries would be a waste of storage but also impossible due to the huge amount of information. Instead libraries exchange data about data, that necessarily is machine readable. There are two concepts nowadays which make use of metadata:

- Metadata Standards
- Metadata Frameworks

2.9.1 Metadata Standards

When using a standard, one agrees to stick to the suggested terminology. Doing so increases the interoperability of documents between other institutions using this standard. The first approach to exchange metadata in order to build a global indexing mechanism was in 1969 by the *Library Of Congress (LOC)* and known under the name *MARC II (=machine readable cataloging record)*. Than and now the idea is to define a format on which the community agrees. To be able to get a broad agreement, the "global players" of the community (representative of different fields e.g. libraries, education, international organizations, foundations, ...) try to define a standard. Today there are several metadata standards for different fields of application. Every standard tries to define descriptors, also known as categories, that are best for describing the belonging media. With many standards the interchangeability between them decreases. If a library, for example, wants to index multimedia or other documents, it may has to transform the different standards to their own standard, to be able to represent the documents in their database. Figure 2.5 gives an overview of some of them.

Dublin Core (DC)

Dublin Core[6] is the result of a workshop, held in 1995 in Dublin. There the community agreed on a standard, that includes 13, now 15, core elements (or categories) which can be used for describing digital documents. As the DC-Standard is an agreement on the vocabulary between the users, it is also a form of controlled vocabulary. The Dublin Core Metadata Initiative wants to keep the standard simple. Other features of the architecture are open extensibility, modularity, refinement and a consistent non-overlapping semantics[7]. A

[6] See http://dublincore.org
[7] See http://dublincore.org/documents/dcmi-ieee-mou/index.shtml

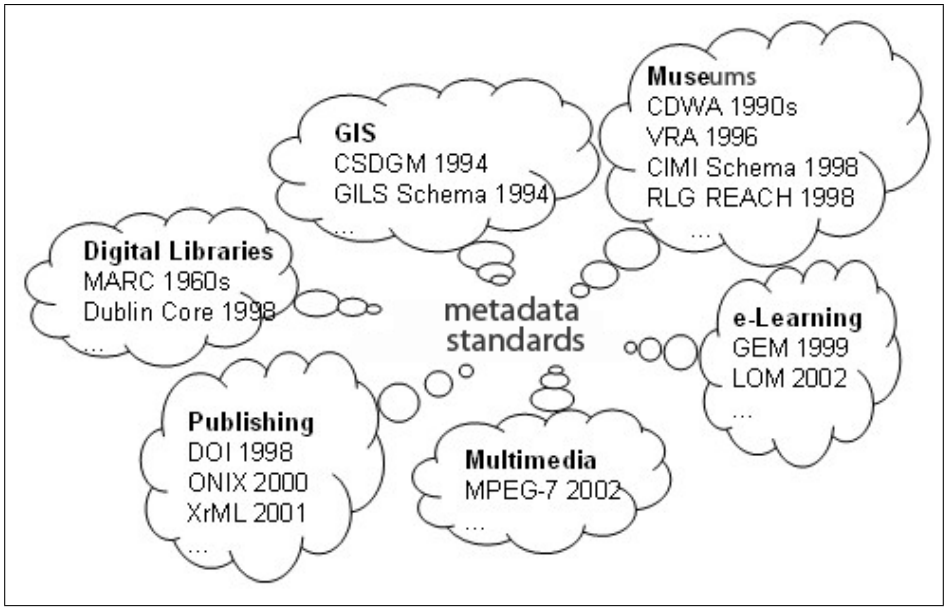

Figure 2.5: Some metadata standards [22]

DC-record can vary in its syntax. The available formats are HTML, XML or RDF. Each element of the element set uses a descriptive label as well as a unique name [27]. Listing 2.1 shows a simple dublin core record that describes a guide to growing roses [12].

Listing 2.1: DC-Example

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/">
   <rdf:Description rdf:about="http://media.example.com/audio/guide.ra">
      <dc:creator>Rose Bush</dc:creator>
      <dc:title>A Guide to Growing Roses</dc:title>
      <dc:description>Describes process for planting and nurturing
          different kinds of rose bushes.
      </dc:description>
      <dc:date>2001-01-20</dc:date>
   </rdf:Description>
</rdf:RDF>
```

2.9.2 Metadata Frameworks

While standards have pre-defined elements to express a specific universe of discourse, frameworks are more general. They only restrict to a syntactical standard. Defining the elements, their relations and logical rules is up to the user. Some frameworks specialise on the use of a specific notion (e.g. SKOS for Thesauri) and some of them can be used for any kind of a controlled vocabulary (e.g. RDF)

SKOS

On the basis of a short example the use of the RDF-based language *SKOS [41]* will now be shown. SKOS "provides a model for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, 'folksonomies', other types of controlled vocabulary, and also concept schemes embedded in glossaries and terminologies [41]."

After going through the above steps a concept can look as follows:

```
Motor vehicles

Used For
    * Automotive transportation
    * Coaches and carriages (motor)
    * Trucks

Microthesaurus:
    * 6.60 Equipment and facilities

      Broader term
          o Vehicles

      Narrower terms
          o Ambulances
          o Armoured fighting vehicles
          o Buses
          o Cars
          o Charabancs (motor)
          o Fire engines
          o Hackney carriages

Related terms
    * Road engineering
    * Road transport
```

Figure 2.6: Motor Vehicles example from UKAT-Thesaurus (www.ukat.org.uk)

Figure 2.6 shows a possible Taxonomy for motor vehicles. The taxonomy includes synonym control (=Used For) and relationships to narrower and broader terms. With SKOS, which stands for Simple Knowledge Organisation System, a transformation from this "handwritten" concept into a machine interpretable (=digital) form is possible. SKOS has a special vocabulary for expressing a thesaurus and is an application of

the Resource Description Framework (RDF). A SKOS vocabulary can be expressed as a graph which makes it easy to navigate. A good tool for representing RDF-files as a graph is IsaViz [38] which is a visual environment for browsing and authoring RDF models represented as graphs. The SKOS vocabulary nearly uses all common abbreviations what makes it to build one. Based on XML a SKOS-Element is defined by its namespace and the elements name. Related concepts are addressed by its URI as it is common practice to represent RDF-Resources - and a concept here is a resource - this way. The most important elements are(descriptions taken from W3C's SKOS Core Guide):

skos:concept The skos:Concept class allows you to assert that a resource is a conceptual resource. That is, the resource is itself a concept.

skos:prefLabel and skos:altLabel The skos:prefLabel and skos:altLabel properties allow you to assign preferred and alternative lexical labels to a resource. This is the same as the equivalence relationship where USE is the preferred label and UF (=Use for) is the alternative label.

skos:broader and skos:narrower To assert that one concept is broader in meaning (i.e. more general) than another, where the scope (meaning) of one falls completely within the scope of the other, use the skos:broader property. To assert the inverse, that one concept is narrower in meaning (i.e. more specific) than another, use the skos:narrower property.

skos:related To assert an associative relationship between two concepts, use the skos:related property.

Listing 2.2 shows a SKOS-Representation of the above shown motor vehicles example.

Listing 2.2: SKOS-Example

```
1 <skos:Concept rdf:about="http://www.ukat.org.uk/thesaurus/concept/1110">
2         <skos:prefLabel>Motor vehicles</skos:prefLabel>
3         <skos:altLabel>Automotive transportation</skos:altLabel>
4         <skos:altLabel>Coaches and carriages (motor)</skos:altLabel>
5         <skos:altLabel>Trucks</skos:altLabel>
6         <skos:inScheme rdf:resource="http://www.ukat.org.uk/thesaurus"/>
7         <skos:inScheme rdf:resource="http://www.ukat.org.uk/thesaurus/
              micro/660"/>
8         <skos:broader rdf:resource="http://www.ukat.org.uk/thesaurus/
              concept/498"/>
```

```
9           <skos:narrower rdf:resource="http://www.ukat.org.uk/thesaurus/
                concept/12105"/>
10                  ...
11          <skos:related rdf:resource="http://www.ukat.org.uk/thesaurus/
                concept/5092"/>
12          <skos:related rdf:resource="http://www.ukat.org.uk/thesaurus/
                concept/5093"/>
13          <rdfs:seeAlso rdf:resource="http://www.ukat.org.uk/thesaurus/
                term.php?i=1110"/>
14  </skos:Concept>
```

OWL

In February 2004 the W3C published the recommodation [42] for a *Web Ontology Language (OWL)*. OWL is a combination of the former languages DAML+OIL and is RDF-based. It provides a great machine interoperability and is often mentioned with the Semantic Web. It is used to build ontologies and offers the most complexity. OWL uses the ideas of description logic, set theory and inheritance as known from object oriented programming. Like RDF it models the relations between objects. OWL has three sublanguages that vary in its expressiveness.

OWL Lite: supports users who primarily need a classification hierarchy and simple constraints. Like a cardiniality of 0 or 1. Provides a quick migration path for thesauri and other taxonomies. It also has a lower formal complexity than OWL DL.

OWL DL: for users who want maximum expressiveness while retaining computational completeness and decidability. All language constructs can be used but under certain restrictions. OWL DL is so named due to its correspondence with description logics.

OWL Full: gives users maximum expressiveness and the syntactic freedom of RDF. There is no computational guarantee. ”It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full [42].”

Every OWL-mode is an RDF-model and every RDF-model is an OWL-Full document. The pre-defined elements can be used by a reasoner to answer questions or classify new elements. OWL has different features related to RDFS [39] like Class, rdfs:subClassOf, rdf:Property, rdfs:domain and more. To model equality and inequality it has for example equivalentClass, equivalentProperty, sameAs, differentFrom features. It is also possible to further specify property characteristics or property restrictions. With OWL FULL the possibilities when constructing an ontology are nearly impossible. A good example which shows the mightiness of OWL is the wine ontology [33], which is also used in the *"OWL Web Ontology Language Guide [42]"*. Listing 2.3 shows that WhiteWine is exactly the intersection of the class Wine and the set of things that are white in color. This means that if something is white and a wine, then it is an instance of WhiteWine.

Listing 2.3: OWL-Example

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasColor" />
      <owl:hasValue rdf:resource="#White" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

A practical example for machine interoperability is the wine agent[8] from Stanford university which uses the above mentioned ontology, to recommend wines to accompany meal courses.

[8] See http://www.ksl.stanford.edu/people/dlm/webont/wineAgent/

Chapter 3

Technical Background

It has already been stated, that information science has a data-centric focus. It concerns about how to organize data and the possibilities of assigning metadata to it to increase the navigation and findability of stored data records. Computer science in contrast has its focus on algorithms. Where information science tries to structure and organize the information, computer science offers possibilities to find the right piece of information in an organized structure. To handle the information overload, an automated approach for retrieving information is a common paradigm nowadays. While in the late '80s standard categorization mechanisms were applying rules (=inference) to classify media (=Knowledge Engineering), *Machine Learning (ML)* approaches have evolved in the recent years. ML is a subfield of artificial intelligence (AI) and offers several algorithms to automatically categorize media. Automatic categorization is used in many contexts and since increasing computing power, during the last few years, became a major subfield of information systems. Automatic categorization saves time and manpower and, as recent studies have shown, is nearly as accurate as an expert user [32]. When looking for information, it is either possible to retrieve documents containing the relevant information or to directly get the relevant text passages of retrieved documents. The first method is what search engines do and is known under *Information Retrieval (IR)*. The second one is quite different from IR and also more complex. *Information Extraction (IE)* tries to analyse the language and the semantics of the searched documents [8]. Though these two sciences use completely different algorithms, it is possible to combine IE and IR in order to get better search results. How a combination of IE, IR and ML-techniques can help to retrieve good search results and how to analyse the results will be shown in the following sections.

3.1 Categorization, Classification

Information can be organized using categories, so the next question is, how to find the "right" documents for each category. Right documents are assigned with a boolean value that is "true" for each pair of $\langle d_j, c_i \rangle \in DxC$, where D is a domain of documents and $C = \{c_1, ..., c_{|C|}\}$ is a set of predefined categories. And "false" if d_j should not be filed under c_i [32]. When categorizing text documents, a text document (d_j) can either be assigned to one or more categories. The first case is also known as single-labeled and the latter as multi-labeled categorization. A special single-labeled case is binary categorization. This means that the document d_j can only have category c_i or its complement $\bar{c}_i$ assigned to it. Binary TC can also be used to reduce multi-labeled categorization to single-labeled categorization. 50 years ago, classification was done manually by human experts like librarians for example. At that time the *Dewey Decimal Classification (DDC)* (see section 2.8.3 for more information) was widely used. Classification systems categorize the knowledge of a special domain of discourse. In the case of the DDC the domain of discourse is all human knowledge. To be able to classify documents (in the case of the DDC, the documents are books) under one category (=binary classification) the classification system has to be mutually exclusive and exhaustive within its system [19].

With developing technology and a rising demand for electronically stored documents, first experiments with automatic classification (also called categorization) began. Automatic text classification, as it is in this case, is the process of an automatic assignment of documents to a pre-defined class. As already discussed in Chapter 2, pre-defined classes (also known as categories) are the result when building a controlled vocabulary, like a thesaurus or a taxonomy. The only difference now is that there should not be any expert user who files all documents under the given classes. Which would be a very time consuming task by the way.

Back in the 80's the first approach to automatically classify text documents came from the field of knowledge engineering and was based on a set of rules. Figure 3.1 shows an example.

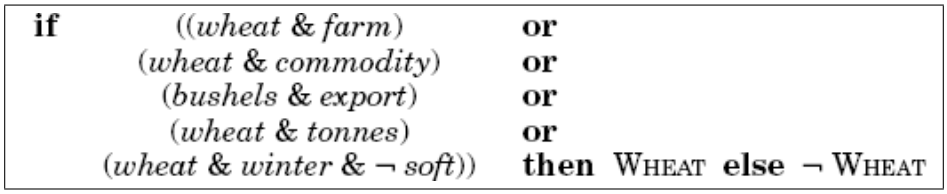

Figure 3.1: Rule-based classifier for the WHEAT category;

Nowadays the more convenient ML approach is used most of the times. There,

"a general inductive process (also known as the learner) automatically builds a classifier for a

category c_i by observing the characteristics of a set of documents manually classified under c_i or $\bar{c}_i$ by a domain expert [32]."

The advantage in contrast to the rule-based Knowledge Engineering method is, that there already are different implementations of classifiers, so that they only have to be trained. And it is easier to manually classify a small set of documents than to build and modify rules. No matter which classifier is used, the way of improving its quality is always the same.

3.1.1 Train and Test

When starting to classify text documents the used classifier is "empty". This means that it first has to learn which documents should be filed under the given categories. Therefore a so called training set is needed. A training set is part of an *Initial Corpus*, which is representative for all documents but smaller. All documents together are also known as corpus. After building the initial corpus, an expert user manually classifies documents, which should at least contain one positive and one negative example of each category. To train the classifier, the initial corpus is than divided into two parts. The test- and the training set. The test set is used by the learner to see which documents suit to which category (=indicative process). After the classifier is trained, the other part of the initial corpus is used to measure its effectiveness. Therefore the results of the automatic classification of the learner and those of the expert user will be compared. If the result is satisfying (how to measure this is part of section 3.5) the classifier can be used on the corpus. If the result is unsatisfying (e.g wrong categorized documents, algorithm is slow) the learner can be tuned or boosted to increase its effectiveness. Regarding the runtime, a computation is considered feasible if it can be done in polynomial time.

3.2 The Data Cloud

Computers can fulfill tasks a lot faster than humans, when looking for information. But with growing computing power also the amount of data has grown, and to process large documents quickly new search strategies are needed. Though a linear scan through documents for the requested word is still sufficient for single or a small amount of documents, it is inefficient for larger collections. To retrieve a result through a search task is a common problem for different kinds of media.

To be able to retrieve results quickly, the amount of data has to be reduced. When looking for similar images, a common way to reduce its size is to extract special features of the image. An appropriate feature

for example could be a vector representation of its edges or the average RGB-values. In text analysis a feature could be an index that maps words and their occurrences in the text. And again this feature could be represented by a vector. All features representing one document build the feature vector and all feature vectors of all documents form the data cloud. Once the data cloud contains comparable vectors - which is the case when every document in the data cloud is represented by the same features - we could use it to find similar content. How to build a data cloud when analyzing text documents is part of the next sections.

A simple way to index a small amount of documents or using only a few terms for indexing, is a *Binary Term Document Incidence matrix*. Here for example the columns represent the documents and the rows the terms. If a term *i* occurs in a document *j* than $a_{ij} = 1$ and when there is no occurrence of the term than $a_{ij} = 0$. A better and more common approach is to weight the terms. Where an entry in the matrix corresponds to the ”weight” of a term in the document. The weight of a term changes if it is more frequent in a document (=more indicative) or when it appears in many documents (=less indicative). After having built the matrix it is possible to pose boolean search queries. The simplest form of the matrix contains zeros and ones. When looking for a document which contains the terms ”information” and ”retrieval”, the matrix will first return the row vector for the term information which is: ”1001011” where the value one indicates a positive matching and zero a negative. Next it will look at the term retrieval which is ”0101101”. The last step is to perform a binary operation as the search query wants only those documents which contain both terms. Combining the retrieved vectors by and the result is: ”0001001”, which means that two documents contain the desired terms. With an increasing amount of documents, the size of matrix increases and eventually become too big to get results in an appropriate time. Therefore a better solution when having large document collections is to store a mapping from words to their locations in a document and to ignore those that don’t occur. This concept is also known under the name *Inverted Index* and ”is the most popular data structure in document retrieval [44]”. The inverted index can be divided in two parts. The dictionary and the postings list. The dictionary is a list of all the indexed terms, every single term points to the postings list which contains the document references.

Given the three sentences S_0 = “information retrieval is a science”, S_1 =“is this the information”, S_2 =“this is the science magazine” the inverted index would look like as depicted in table 3.1.

A term search for “information”, “science” and “is” would give $\{0, 1\} \cap \{0, 2\} \cap \{0, 1, 2\} = \{0\}$. To be able to profit from the inverted index it has to be built in advance. But to get good results it is necessary to do some pre-processing.

dictionary	→	*postinglist*
"information"	→	{0, 1}
"retrieval"	→	{0}
"is"	→	{0, 1, 2}
"a"	→	{0}
"science"	→	{0, 2}
"this"	→	{1, 2}
"the"	→	{1, 2}
"magazine"	→	{2}

Table 3.1: An inverted index three sentences

Major steps in inverted index construction are:

1. Collect the documents to be indexed

2. Tokenize the text

3. Linguistic pre-processing

4. Indexing

Once the documents are collected, the next step is to extract all words from them to build the dictionary. This can be done using a *Tokenizer*. A tokenizer chops text into tokens. Depending on how the tokenizer is configured, the gained tokens are more or less useful. Often the first step of a tokenizer is to transform the text to lower case. Then it tries to find the encoding and the language of the given document. Different languages need different tokenizers. The german language for example has many compounds whereas the english language often uses hyphenation (eg. don't) and in france the article changes if the following substantive begins with an vowel. This means that the language is related to the decision whether some term could be considered as a token or not. After it is clear in which language the documents are written, the tokenizer chops the document, based on predefined rules, into tokens. The rules can handle different problems like how do I know if this is a token or not (white spaces, punctuation), or how do I split words with a hyphenation in it. "Don't" for example has a few possibilities like: "do", "don", "do" "n't" or "dont". But which one is the right one? Another thing that has to be taken care of are names. When searching for the phrase "World War II" for example, a satisfying result will obviously will not be one with all documents containing the terms "world", "war" or "II". There are further possibilities to improve the index. Besides of a Tokenizer, the most common algorithms are Stemmer, Lemmatizer and POSTagger.

3.2.1 Stemmer, Lemmatizer and POSTagger

In most languages words can be related to other words or variants of it (morphology), which have similar semantic meaning. As for the dictionary, these words can be considered as equivalent. Stemming algorithms try to reduce those words to the root. Based on a set of rules, a *Stemmer* decides if a word can be reduced by chopping off the end or not. A widely known and effective stemming algorithm is Porter's algorithm[1]. Instead of chopping, a *Lemmatizer* performs a full "morphological analysis" and tries to correctly find all Lemmas, which is the canonical form of the word (eg. "walk" in "walking"). Although a Lemmatizer is more accurate it does not improve IR performance.

Sometimes it is necessary to mark up the words in a text as corresponding to a *Particular Part-of-Speech (POS)*. A POS can identify if a word is used as a noun, verb, pronoun, adjective, etc. For example: "They heard high pitched *cries* in the middle of the night." and "The baby "cries" all night long." According to its part-of-speech "cries" in the first sentence is acting as a noun, whereas in the second sentence "cries" is used as a verb. *POSTagging* algorithms try to determine the POS Tag for a specific instance of a word.[2]

3.2.2 Term weighting

Term weighting is used to improve the accuracy of the retrieved documents and to act as a ranking mechanism. In practice, an often used weight is the *Term Frequency-Inverse Document Frequency* $(tf - idf)$ (=equation 3.1). This weight gives information about the importance of a word in a document to all documents (=collection or corpus). A high weight in tf-idf is reached by a high term and a low document frequency of the term in the whole collection of documents. This has the effect, that common terms will be filtered out by assigning it a low weight.

$$tf - idf = tf \bullet idf \tag{3.1}$$

The *Term frequency* (tf) (=equation 3.2) in a given document is the number of occurrences of a term t in a document d. Where n_t is the number of occurrences of the considered term, and the denominator is the number of occurrences of all terms.

$$tf_t = \frac{n_t}{\sum_k n_k} \tag{3.2}$$

Because a high term frequency is not very significant (all terms are considered equally important) a more

[1] See http://www.comp.lancs.ac.uk/computing/research/stemming/general/porter.htm
[2] See http://www.cs.ualberta.ca/ lindek/650/Slides/POSTagging.ppt

meaningful type of weight-mechanism is often used. With inverse document frequency it is possible to state how important a word is. The *inverse document frequency (idf)* (=equation 3.3) is a measure of the general importance of the term (obtained by dividing the number of all documents by the number of documents containing the term, and then taking the logarithm of that quotient)[3].

$$idf_t = log\frac{N}{df_t} \tag{3.3}$$

Where N is the total number of documents in the corpus and df_t, the number of documents where term t appears.

idf_t, d is:

1. highest, when t occurs many times within a small number of documents
2. lower, when the term occurs fewer times in a document, or occurs in many documents
3. lowest, when the term occurs in virtually all documents

A weight could either be binary, where 0 states an absence and 1 a presence of the feature, or non-binary. Documents from the origin corpus are now represented by features and their matrix. This enables a learning algorithm (=classifier) to categorize documents much quicker than it would have been without. Knowing from section 3.1, there are many different existing classifiers. Some of them and the ideas behind them, will be illustrated in the next section.

3.2.3 Indexing

Having a weight mechanism for terms, it is possible to edit the documents and its terms as vectors. In a given document space where every document d_i is represented by its terms and their weights, each document can also be written as a t-dimensional vector, where t is the number of the terms. $D_i = \{d_{i1}, d_{i2}, ..., d_{it}\}$ where d_{ij} representing the weight of the jth term [31]. The *Vector Space Model (VSM)* opens the door to other information retrieval operations like clustering, document classification or similarity searching.

Given the vector representation of terms (bag of words), there is now a mean to compute similarity between two documents. One possibility to measure similarity between two vectors is to calculate its difference, where a low distance would indicate a high similarity. A problem with this approach is, that documents with different lengths can seem to be completely different, although they are rather similar. Therefore, document

[3] http://en.wikipedia.org/wiki/Tf-idf

length is crucial when calculating similarity. A convenient way of ranking documents - i.e., by measuring how close a document's vector is to a query's vector - is the *cosine similarity* (=equation 3.4).

> "As the angle between the vectors shortens, the cosine angle approaches 1, meaning that the two vectors are getting closer, meaning that the similarity of whatever is represented by the vectors increases [11]."

With cosine similarity documents can be compared regardless of their length.

$$sim(d_1, d_2) = cosine\Theta = \frac{\vec{V}(d_1) \bullet \vec{V}(d_2)}{\left|\vec{V}(d_1)\right| \left|\vec{V}(d_2)\right|} \tag{3.4}$$

The computed VSM-Matrix now consists of meaningful text units (=weighted terms without function words) and is the so called indexed representation of the corpus, which is all documents of interest. To further improve the quality of the index and also to prevent overfitting a *Dimension Reduction (DR)* should be performed before the use of the classifier. There are two ways of DR, namely the local and the global. Where the first uses different sets of documents for different categories and the latter not. DR also distinguishes in the way of handling the terms of the corpus.

- DR by term selection: the reduced result is a subset of the terms of the initial corpus
- DR by term extraction: the resulting terms are of a different type than the initial corpus (e.g. phrases instead of terms)

Both methods use different techniques to improve the index. Term selection for instance may use *Document Frequency*, which selects terms that occur in the highest number of documents. Other forms of term selection, which are more complex, are *Mutual Information*, *Chi-Square* or *Information Gain*.

While *Term Selection* only reduces the terms of the initial corpus, *Term Extraction* tries to group terms with a semantic relation, so that one can use groups instead of using the terms [32]. Algorithms which use term selection for DR, are *Clustering algorithms* or *Latent Semantic Indexing (LSI)* mechanisms.

3.2.4 Clustering

Clustering is the most common form of *Unsupervised Classification*. An unsupervised classifier does not have a training or test set. Which means that the classifier does not know how many categories exist and how documents may be related to them. Clustering is to find a set of documents (=cluster) based on a measure of similarity that is clearly different from others but internally coherent. If documents can be an instance of only

one cluster, it is called hard assignment and is done with a partition algorithm. Soft assignment or fractional membership is, when documents can be a distribution over all clusters (=non-partional).

> "Cluster hypothesis: documents in the same cluster behave similarly with respect to relevance with information needs [25]"

The hypothesis states, that if a requested document of a cluster fulfills the information need of a user, the rest of the documents in the cluster do that as well, or in other words, may also be relevant to it. Based on this, several navigation structures have developed.

scatter-gather "clusters documents into topics and presents descriptive textual summaries to the user. The summaries consist of topical terms that characterize each cluster generally, and a number of typical titles that sample the contents of the cluster. Informed by the summaries, the user may select clusters, forming a subcollection, for iterative examination. The clustering and reclustering is done on-the-fly, so that different topics are seen depending on the subcollection clustered [18]"

hierarchical tree a hierarchic presentation of category labels where the user can browse through

To build clusters out of a vector representation, a common practice is to use a *k-Means clustering algorithm*. K-means forms documents to an unknown number of K clusters under an objective function, which minimizes or maximizes the average distance (=similarity) between a computed center (also known as centroid) an the surrounding documents. Here, two problems arise. How to find an appropriate value K and what is a good centroid. With k-means both problems can be solved. To prove whether a centroid is good or not, k-means uses the *Residual Sum of Squares (RSS)-function* as its objective function. RSS is a function to measure the squared distance of documents from their centroid and is given by equation 3.5. By minimizing it, the centroids moves iteratively from a randomly chosen position at the beginning to a new point where the RSS is lower than before. That means, simply said, that the algorithm finds automatically the point with the most documents around it.

$$\vec{\mu}(w) = \frac{1}{|w|} \sum_{\vec{x} \in w} \vec{x} \tag{3.5}$$

$$RSS_k = \sum_{\vec{x} \in w_k} \left\| \vec{x} - \vec{\mu}(w_k) \right\|^2 = \sum_{x \in w_k} \sum_{m=1}^{M} (x_m - \mu_m(w_k))^2 \tag{3.6}$$

The algorithm for k-means starts with determining the number of clusters K, and assuming the centroid of this clusters. As centroids are randomly chosen and under circumstances could lead to overlapping clusters,

good starting points are to place them farthest away from each other [10]. Next, the algorithm determines the distance of each object to the centroids and assigns it to the nearest centroids. Then it recomputes the centroids in order to minimize the RSS. This is done as long as a stopping criterion is met (e.g. the centroids do not move anymore, fixed number of iterations, etc.) as it is depicted in figure 3.2. But there is still the problem of finding a good value K. As $RSS_{min}(K)$ is a monotonically decreasing function, a heuristic method to find an appropriate value K is to increase K stepwise and look at the RSS_{min} values of the curve. Those points where a significant decrease could be recognized (also called "knees" in the curve) are possible values for K. Another possibility could be, to simply estimate the possible numbers of categories, which is feasible if the corpus is well-known.

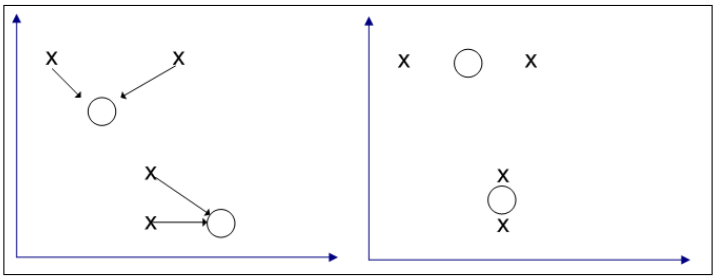

Figure 3.2: After one iteration where k-means first assigns documents to the nearest centroid, they move to fulfill minimum RSS

The aim of the objective function is to maximize intra-cluster similarity (similar documents within a cluster) and minimize inter-cluster similarity (documents from other clusters are dissimilar). This is also known as an *Internal Criterion*. To see how well the algorithm classifies, it can be tested against an external "gold standard". This standard defines a set of classes for evaluation benchmarking. There are several *External Criterions* to measure quality, like *Purity*, *Entropy*, *Rand index*, *Recall, Precision, F-measure* and more. Purity for example is defined as the ratio of the number of documents in the majority class to the total number of objects. That means that purity states classification accuracy under the assumption that all objects of a cluster are classified to be members of the dominant class for that cluster [34].

3.3 Classifier

When using classifiers, different types are available. Very well known are the probabilistic classifiers. But there are a lot of different approaches to to build "learners". There are *Neural Networks (NN)*, *Decision Based Classifiers*, the *Support Vector Machine* and some more. Regardless which classifier will be applied, the procedure when using *Supervised Learning* is always the same. This section will give an overview of

some of these algorithms.

3.3.1 Probabilistic Classifiers

As its name says, probabilistic classifiers file documents under classes on the basis of probabilities. Therefore they are also known as statistical classification algorithms. Probabilistic classifiers use the *Bayesian Theorem [3]*.

When classifying documents, the problem is to find a function that assigns a value v to $\langle d_j, c_i \rangle \in DxC$. In the case of a statistical classifier, v denotes the probability under which d_j belongs to c_i. According to Bayes the problem could is given as equation 3.7

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{3.7}$$

The left side of the equation is also called the a posterior probability. The numerator consists of the conditional probability $P(A|B)$ and the prior probability $P(B)$. The denominator is used for normalization. Conditional probability means that it denotes the probability under the condition that A happens after B has already happened.

The following example should clarify this. Given a training set where documents either could be about "sport" or "art" a distribution could look like figure 3.3.

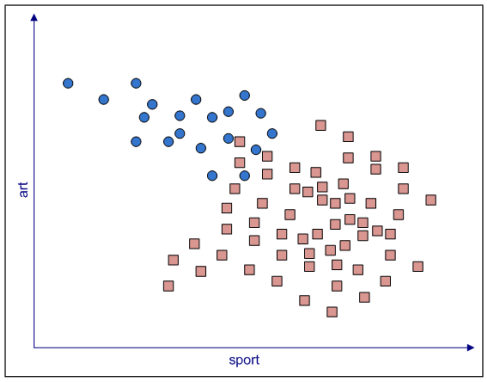

Figure 3.3: possible distribution for "sport" and "art"-documents

Where the "squares" represent documents that belong to "sport", and the "circles" are for those documents which are related to "art". Given this illustration it is possible to determine the prior probability. Since the total number of documents is eighty the prior probability of each category is:

	F1	*F2*	*F3*	*F4*
$P(x = i\|w_1)$	0,1	0,1	0,35	0,45
$P(x = i\|w_2)$	0,4	0,3	0,2	0,1

Table 3.2: Probability distribution using a numerical value recognition feature. Where $w_1 = SPORT$ and $w_2 = ART$.

$$\text{prior probability for SPORT} = \frac{\text{number of sport objects}}{\text{total number of objects}} \tag{3.8}$$

$$\text{prior probability for ART} = \frac{\text{number of art objects}}{\text{total number of objects}} \tag{3.9}$$

Which is:

$$P(S) = \frac{60}{80} = 0,75 P(A) = \frac{20}{80} = 0,25 \tag{3.10}$$

Given only the prior probability, a new document would be, with a probability of 75 %, a document of the category sport. Filing every new document under sport would therfore lead to a failure quote of 25 % which is too high to use it in praxis. So what is needed now, is an additional means to classify the documents - a so called feature. Other approaches are to calculate the maximum likelihood as it is discussed in section 3.2.4. Using a feature, documents can be further distinguished. Supposing that the documents are provided by a newswire and only contain short newsflashes, one possible feature could be the recognition of numerical values. As a newsflash uses them often to inform about the final score, or the ranking from first to third place. In articles about art there may often be a date which also can be recognized by the feature. This feature can than be used to predict a category more precisely. A possible probability distribution is shown in table 3.2.

The feature splits the occurrences of numerical values into four groups where group *F1* and *F2* have more “date” and less “score” values (=good probability for an art document) and group *F3* and *F4* have good chances to be a sport document as they contain more “score” values. $P(x = i|w_j)$ shows the conditional probability for each class. Together with the prior probability, it is now possible to compute the a posterior probability. Since probabilities should have a sum of 1, over all possible hypotheses *Normalization* can be used to achieve that.

Normalization is done in the following way:

$$P(x = i) = \sum_{j=1}^{c} P(x = i|w_j) P(w_j) \tag{3.11}$$

Using the results from the preceding calculations Table 3.3 shows the a posterior probability. Considering

	1	*2*
$P(w = i\|x = i)$	0,27	*0,73*
$P(w = i\|x = i)$	0,33	*0,67*
$P(w = i\|x = i)$	*0,72*	0,28
$P(w = i\|x = i)$	*0,87*	0,13

Table 3.3: The final a posterior probability. The emphasized values are those chosen by the *Bayes Decision Rule*

Bayes Decision Rule those class with the highest probability will be chosen for classification. In the case of the above example a document of group F1 or F2 would be classified under category 2, which is art and documents having a feature recognition for group F3 or F4 are filed under category 1, which is sport. A global failure rate computation will now be 24 % which is a bit lower then when using only the prior probability but still to high. In automatic text categorization the common use of the *Bayes Classifier* is therefore slightly different.

When using text documents, it is often hard to determine the conditional probability and to calculate discrete values, as used above. To get good results anyway the Bayes Classifier uses a trick. It assumes that each attribute in a text document is conditional independent. This is hardly ever true for words in documents, but nevertheless using the *Conditional independence assumption* leads to surprisingly good classification decisions [25]. Based on that assumption the Bayes Classifier is also called *Naive-* or *Idiot Bayes Classifier*. When using the *Conditional Independence Assumption*, every feature corresponds to a word. Studies showed that *Naive Bayesian learning* gives better test set accuracy than many other known methods. Also the computational time to train a classifier is with a linear duration very efficiently [14].

3.4 Vector space model

A document representation for further processing, is normally given by a vector. This vector sometimes is binary (e.g. a simple word count), as it is with the Naive Bayes Classifier, or it contains real valued components for each term. Real valued components can be weighted terms (e.g. tf-idf weight). A weight therefore is for how much a term t_k contributes to the semantics of d_j. The real valued representation of terms is also known as the *Vector Space Model (VSM)*. A basic assumption here is the contiguity hypothesis, which claims that documents in the same class form a contiguous region and regions of different classes do not overlap. A contiguous region is formed by the "features" of the document. Good features for specific classes will lead to higher values on terms which are representative for this class. Terms like "soccer", "ultimate frisbee", "basketball" or "player" will have higher values for the class "SPORT". Therefore, the use of the right feature, the

document's representation is crucial and responsible for good or bad classification results, when creating the index. A vector space should always be weighted and normalized. Having a vector space model, the next step is to classify the documents in it. Most classifiers today are linear, which means that the decision boundary is given by a line or in higher dimensions by a hyperplane. A decision boundary separates the classes from each other. In other words, linear classifiers rely on a simple linear combination of the features. A linear classifier for category c_i is a vector $\vec{c}_i = \langle w_{1i}, ..., w_{|T|i} \rangle$ belonging to the same $|T|$-dimensional space in which documents are also represented.

Non-linear classifiers are more complex and use more additional "layers" of units, which in Text Classification usually represent higher-order interaction between terms [32]. Simply, the decision boundary in a two-dimensional space is a curve (=non-linear). The k-NN (=k-nearest neighbor) is a non-linear classifier for example, while Rocchio or Naive Bayes are examples for a linear classification algorithm. Linear classification is binary, which means that the decision boundary only separates between those documents who fit into c_i and those who do not, which is a classification under $\overline{c}_i$.

When using linear classifiers, decision boundaries are lines in a two-dimensional vector space and will become hyperplanes with higher dimensions. A hyperplane also divides documents into two classes c_i' and $\overline{c}_i$. But having a hyperplane does not necessarily mean that it is a good classifier. Which is, when the hyperplane misclassifies a lot of documents. The aim therefore is to find a hyperplane which has also a high classification effectiveness on the test set. There are two ways of finding this hyperplane:

1. methods that rely on the global distribution of documents. (Naive Bayes, Rocchio and linear regression)
2. methods that give more weight to documents close to the decision boundary. (Support Vector Machines, logistic regression)

Chapter 3.4.1 will discuss the *Support Vector Machine (SVM)* in more detail.

Non-linear classifiers use more complex methods to decide under which category c_i a document d_j should be filed. But although this methods are more complex the result in comparison to a linear classifier are not better. The reason therefore is a tradeoff between *bias* and *variance*. Because of this tradeoff there is no universally optimal learning method. Bias is the squared difference between the probability of a document d being in a corpus c. Bias is large if the learning method produces classifiers that are consistently wrong, or if different training sets cause errors on different documents. Of a bias could be thought as a kind of domain knowledge, which was built into a classifier. If a human expert user knows in advance, that the true boundary

between two classes is linear, then a linear learning method is more likely to succeed than a nonlinear method. Variance is the variation of the prediction of the learned classifier or between the estimated and the true class. Variance is large if different training sets give rise to different classifiers.

Non-linear algorithms have low bias and high variance, while linear ones tend to have high bias and low variance. Figure 3.4 graphically displays the tradeoff between a linear and a non-linear separator. Where the bias of the linear function f_1 is small but therefore the variance is high. For further details refer to [25].

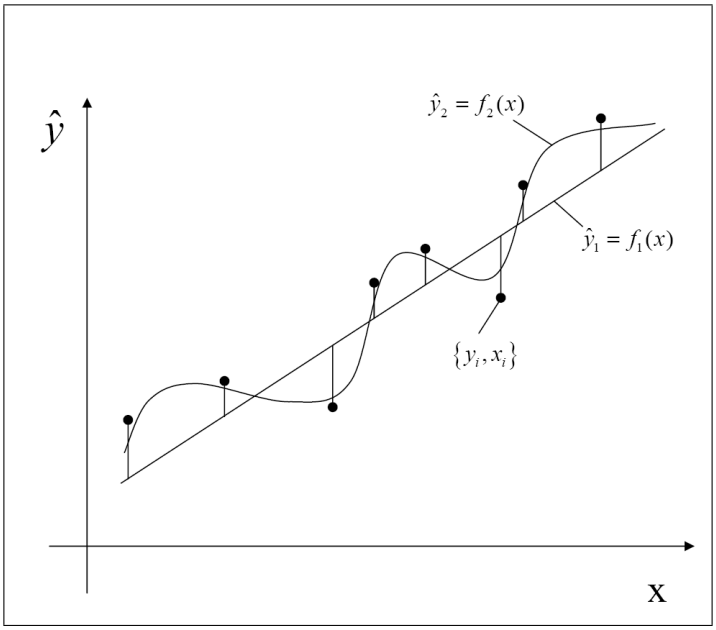

Figure 3.4: The Bias-Variance tradeoff between a linear and a non-linear separator

As it makes nearly no difference if the classifier is linear or non-linear, the most used classifiers are linear ones, because of the easier implementation. But what if a set of words has more than two different classes to classify. If $J > 2$ the mutual exclusiveness determines the further processing. If documents can belong to one, more than one or no class (=not mutual exclusive) the classification is also called any-of, multilabel or multivalue. Solving an any-of classification is quite simple. The first step is to train a classifier for each class and apply each classifier separately to the test document. A classifier's decision has no influence on the decisions of the others. This way, documents can be classified under more than one class.

If classes are mutually exclusive, each document must belong to exactly one of the classes. This form of classification is also called one-of, multinomial or multiclass classification. One-of classification is less common than any-of classification because often text documents belong to more than one class. Anyway,

one-of assumptions are often made, even if classes are not really mutually exclusive. When having more than two classes, the number of the distinct regions given by decision boundaries (hyperplanes) is not equal to the number of classes. Due to this, a combination method when using linear classifiers for one-of classification has to be used. The easiest method to do so is to rank the classes and then select the top-ranked. Ranking can be achieved by similarity measure (e.g. euclidean distance), probability of class membership, or weighting. The algorithm is as follows:

- Build a classifier for each class, where the training set consists of the set of documents in the class (positive labels) and its complement (negative labels).
- Given the test document, apply each classifier separately.
- Assign the document to the class with
  - the maximum score,
  - the maximum confidence value,
  - or the maximum probability

3.4.1 Support Vector Machine

Besides the statistical methods for classifying data, other classifying techniques are used for solving the problem of relating a data-item x_i to a class c_i. One of these techniques is linear classification. A special kind of a linear classifier is the *Support Vector Machine (SVM)*. In recent years, *SVMs* witnessed a boom because of its good generalization rate. They have been successfully used for handwritten digit recognition, face detection in images, text categorization and more. There, the *Support Vector Machines* classification effectiveness matches, or might even be significantly is better than that of competing methods.

A linear classifier tries to separate data through a decision boundary. Origin algorithms tried to find any decision boundary and stopped when they found one. A *SVM* instead tries to find the maximal separating one. As SVMs are trained on the basis of a training set (=supervised classifier), the “machine” will achieve the best generalization performance when it finds the right balance between the accuracy on a particular training set and its ability to learn training sets without errors. Since the invention by Vladimir Vapnik in the late seventies, support vector machines have improved their performance and are now a state-of-the-art classifier. Although the basic ideas are relatively simple to understand, a more detailed insight needs some mathematical skills. This introduction will not cover all the details. A good point to start with is Burges’ tutorial [4] or Vapnik’s book [37].

A SVM can be used to classify two-class linearly separable data, nonlinear data as well as non separable data points. It has to be noticed that years before the SVM was invented, a similar idea existed. The so called perceptron. It was also a linear classifier which looked very promising. But after it has been proven that it is not applicable on nonlinear separable data it was detached by neural networks.

Linearly separable data

The technique of every SVM is to map input vectors to a higher dimensional space, where a maximal separating hyperplane is constructed. That means that in $\mathbb{R}^3$ the hyperplane $h = 4$, or more general in $\mathbb{R}^N : h = N+1$. A maximal separating hyperplane describes a decision surface that is maximally far away from any data points, this is also called the margin. Small decision surfaces increase the error rate because data points near a decision boundary have a higher probability of misclassification. Broader surfaces (a.k.a. fat margins) therefore lead to fewer misclassification what decreases the generalization error. Large-margin classifiers like SVM use a measure of distance (e.g. euclidean distance) for finding those hyperplanes which have a maximal distance between the data points. The data points, which in Text Classification are documents, are of the form $\{(x_1, c_1), ..., (x_n, c_n)\}$ where $c_i \in \{1, -1\}$. Where x_i is a real vector and c_i denotes whether this vector belongs to c_i or not. The maximum decision surface is determined by only a small set of data points which are also known as the *Support Vectors*.

The optimal dividing hyperplane is given by $\langle w, x\rangle + b = 0$, where w points are perpendicular to the *Canonical Hyperplanes*. The maximum margin is then given by the parallel hyperplanes and the support vectors. The maximum margin is found, when the distance (δ) between the support vectors and the separating hyperplanes is highest. Canonical hyperplanes are so chosen that $\langle w, x\rangle + b = -1$ for the border of the negative samples and $\langle w, x\rangle + b = 1$ for the border of the positive samples. The distance from a point to the separating hyperplane is calculated by 3.12.

$$d(w, b; x) = \frac{|\langle w, x\rangle + b|}{||w||} \tag{3.12}$$

For given support vectors x_1 and x_2 a maximization of $\delta = 1/||w||$ is equivalent with a minimization of $||w||^2$ which is a quadratic optimization problem that can be solved using a Lagrange function. Figure 3.5 and 3.6 will illustrate this.

Non separable data

In the case of non separable data, the common approach is to introduce *Slack Variables (ξ)*, which tolerate misclassification to a certain level. Data points are not clearly separable if either the used features do not

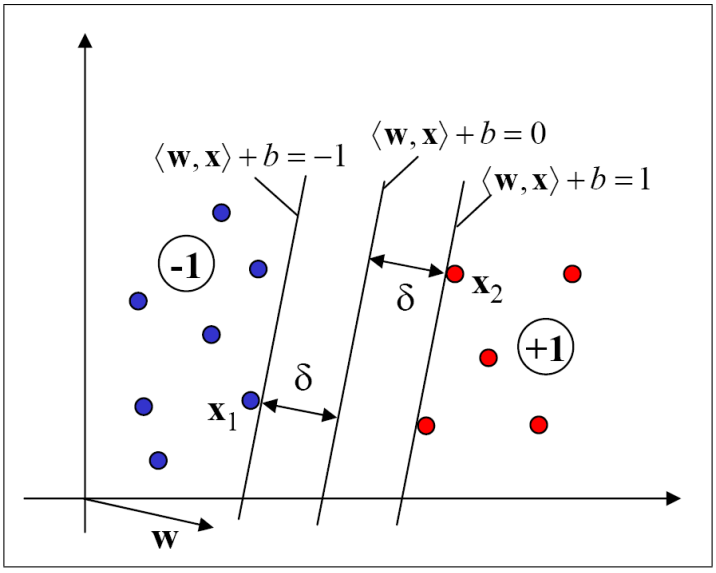

Figure 3.5: Finding a maximum margin by SVM

$$2\delta = \left| \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_2 - \mathbf{x}_1) \right\rangle \right| = \frac{1}{\|\mathbf{w}\|} \left| \underbrace{\langle \mathbf{w}, \mathbf{x}_2 \rangle}_{(1-b)} - \underbrace{\langle \mathbf{w}, \mathbf{x}_1 \rangle}_{-(1+b)} \right| = \frac{2}{\|\mathbf{w}\|}$$

$$\Rightarrow \quad \boxed{\delta = 1/\|\mathbf{w}\|}$$

Figure 3.6: Maximizing δ

allow a good separation or if some data points are noisy. Slack variables comply with the distance of how far away data points are from the optimal hyperplane. Which lead to a new optimization problem that tries to find a margin where the slack variables are minimal.

$$\|w\| + C \sum_{l=1} \xi \rightarrow min \tag{3.13}$$

Where C controls the error classification rate. A high value of C means that there are many slack variables which lead to a high "punishment", whereas a little C indicates few slack variables or a small error rate. $\sum \xi_i$ is the maximal error rate. Again there is a trade off between how large the margin can be and those points which are misclassified around the margins.

Figure 3.7 gives an example of an SVM with slack variables.

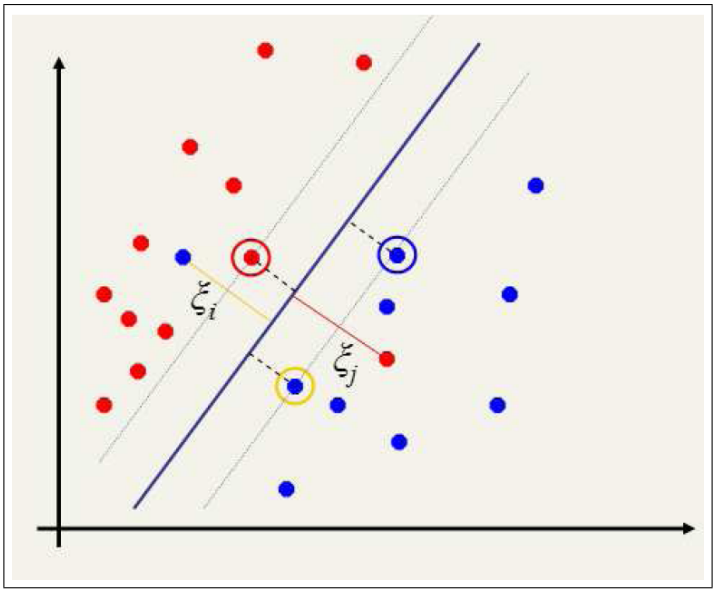

Figure 3.7: Classification using slack variables

3.5 Evaluation in Information Retrieval

Evaluation in IR can be done at several levels:

- Processing: Time and space efficiency
- Search: Effectiveness of results
- System: Satisfaction of the user

While all three levels play an important role when designing and testing a retrieval engine, the effectiveness of the system can be considered as its most important feature. Effectiveness relates to the classifier and measures its ability to make the right classification decisions. A right decision when using classification could either be the right relation of a document to a category (=ML approach), or to retrieve the right search result to a users search request (=IR approach). In principle both approaches are equal and deal with the relevancy of a document to a category (where the search result is a dynamically created category). The classifiers response should in the best case contain all relevant and zero non-relevant documents. As with statistical measures, a document could be a *True Positive (tp)*, which is a relevant and correctly under c_i classified, document or a *True Negative (tn)*, which is a non-relevant, not under c_i classified document. An incorrectly under c_i classified, not-relevant document is called a *False Positive (fp)*, contrary to a *False Negative* (fn), which is an incorrectly not under c_i classified, relevant document. Table 3.5 summarizes this.

	Relevant	*Not relevant*
Retrieved	true positives (tp)	false positives (fp)
Not retrieved	false negatives (fn)	true negatives (tn)

The most used values for measuring efficiency in information retrieval are *Recall (R)* and *Precision (P)*. Recall is the fraction of relevant documents that are retrieved, or a measure of the completeness of the returned documents, which is derived by equation 3.14. Precision is the fraction of retrieved documents that are relevant, which means, that it is a value for the usefulness of the result. Precision can be calculated by using equation 3.15. Obviously, there is a trade-off between those two values. A high recall, which is 100 % when returning all documents, can lead to low precision. Search engines on the Web often have a good recall but low precision, which means that they offer a lot of documents to a request but only a few of them are relevant to the information need of the user.

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant}) \tag{3.14}$$

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved}) \tag{3.15}$$

When using ranked retrieval, the trade off can be used to address different groups of users. Ranked retrieval is, when those search results with high precision are displayed first. The information need of a simple web searcher is probably satisfied with the first ten results, while a user who needs more detailed information is not. Therefore precision gives way to recall when displaying the results. Figure 3.8 shows this trade off graphically. Besides recall and precision and also oft-used measure in IR is the *F-Measure*, which combines recall and precision with an equal weight.

$$\text{accuracy} = \frac{tp + tn)}{tp + fp + fn + tn} \tag{3.16}$$

In machine learning, another used method to measure for a classifiers quality is *Accuracy* which is given by equation 3.16 and is the fraction of the correct classifications. The problem with this measure in information retrieval is, that almost all documents (ca. 99.9 %) are non-relevant to the search request. A high accurate classifier would therefore classify documents as non relevant or in the case of lowering it, in order to show some results to the user, will have many false positives.

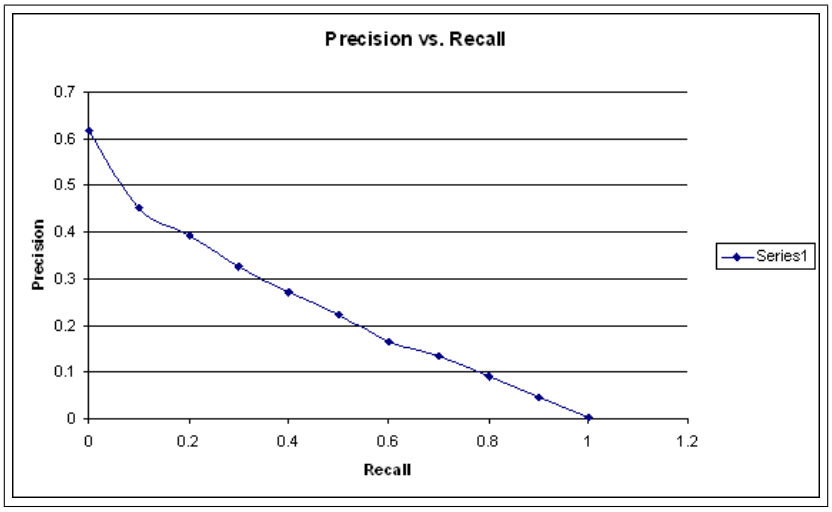

Figure 3.8: Recall-Precison Graph

Chapter 4

Approach

As mentioned in chapter 1, this thesis tries to find a satisfying solution for enriching metadata records in a semi-automatic way. A metadata record is considered as enriched, if it is ontology-aware. That means that it contains a relation to a concept in a controlled vocabulary, which describes the semantics of the record best. In the field of digital libraries such a controlled vocabulary often already exists. Controlled vocabularies or Ontologies, are defined by expert users and try to depict a fitting universe of discourse. In help with *Semantic Web Technologies* enriched metadata records can be used to improve search capabilities and discovery mechanisms. As a manual enrichment process by an expert user would be too time consuming and tiresome, the goal of this approach is to find a semi-automatic way, which will have the same results as it would have been by an expert user. Therefore the previously discussed topics are needed. The problem in detail and a possible solution is part of this chapter.

4.1 The problem

Institutions nowadays have large metadata collections and their own vocabularies, which are defined in their own standard. As older standards are not machine-processable and interoperable the benefit of such metadata records is very restricted. Search and discovery services are still based on simple string-match algorithms. To exploit the full power of *Semantic Web Technology* the data has to be transformed. A transformation has the advantage, that the new records will be machine processable, and in combination with an Ontology, they will be able to depict formal semantics, which in turn increases the scope and possibilities of search engines. Also the data is more flexible, which is important, when the related Ontology changes or will be replaced. From a syntactical point of view such a transformation is easy to proceed. The more difficult part is to make the new

metadata records ontology-aware, so that all capabilities of the Semantic Web can be exploited. This process, which is also known as the enrichment step, needs to be performed in an efficient way. Manual enrichment of all metadata records would be a tedious process for the end-user.

An efficient approach is to use the power of machine-learning techniques, which can solve the enrichment task after a short training period by the end-user, automatically. Thence this step can be considered as semi-automatic because of the learning process of the algorithm, which is when an end-user manually relates exemplary records to concepts.

4.2 The solution

When enriching metadata records the first questions is, which concepts are available and how are they related? Remembering the design goals of section 2, concepts can be represented using ontologies. Ontologies are defined by an expert user, who tries to cover the whole domain of interest, with suitable concepts and relations. The expert user also has to decide which level of semantic expressiveness is needed. Sometimes a controlled vocabulary is sufficient and sometimes a more detailed ontology, like a thesaurus, is required.

Having found a convenient representation, and considering that the existing legacy metadata records have already been syntactically transformed from their proprietary format into a more flexible one, the record is ready to be enriched. This is where the field of machine learning comes into play. As discussed in chapter 3, text recognition needs a unique representation to be able to compare the underlying documents. For that reason the existing metadata records need a unique representation, like a $tf - idf$-matrix for example. A $tf - idf$-matrix, represents a documents terms in form of weights, which is a value for the expressiveness of each term. To reduce the number of irrelevant terms in such a matrix, the records have to be cleaned by the means already discussed (e.g. Stemmer, Tokenizer, etc.). Having a cleaned $tf - idf$-matrix, which represents all metadata records in an equal way, a machine-learning algorithm can use this data to classify it. Classification, which is nothing else than the relation of a record to a fitting concept, can be done in two ways. The unsupervised, this is when no end-user trains the algorithm, or the supervised one. While in the first case an unsupervised classification could only lead to a raw grouping of all records, the supervised classification should have, after a small training period, nearly the same results as they would have been when classification is done by an end-user.

4.2.1 Unsupervised Classification

As a pre-process step, it is possible to group similar records with the help of a clustering algorithm. This can be done easily, because no additional information except the $tf - idf$-matrix is required. The downside here is, that the $tf - idf$-matrix now contains a lot of features, every remaining word after stemming and filtering, where most of them contain the value "0" because of its global unimportance, what leads to poor results.

4.2.2 Supervised Classification

The step of supervised classification is a bit more complicated. Supervised means that a classifier is created on the basis of a training set which contains related classes for each record. Before the training of the classifier can begin, the $tf - idf$-matrix has to be extended with the corresponding concepts. As the prototype will show, a possible way to do that is via a graphical drag and drop interaction. Similar to the creation of the training set, is the creation of the test set, which is also important to be able to say something about the classifiers performance. The test set should contain different records than the training set because otherwise the classifier would return unusually good results. Having both training and test set, the classifier is prepared to be trained and tested. But in order to do that, a small adjustment of the two $tf - idf$-matrices is needed. A $tf - idf$-matrix consists of a term representation of all underlying documents, where each term can be seen as a feature. After training the classifier, a unique matrix, because of the terms of the training set, would be the result. When trying to apply this classifier on other metadata records, the $tf - idf$-matrix for this records would be different than that from the training set. Hence, the two matrices have to be adapted. To convert the two matrices, in a way that a classifier can use them for training and automatic classification, one has two possibilities. The first would be to use only those features (=terms or attributes) which both matrices have in common. A second way would be to delete the redundant attributes of the "test" matrix and insert the missing features of the "original" with a value of zero.

Once the matrices are prepared, a machine learning algorithm can be used to train a classifier which will later classify additional records. Good classifiers, among others, are the support vector machine or the bayes classifier. Before applying the trained classifier to the remaining records, the expert user should have a look on its classification results. To do so, he will test it on the previously built test set. If the classification results are satisfying, which is when about 90 percent of all records are classified correct, the classifier can be used to work off the other records. Normally, automatic classification is the prediction of the belonging class on the basis of the features. In this special case, a prediction is synonymous with the enriching of the metadata record by the predicted class.

Regardless of the automatic classification, metadata records could also be enriched by the implementation of logical rules. Where a simple rule could trigger if a specific word occurs. A decision rule classifies a document d_j under a category c_i on the occurrence of a special term x. It can therefore be seen as an if-then-statement of the form: "if d_j contains x then relate d_j to c_i". Each category can have multiple rules. A decision rule will always relate the document to the specified category no matter what the automatic classifier suggests. How such a rule could look like and increase the enrichment result will be shown in chapter 5.

Having reached the state where all records are classified, the expert user should have one last chance to view the result before he finally writes the new, enriched metadata record to the disc or database. After the final enrichment step, all records are ontology aware, which means that they have an additional element, that points to the concept that describes its content best.

To get a better understanding of the whole process figure 4.1 illustrates the workflow.

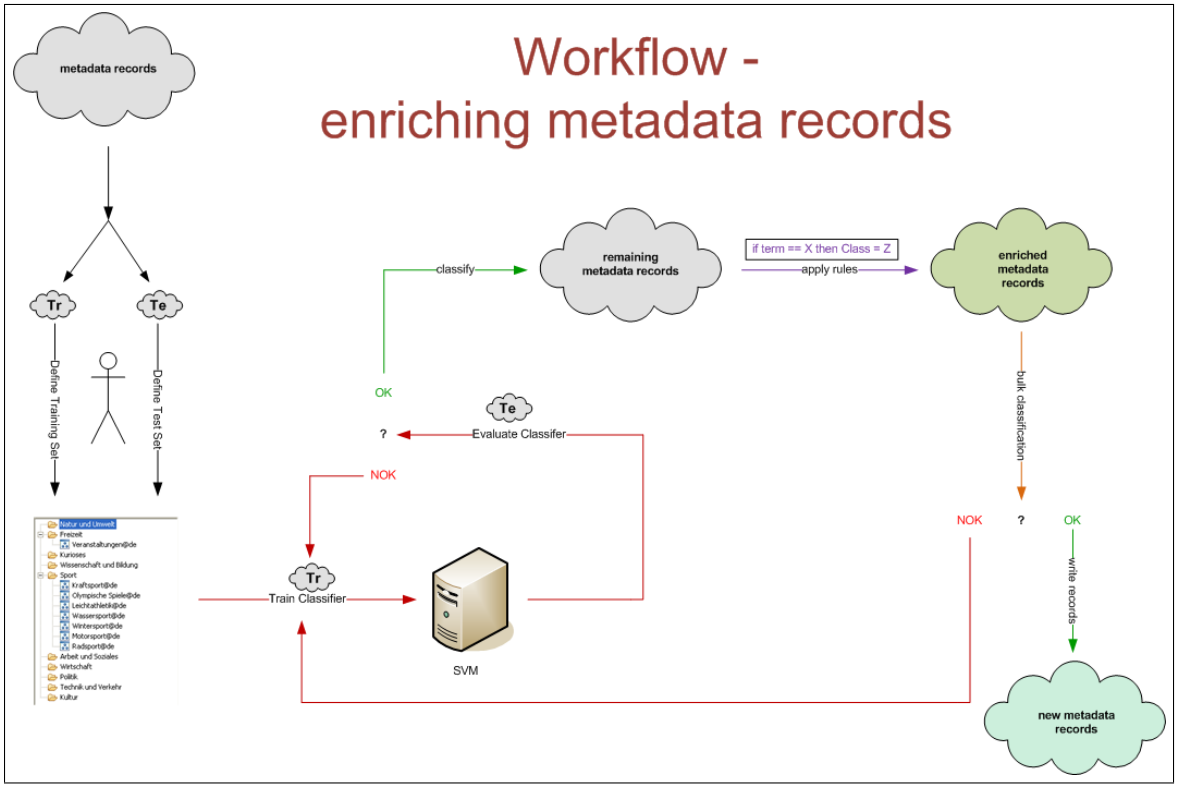

Figure 4.1: Workflow when processing metadata records

Chapter 5

Prototypical Implementation

Chapter 4 gave a general overview of the possibility to enrich metadata records. This chapter shows a concrete implementation and its prototype. The institution, for which the prototype has been designed, has an image database, where each image is described by a metadata record, which in our case is exposed in RDF. The elements give information about the photographer, the date, the type of photograph, the title, some descriptive key words and other things. It also contains an unique identifier. Listing 5.1 shows an exemplary metadata record.

A record contains several elements, which describe the underlying content digest, which in this case is an image. For the description the institution uses a pre-defined set of elements, that can later be used by the expert user and the machine learning algorithm, for classification. Normally an expert user would create the concepts for the thesaurus by examining the images and the ways described in section 2.4. In this special case the metadata records already have a classification which can be used instead. Listing 5.2 shows an example of such an element which will be discussed in more detail in chapter 6. Based on this elements it was possible to build a thesaurus without further considerations. As it is also part of the Case-Study, this brings a few problems with it. One of the most important is, that the concepts, when using the pre-defined elements, are not mutually exclusive. But mutual exclusiveness is a crucial factor when using an automatic classifier. Nevertheless it is possible to build a new thesaurus with new concepts, regardless of the "Klassifikation"-element.

Listing 5.1: Example of a metadata record

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3                                    xmlns:onb="http://bricks.org/ontologies
```

```
                                        /onb/1.0#"
4                                     xmlns="http://bricks.org/ontologies/onb
                                        /1.0#">
5   <onb:Bild rdf:about="
        oai:9031205d45c56d529d3f7eb46b8b3342877162fa:uuid:uuid:179">
6     <onb:Autor>Hilscher, Albert</onb:Autor>
7     <onb:Beschreibung>Er\"offnung␣der␣ersten␣Ausspeisestelle␣vor␣dem␣
          Augartenpalais;␣Pr\"asident Seitz besichtigt mit dem Leiter der
          Aktion, Capitain Torrey und dem Generalkommissar Dr. Geist die
          Ausspeisestelle.</onb:Beschreibung>
8     <onb:DargestelltePerson>Seitz, Karl</onb:DargestelltePerson>
9     <onb:Datensatznummer>1086319</onb:Datensatznummer>
10    <onb:DatierungVon>1919-01-01</onb:DatierungVon>
11    <onb:DatierungBis>1919-01-01</onb:DatierungBis>
12    <onb:Institution>\"Osterreichisches␣Institut␣f\"ur Zeitgeschichte (
          OEGZ)</onb:Institution>
13    <onb:Inventarnummer>S 646/48</onb:Inventarnummer>
14    <onb:Klassifikation>Arbeit und Soziales</onb:Klassifikation>
15    <onb:Medientyp>Fotografie</onb:Medientyp>
16    <onb:Rechteinhaber>BAA/OEGZ</onb:Rechteinhaber>
17    <onb:SchlagwortFrei>Erste Republik (terreich)</onb:SchlagwortFrei>
18    <onb:SchlagwortFrei>Kinder</onb:SchlagwortFrei>
19    <onb:SchlagwortFrei>Kinderhilfswerk</onb:SchlagwortFrei>
20    <onb:SchlagwortFrei>USA</onb:SchlagwortFrei>
21    <onb:SchlagwortFrei>Soziales</onb:SchlagwortFrei>
22    <onb:SchlagwortFrei>Armut</onb:SchlagwortFrei>
23    <onb:SchlagwortFrei>Er\"offnung</onb:SchlagwortFrei>
24 ␣␣␣␣<onb:Technik>FOTOGRAFIE</onb:Technik>
25 ␣␣␣␣<onb:Titel>Jugendhilfswerk␣der␣Amerikaner␣f ner␣Kinder</onb:Titel>
26 ␣␣␣␣<rdf:type␣rdf:resource="http://www.bricks.org/base_ontology#DLObject
       "␣/>
27 ␣␣</onb:Bild>
28 </rdf:RDF>
```

Listing 5.2: The Classification Element

```
1 <onb:Klassifikation>Arbeit und Soziales</onb:Klassifikation>
```

The prototype itself is an *Eclipse [13] Rich Client Platform (RCP)* application written in *Java [35]*. The RCP has several advantages such as the easy use of plug-ins and the use of the SWT-library which is a fast graphic library. It also offers the ability to deploy native GUI applications to a variety of desktop operating systems, such as Windows, Linux and Mac OSX and an integrated update mechanism for deploying desktop applications from a central server. Unlike other Java programs, a RCP is nearly as fast as a C++ application what very increases the "feel" when interacting with the program. The tool uses some plug-ins such as *GATE [16]* or *WEKA [29]*, to handle the information retrieval and machine learning tasks. GATE in addition uses other plug-ins like Jena or Lucene which will be discussed shortly.

A big part of the prototype which is named "rdfEnricher", deals with the processing and visualizing of records, like the ability to load the previously built thesaurus or the processing of the metadata records. The thesaurus is defined in *SKOS* and can be represented by the prototype as a tree, which makes it easy for users to get an overview of the available concepts. As seen in section 2.9.2, a SKOS-record defines its concepts with the tag "skos:concept" and structures them by using "skos:broader" or "skos:narrower". The used tree structure perfectly fits this hierarchical structure. Metadata records on the other hand are listed by their ids. The content of each id can be viewed in a table by clicking on the id. The table is alphabetically sorted and displays every element and its according data. If the record contains equal elements but with different data the table will separate the data by a comma. Figure 5.1 shows the main window of the "rdfEnricher".

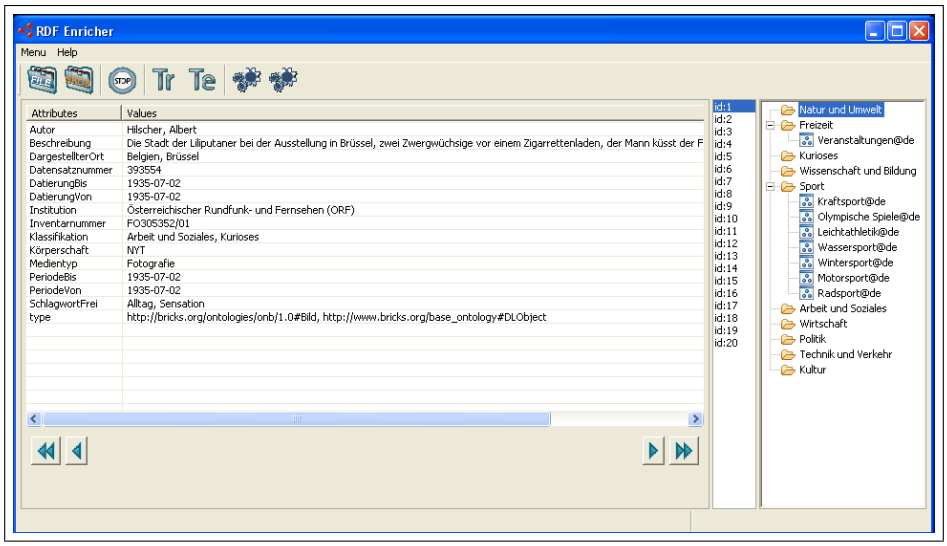

Figure 5.1: The main window of the "rdfEnricher" showing the thesaurus on the right side, whereas the remaining two windows represent the metadata record

As both the thesaurus and the metadata records are defined in RDF, the prototype needed a plug-in to

distinguish between elements, attributes and its content. Therefore the *JENA [23]-Framework* has proven helpful. Thus it is possible to process each element individually what is needed when it comes to the task where the records should be indexed. To load metadata records into the graphical user interface (GUI), the user has to click at the “File”-icon on the tab-Bar or choose it from the menu. After clicking, a dialog appears where the user has to pick the directory which contains the records. The records will then be parsed and displayed. The thesaurus can be loaded by clicking at the “SKOS”-icon or choosing the task from the menu. In the following dialog, the user has to choose the RDF-file which represents the thesaurus.

5.1 Classification

After loading thesaurus and metadata records, the user can either define or change a stop-list, or start to classify the metadata records in order to build the training set. The stop-list can be opened by pressing on the stop-icon in the tab-bar or via the menu. A stop-list contains words which will not be considered when building the $tf - idf$-matrix. Normally, stop-words or functional-words are those words which do not carry meaning and are therefore irrelevant for the matrix. But it can also contain character sequences like numbers and punctuation. Depending on the the language of the metadata records, stop-words vary from language to language. Figure 5.2 shows the opened stop-list and some german words as the metadata records are in german. If the stop-list contains all unnecessary terms the user can proceed with the creation of the $tf - idf$ by building a training set.

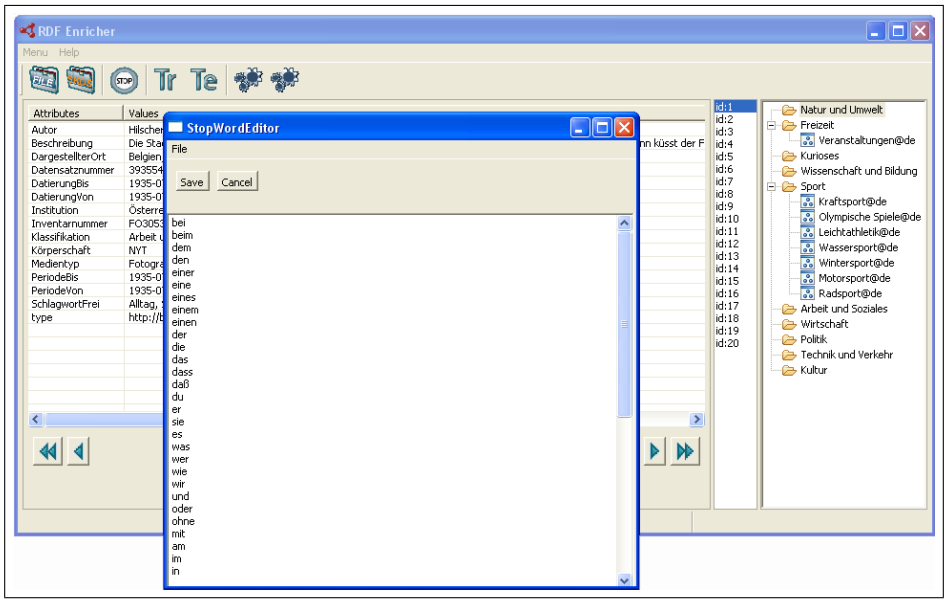

Figure 5.2: The definition of the stop-list

The loaded metadata records have to be related to its concepts represented by the thesaurus. To relate a metadata record to a concept, a user has to mark a specific record-id and drag&drop it to the right node of the thesaurus. Which class fits best depends on the content of the record which is also displayed when clicking on it. After the user has dropped the record to a concept the content of the record will show an additional element marked by the name “CLASS” as figure 5.3 shows.

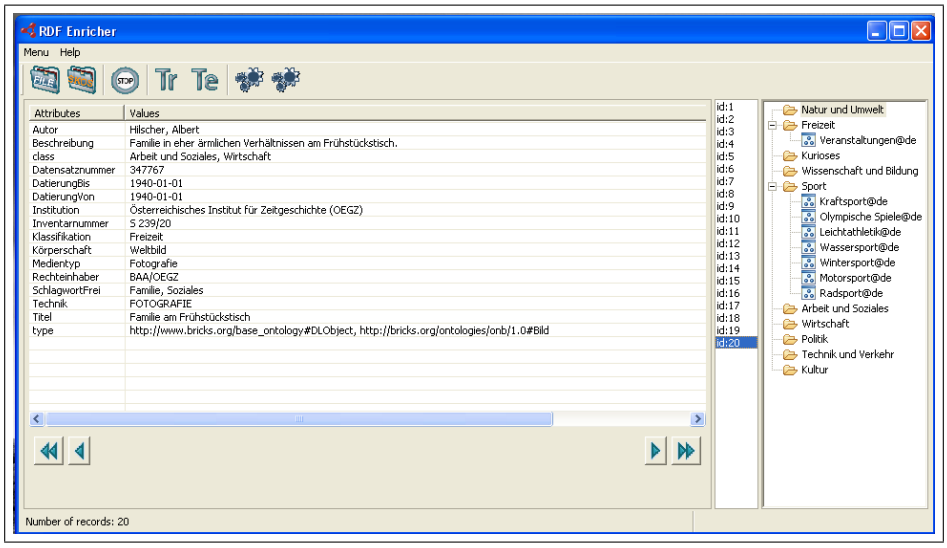

Figure 5.3: The additional “CLASS”-element after dragging&dropping a record to its class

After all records have been classified, the user can choose to build the training set. Therefore he has to click either at the button in the tab-bar or via the menu. The following dialog is crucial for the creation of the $tf - idf$-matrix because here the user can choose which elements should be taken for building the matrix. In the specific case of the “rdfEnricher”, many elements are worthless when using automatic classification. The element “date” for example can not be considered for classification because one can not decide to which class a record belongs based on its date. In the worst case, such an element can lead to wrong classification results. Also the type or the author of a photograph is not important for classification. Significant elements could be the descriptive key words, the title or the description element. All elements which should be taken into consideration can be checked in the dialog as figure 5.4 shows. After clicking the OK-button, the prototype builds an $tf - idf$-matrix. Therefore it uses the german-analyzer of the lucene-framework which recognizes the tokens (=terms), separates the words from the stop-list, stems them and calculates the $tf - idf$ value for each term. To get good $tf - idf$-values, it is also important that every class is represented by an equal amount of documents. After the creation of the tf-idf matrix for the training set the user can build the test set to measure the classifiers performance.

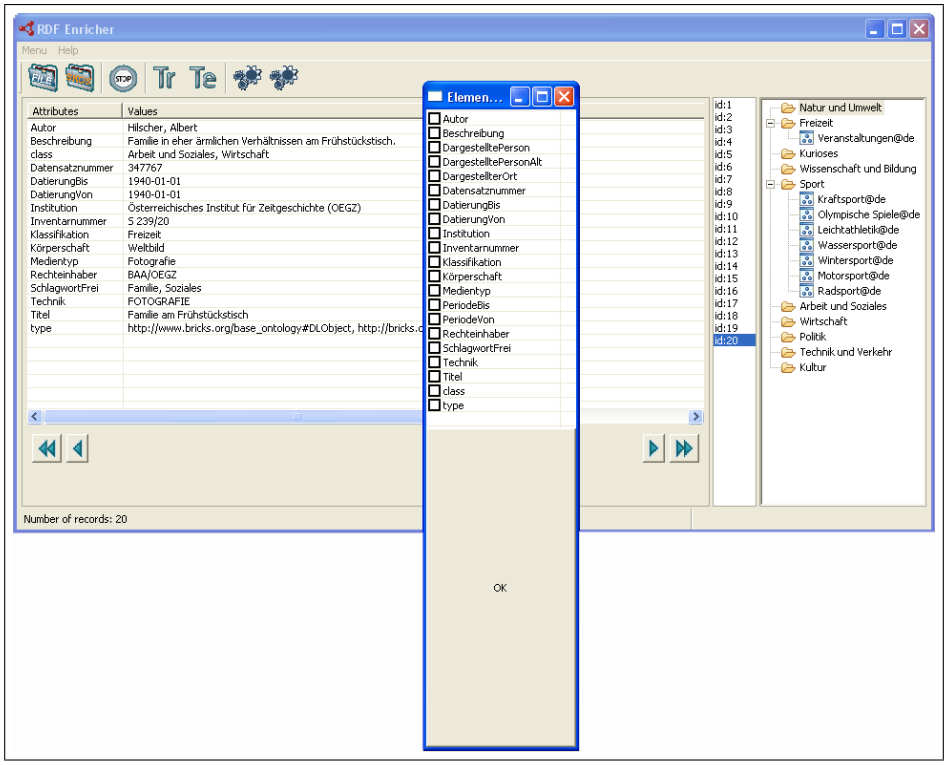

Figure 5.4: The elements-selection dialog when building a training set

The creation of test set is very similar to the one of the training set but easier. The user only has to load the new records into the GUI and classifying them by drag&drop. The last step of this process is to click at the icon with “Te” on it. The used elements are the same as defined at the creation of the training set and must not be defined again. The resulting two matrices can now be used to build the classifier. As written in chapter 4, the matrices have to be adjusted to be comparable. So when clicking at the “build classifier”-button the “rdfEnricher” compares the attributes of both matrices and finds those which occur in the training and test set. The result are two new matrices where each matrix has the same amount of attributes, so that they can be used by the classifier.

Having the new $tf - idf$-matrices, the “rdfEnricher” will build two ARFF-files out of it. ARFF-files are used by the *WEKA [29]* framework which is an API written in Java that provides several classes for the task of machine learning classification. An *Attribute-Relation File Format (ARFF)* file is an ASCII text file, that describes a list of instances sharing a set of attributes. ARFF-files consists of two different sections. Namely the header- and the data-section. The header gives information about the relation and the possible attributes. An attribute describes the value in the data section and can be numeric, nominal, a string or a date.

Lines that begin with a % are comments. The @RELATION, @ATTRIBUTE and @DATA declarations are case insensitive. Attributes must not have blanks or apostrophes in it, otherwise the framework returns a failure. But that is nothing that the user concerns because the "rdfEnricher" takes care of the creation of the ARFF-file. The last attribute in the header section is considered as the class-attribute, regardlass if it is defined as such attribute or not. Listing 5.3 shows an ARFF-record which was used to describe the "Iris Plants Database". WEKA also offers a GUI which can be used to test several classification or clustering algorithms in a convenient way. There is also a large amount of downloadable datasets which can be fount under http://www.cs.waikato.ac.nz/ml/weka/.

Listing 5.3: Example of an ARFF-record

```
1
2   \% 1. Title: Iris Plants Database
3   \%
4   \% 2. Sources:
5   \%      (a) Creator: R.A. Fisher
6   \%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
7   \%      (c) Date: July, 1988
8   \%
9   @RELATION iris
10
11  @ATTRIBUTE sepallength  NUMERIC
12  @ATTRIBUTE sepalwidth   NUMERIC
13  @ATTRIBUTE petallength  NUMERIC
14  @ATTRIBUTE petalwidth   NUMERIC
15  @ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
16
17
18  @DATA
19  5.1,3.5,1.4,0.2,Iris-setosa
20  4.9,3.0,1.4,0.2,Iris-setosa
21  4.7,3.2,1.3,0.2,Iris-setosa
22  4.6,3.1,1.5,0.2,Iris-setosa
23  5.0,3.6,1.4,0.2,Iris-setosa
24  5.4,3.9,1.7,0.4,Iris-setosa
25  4.6,3.4,1.4,0.3,Iris-setosa
26  5.0,3.4,1.5,0.2,Iris-setosa
```

```
27  4.4,2.9,1.4,0.2,Iris-setosa
28  4.9,3.1,1.5,0.1,Iris-setosa
```

Each record is represented on a single line. Attributes are delimited by commas and must appear in the order they were declared in the header section. Missing values are represented by a single question mark, as in: “5.1,3.5,1.4,0.2,?”. A record used by the “rdfEnricher” will probably have hundreds of attributes where the last stands for the related class. With automatic classification those classes will only occur in the training- and test-set while others will have a ’?’ instead of the class.

Once the ARFF-files are written, the process continuous with the training of the classifier. Therefore the SVM is used as it has profen to return the best results. In the next step the trained classifier automatically relates the data of the test set to the most likely classes and compares it results with the results of the pre-defined test set. A pop-up-window will inform the user of its performance. Expecting that the results are satisfying the step of building the classifier can be considered as ready.

Before loading the remaining data records into the GUI and classify them, the user has the possibility of a further specification by using logical rules. To define such a rule, he can click with the right mouse button on a concept and type in those words which always indicate a relation to the concept. An example of such a word could be “football” for the concept sport. Figure 5.5 shows the rules for the concept “Kultur” (=culture). Logical rules will trigger regardless of the result of the classifier what can very much increase the classification result. Although classification rules can increase the results, the prototypical implementation can also be used without defining any of them.

Having now proceeded all steps to achieve good classification results, it is time to classify a larger amount of metadata records, where automatic classification will save a lot of time and therefore money. With the “rdfEnricher” this step is pretty much the same as it was with the test set. But instead of classifying the data via drag&drop, this time the classes will be defined automatically. The created ARFF-file will therefore have question marks instead of the class at its last attribute. But instead of a pop-up-window containing some statistics the user only get a short information of how many records have been classified. The result can partially be examined by browsing through the records, where each record now has the additional “CLASS”-element. If the expert user thinks that the classifier did a good job and that most of the records are correctly related to its classes he can finally write the new xml-records to disc, to a database or whatever. In contrast to the old metadata records, this one includes an element that, similar to the identifier, points to the class of the record as figure 5.6 demonstrates. In the case of bad classification results, the user can go back and built a new classifier or boost the old one by adding additional records.

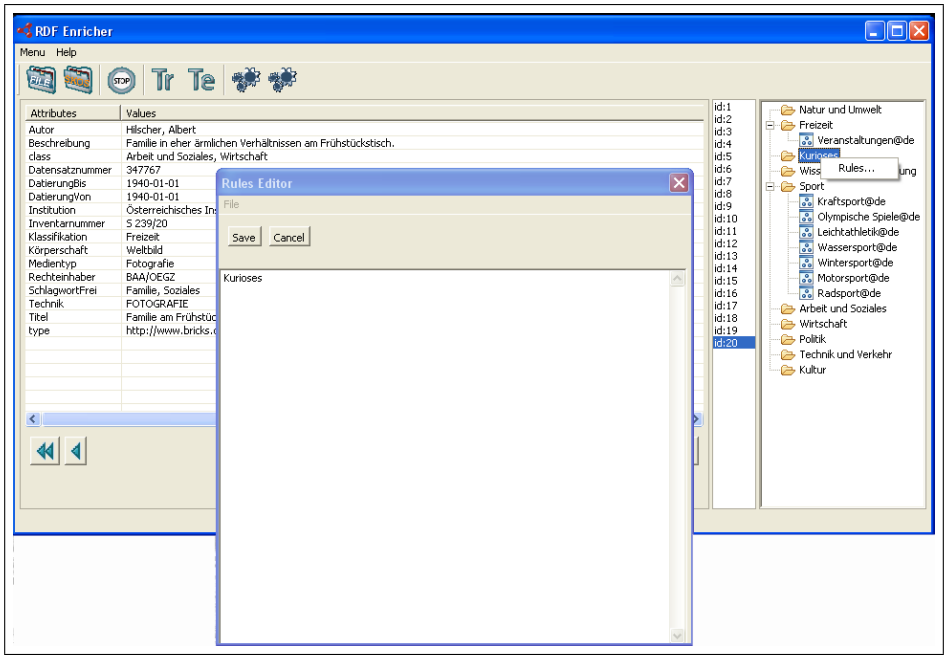

Figure 5.5: The definition of the logical rules dialog

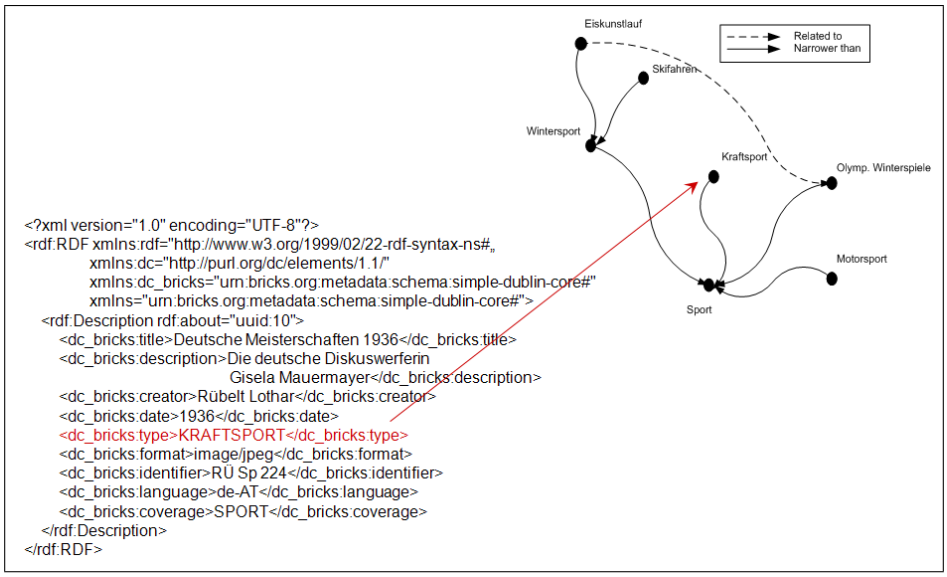

Figure 5.6: An element expressing the xml-concept-relation

Chapter 6

Case Study

This chapter will give some statistic measures of the results of the classification process used in the demonstrator. The data used for classification, was taken from the "ÖNBs", the Austrian National Libraries image archive. Each record describes an underlying image by the use of specific elements. Unlike other metadata records, the ÖNB-records already include a classification element, that describes the record's relation. Another unusual way of classification is, that a record can be related to more than one concept.

6.1 The Data Set

To train the classifier, but also when relating records manually to concepts, only a few of the record's elements are relevant for consideration. A relevant element is an element which describes the records content in some way. In contrast to those elements, that give information about the surrounding facts, like the used camera type or the date of creation. Table 6.1 gives an overview of all elements and a short description. The elements which will be useful for classification are "Beschreibung", "SchlagwortFrei" and "Titel". All other elements can be disregarded.

Table 6.2 depicts the possible concepts, which were, in the case of the "ÖNB"-records, pre-defined. Typically a thesaurus will be newly created by an end-user.

6.2 Scenario

The case study will test the classifier results on the basis of three training sets, where each set will vary in the amount of the concepts. Each concept will be represented by *ten* records which should give sufficient

element	*description*	*relevant*
“Autor”	the name of the author	no
“Beschreibung”	a short description of the content	yes
“DargestelltePerson”	the name of the displayed person	no
“Datensatznummer”	number of the record	no
“DatierungVon”	creation date	no
“Institution”	the name of the institution	no
“Inventarnummer”	inventory number	no
“Klassifikation”	classification	no
“Körperschaft”	statutory corporation	no
“Medientyp”	type of media	no
“Rechteinhaber”	rights owner	no
“SchlagwortFrei”	keyword, which shortly describes the content	yes
“Technik”	technique	no
“Titel”	title	yes

Table 6.1: Description of the elements of a legacy metadata record

concept name	*translation*
“Sport”	Sport
“Freizeit”	Leisure
“Arbeit und Soziales”	Employment and Social Concerns
“Wirtschaft”	Economy
“Technik und Verkehr”	Engineering and Traffic
“Politik”	Politics
“Kurioses”	Curiosity
“Wissenschaft und Bildung”	Science and Education
“Kultur”	Culture
“Natur und Umwelt”	Nature and Environment

Table 6.2: Description of the Thesaurus’ Concepts

information for a specific concept. From now on, the sets will be identified by small, medium and large set. Where the small set consists of *two*, the medium set of *five* and the large set of *ten* different concepts. At the beginning only the classifier, not the logical rules, will be applied on each set. To test the performance of the classifier, there are also three test sets which are equivalent to the training set, except they have different records.

Further, the results will always give two charts of the classification, as well as the recall and precision measurements for every concept. The two charts are different in the elements which were taken into account to train the classifier. As it will be proven later, in the specific case of the data, it is very important which elements to choose. One of the three elements, needs to be taken under a closer look. The "SchlagwortFrei" element gives the impression to be very significant at the first glance. As terms that reappear in documents, will become more meaningful (=high weight), a classifier's results will strongly depend on those terms. The problem here is, in combination with the possibility of multiple classification, that those terms can falsify the results of the classifier. That means, that in the case of having records with non-overlapping concepts, the terms from "SchlagwortFrei" will be mutually exclusive and as a result will be a good indicator for the classification algorithm. With increasing concepts, the records will have "SchlagwortFrei"-elements, that will contain overlapping terms, what will lead to a fuzzy decision boundary between concepts. In the case of the "ÖNB" this problem was solved, by relating the record to all concepts which came into question. Unfortunately, an automatic classifier is not good at handling terms, that would fit to several concepts. Once the classifier is trained on such records, the whole classification process will perform badly. The results therefore contain a classification including the "SchlagwortFrei" element and one where the element was not taken into account.

A similar problem occurs with concepts that are too abstract to be interpreted by a machine. This is the case with the concept "Kurioses" which contains records that illustrate an oddity in some way. Due to the fact, that terms which normally are relevant for other concepts, are now significant to a different concept, again will influence the performance of the classifier. For example the two sentences: "a man on a bike" and "a parrot on a bike", will have high weights on the term "bike". When training the classifier, it will be likely that most of the records, containing "bike" will be classified under "Traffic", whereas at least one will be classified under "Curiosity", what will have a global impact on the classifier's performance.

To be able to test several classification algorithms, the prototypical implementation was designed flexible enough to easily switch between them. After a few tests with different machine-learning classifiers, like the

Naive Bayes or the *Support Vector Machine (SVM)*, it pointed out, that the SVM achieved the best results.

6.3 Results

6.3.1 Small set with two classes

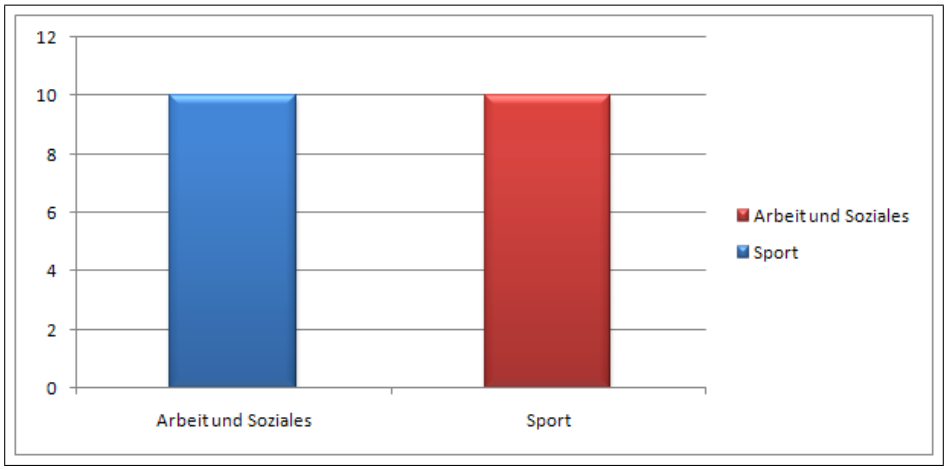

Figure 6.1: The small test set including all elements

The chart shows the small dataset with the two classes “Arbeit und Soziales” and “Sport”. Classification with two concepts performs very well. The percantage of correctly classified instances was *100 percent*. This means that precision and recall are both *1.0*.

When not including the “SchlagwortFrei”-element the number of correctly classified instances decreases to *85 percent* of all records. The statistic accuracy by concept is given by a precision of *0.769* for “Arbeit und Soziales” and *1.0* for “Sport”, whereas the recall of the concept “Arbeit und Soziales” is *1.0* and the one of “Sport” is *0.7*.

6.3.2 Medium set with five classes

Again there are two charts representing the results of the classifier. Here the classifier was trained on five classes, which all are relatively distinctive from each other. As the comparison of figure 6.3 and 6.4 will show, not including the “SchlagwortFrei” element would lead - in this case - to better classification results. Again, this is because sometimes the element includes different (not mutually exclusive) class names that irritate the classifier. The classifier trained without the category “SchlagwortFrei” here performs better.

Table 6.3.2 and table 6.3.2 will give detailed information of the accuracy by class.

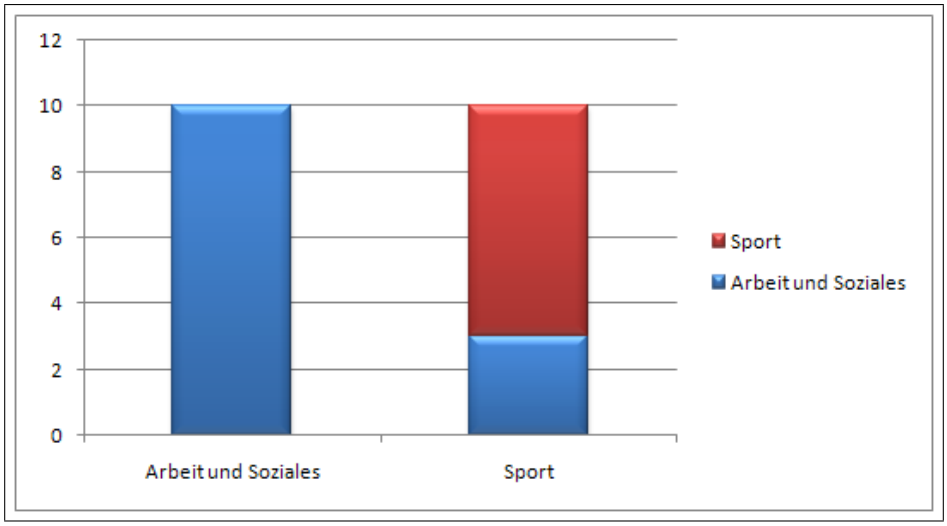

Figure 6.2: The small test set not including the “SchlagwortFrei” element

precision	*recall*	*class*
1.0	0.8	Wirtschaft
0.889	0.8	Wissenschaft und Bildung
1.0	1.0	Sport
0.714	0.5	Technik und Verkehr
0.563	0.9	Natur und Umwelt

Table 6.3: Recall and precision on medium test set when using “Title”, “Beschreibung” and “SchlagwortFrei”

6.3.3 Large set with ten classes

Figure 6.5 and figure 6.6 show, that with increasing classes, the performance of the classifier decreases. Especially when it is the case of not considering the “SchlagwortFrei” element. A closer look reveals that this so, because the data is not clearly separable. Also an expert user would have problems to uniquely relate a record to a specific concept. The original contained multiple classifications and keywords for several concepts to which the records fit. In the case of automatic classifying this leads to bad results because of the fuzzyness of the decision boundaries. Even a few records with unclear margins, can lead to bad classification results.

Nevertheless, the classifier correctly classified *73 instances* out of 100 with including the “SchlagwortFrei” element. But, as stated earlier, with increasing concepts, some records will contain terms, which are significant for different concepts. This will confuse the classifier and interfere the results. With ten concepts and not including the crucial “SchlagwortFrei”-element, the classifier’s performance gets useless. This is also because of some originally weak defined concepts, like “Kurioses” or “Freizeit”. Results for the automatic

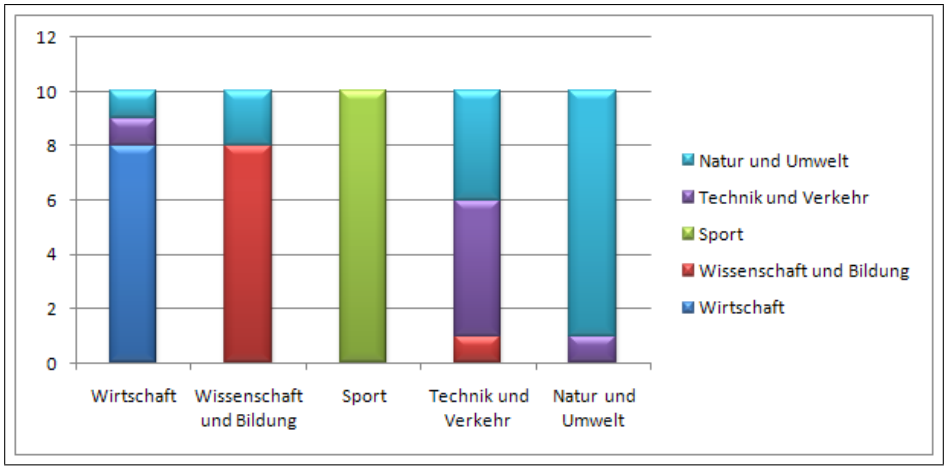

Figure 6.3: The medium test set including all elements

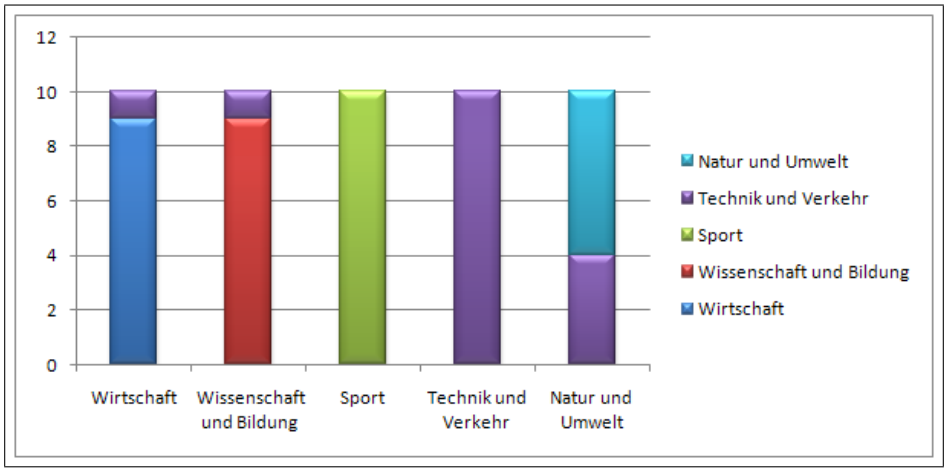

Figure 6.4: The medium test set not including the “SchlagwortFrei” element

classification are given in table 6.3.3 and in table 6.3.3.

The classifier returns 61 incorrectly classified instances out of 100 when only using the elements “Titel” and “Beschreibung”.

6.4 Opportunities

As the results show, automatic classification using all concepts, is not very satisfying. One approach to improve the performance of the classifier could be the further training of the classifier with more records. Anyhow, this would not lead to meaningful performance increase. The main reason for such a result, is the quality of the data records and the fact, that the concepts already existed, what left little space for classifica-

precision	*recall*	*class*
1.0	0.9	Wirtschaft
1.0	0.9	Wissenschaft und Bildung
1.0	1.0	Sport
0.625	1.0	Technik und Verkehr
1.0	0.6	Natur und Umwelt

Table 6.4: Recall and precision on medium test set when using “Title” and “Beschreibung”

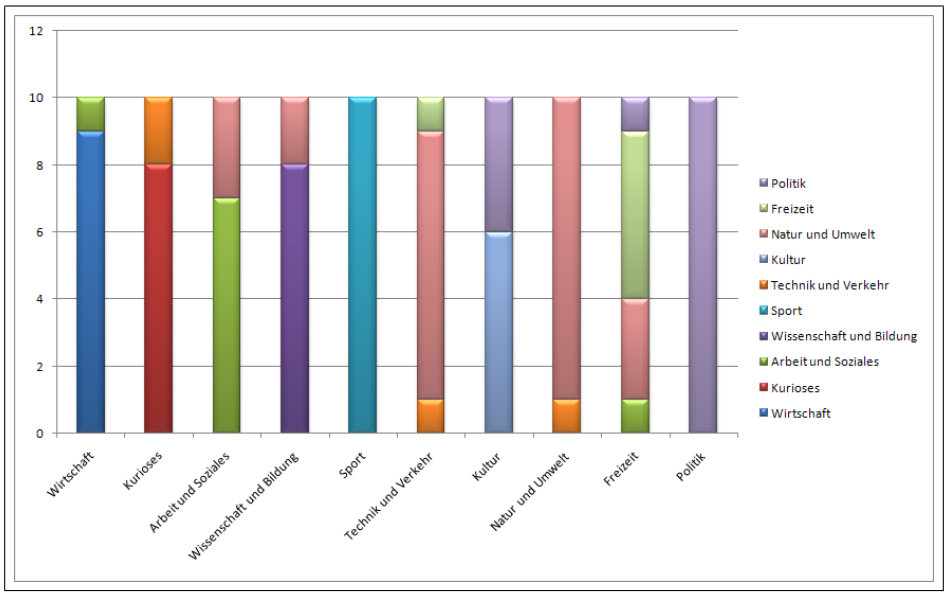

Figure 6.5: The large test set including all elements

tion. Newly created concepts would be defined more disjunctive, so that a classifier could better define its decision boundaries.

Due to the possibility of defining logical rules, the classification result can also be enhanced. In the case of the underlying data, some clearly defined rules may treat about 70 percent of all records. This is because the title or the short description often include one term that is significant for the relation-process. For example the terms “train” or “car” are markers for a relation to the concept “Engineering and Traffic”. Or “holiday” would be a strong sign for a relation to the class “Leisure”.

Another option is to change the classifier in a way, that it is able to relate records to more than one class on the basis of probabilities or distance measurements. This means, that if a classifier finds a record, that has a relation-probability of 50 percent to the concept “Leisure” and of 40 percent to “Curiosity” the classifier

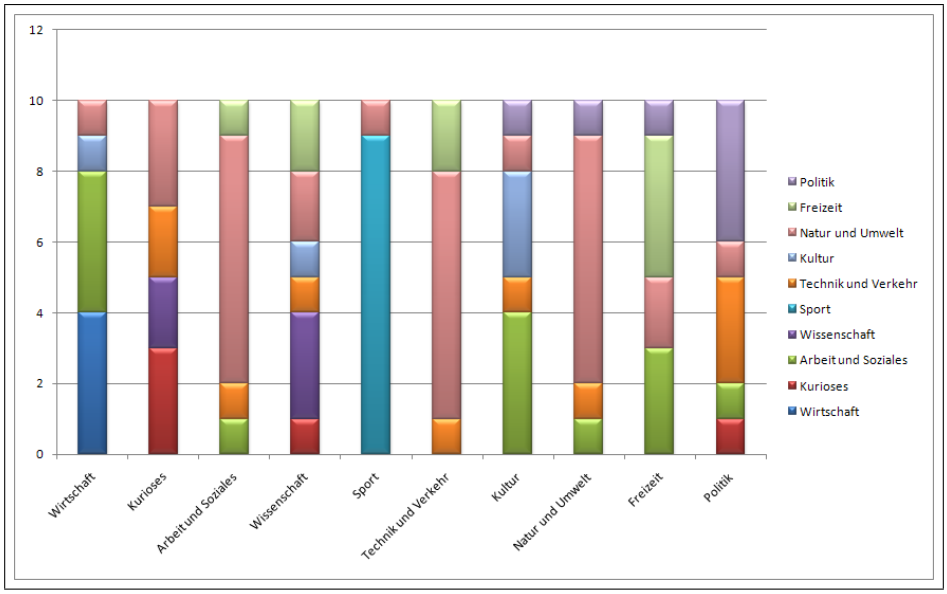

Figure 6.6: The large test set not including the “SchlagwortFrei” element

would relate the record to both classes.

precision	*recall*	*class*
1.0	0.9	Wirtschaft
1.0	0.8	Kurioses
0.778	0.7	Arbeit und Soziales
1.0	0.8	Wissenschaft und Bildung
1.0	1.0	Sport
0.25	0.1	Technik und Verkehr
1.0	0.6	Kultur
0.36	0.9	Natur und Umwelt
0.833	0.5	Freizeit
0.667	1.0	Politik

Table 6.5: Recall and precision on large test set when using “Title”, “Beschreibung” and “SchlagwortFrei”

precision	*recall*	*class*
1.0	0.4	Wirtschaft
0.6	0.3	Kurioses
0.71	0.1	Arbeit und Soziales
0.6	0.3	Wissenschaft und Bildung
1.0	0.9	Sport
0.1	0.1	Technik und Verkehr
0.6	0.3	Kultur
0.219	0.7	Natur und Umwelt
0.444	0.4	Freizeit
0.571	0.4	Politik

Table 6.6: Recall and precision on large test set when using “Title” and “Beschreibung”

Chapter 7

Conclusion

The thesis shows a possibility which can increases the capabilities of search and discovery mechanisms. In contrast to existing approaches, which try to obtain semantic rich metadata descriptions based on a resource's semantic or structural features, this one had its focus on enriching existing legacy metadata records. Enriching is the process of making a metadata record ontology-aware. Furthermore the work shows, that the enrichment process can be done in a semi-automatic way.

The first part of the thesis explained ways to organize knowledge, which is a subject of matter since 2000 years and originally came from librarians and philosophers. The only difference now is, that frameworks handle the stored knowledge and not the human expert user. An important instrument there are ontologies, which define ways to reflect the semantics of the underlying data. Machine-processable data in combination with ontologies, allow a reasoner to exploit this semantics. Search engines would be able to sense the meaning of questions which in turn increase search possibilities. To create a relation between a data record and an ontology, the data record has to be enriched. As such a process would be too tiresome for an end-user, the work offered a semi-automatic way to do the enrichment process.

The second part of the thesis showed actual machine learning techniques, which were later used to make metadata records ontology-aware. Supervised machine learning classifiers separate data on the basis of a trainings phase. The separated data records are grouped together due to its features. Each group is equivalent to an ontologies' concept. To find meaningful features and to guarantee that the classifier functions properly, the field of text retrieval was needed. With text retrieval it is possible to transform data records into a *Vector Space Model (VSM)* which is needed by the classifier. It has been shown, that a *term frequency - inverted document frequency* ($tf - idf$) *matrix*, is a useful transformation for the further use with a classifier.

To prove the works practicability a prototypical implementation was used to enrich legacy metadata records describing images. The data and a pre-defined ontology was from the image archive of the "ÖNB", the Austrian National Library. The implementation allows an end-user to load any ontology and metadata records. After the loading process the user could relate records to concepts of the ontology by dragging a record to the fitting concept. After a few "manual" relations the prototype allowed the definition of an automatic classifier which could then in turn could be applied to the remaining metadata records. The prototype showed that the tiresome and time consuming process of enriching records by an end-user, could be done in much less time by using techniques of machine-learning.

To verify the results, the classifier was tested on different datasets, that varied in its size and the number of concepts. Through the special case of the existing data records, the result was not fully convincing. This was because of several problems of the data's and also the ontologies' nature. It turned out, that some terms are meaningful for different concepts, what as in consequence had influence on the classifiers performance. Another problem was, that some records had multiple relations to concepts, because the concepts of the ontology were not mutually exclusive. Spare descriptions were also a problem. But regardless of this, the classifier worked well on up to five concepts. A test on ten concepts showed that the classifier could not be applied here.

To be able to use such an implementation in practice, the most important thing is to have an ontology with mutually exclusive concepts or concepts that are easily distinguishable. Another performance increase could be reached with the use of logical rules. If it is needed to have multiple relations, the classifier could use probabilities to solve this problem. That means if two concepts have high probabilities, both will be used for enriching.

As a final remark it can be said, that this thesis shows a deployable way on how to increase the use of existing data records by making them ontology-aware.

Bibliography

[1] Thomas Aquinas. *De Ente Et Essentia*. Internet Medieval Source Book, 1255.

[2] Aristoteles. *Organon Band 2. Kategorien. Hermeneutik oder vom sprachlichen Ausdruck.* Meiner, 2001.

[3] Thomas Bayes. An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, a. m. f. r. s. http://www.stat.ucla.edu/history/essay.pdf, 1763.

[4] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. http://research.microsoft.com/ cburges/papers/SVMTutorial.pdf, 1998.

[5] Margarete Burkart. *Grundlagen der praktischen Information und Dokumentation*, volume 4, chapter Thesaurus, pages 160–179. K. G. Saur Verlag, 1997.

[6] Vannevar Bush. As we may think. *The Atlantic Monthly*, 176:101–108, July 1945.

[7] E. F. Codd. A relational model of data for large shared data banks. *ACM*, 13:377–387, 1970.

[8] H. Cunningham. Information Extraction, Automatic. *Encyclopedia of Language and Linguistics, 2nd Edition*, 2005.

[9] Michael C. Daconta, Leo J. Obrst, and Kevin T. Smith. *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. Wilkert, Joe, 2003.

[10] Politecnico di Milano. A tutorial on clustering algorithms. http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial/kmeans.html.

[11] E. Dr. Garcia. Cosine similarity and term weight tutorial. http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html, 2006.

[12] DublinCore. Dublincore homepage. http://www.dublincore.org.

[13] Eclipse. The eclipse homepage. http://www.eclipse.org.

[14] Charles Elkan. Naive bayesian learning. Technical report, University of California, San Diego, September 1997.

[15] Michael R. Genesereth and Nils J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., 1987.

[16] Sheffield NLP Group. Gate. http://www.gate.ac.uk.

[17] Tom Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5:199–220, 1993.

[18] Marti A. Hears and Jan O. Pedersen. Proceedings of the nineteenth annual international acm sigir conference. In *Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results*, June 1996.

[19] Peter Ingwersen. *Information Retrieval Interaction*. Taylor Graham Publishing, 1992.

[20] ISO. Iso 2788:1986, documentation – guidelines for the establishment and development of monolingual thesauri, 1986.

[21] Immanuel Kant. *Logic*. Courier Dover Publications, 1988.

[22] Univ.Prof. Dr. Wolfgang Klas. Multimedia-content management (skriptum vorlesung im ws 05/06), 2001-2005.

[23] HP Labs. http://jena.sourceforge.net/.

[24] Peter Lyman and Hal R. Varian. How much information? http://www.sims.berkeley.edu/how-much-info-2003, 2003.

[25] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *An Introduction to Information Retrieval*. Cambridge University Press, 2007.

[26] William Mitchell, Charles Ford, Don Kraft, and John Talburt. What is the difference between information science and computer science?, 2003.

[27] NISO. *ANSI/NISO Z39.85-2001, The Dublin Core Metadata Element Set*. NISO Press, 2001.

[28] NISO. Ansi/niso z39.19 -2005 guidelines for the construction, format, and management of monolingual controlled vocabularies. http://www.niso.org/standards/standarddetail.cfm?stdid=814, 2005.

[29] University of Waikato. Weka. http://www.cs.waikato.ac.nz/ ml/.

[30] Eleanor Rosch. Principles of categorization. *Cognition and categorization*, 1978.

[31] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of ACM*, 18(11):613–620, November 1975.

[32] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.

[33] Stanford. Wine ontology. http://www.daml.org/ontologies/76.

[34] Alexander Strehl. External (model-free, semi-supervised) quality, 2002.

[35] Sun. The java homepage. http://java.sun.com.

[36] Michael Uschold and Jernst. What are the differences between a vocabulary, a taxonomy, an ontology and a meta-model? http://www.metamodel.com/article.php?story=20030115211223271, January 2003.

[37] Vladimier Vapnik. *Estimation of Dependences Based on Empirical Dat*. Springer Verlag, 1982.

[38] W3C. Isaviz: A visual authoring tool for rdf. http://www.w3.org/2001/11/IsaViz/.

[39] W3C. Rdf vocabulary description language 1.0: Rdf schema. http://www.w3.org/TR/rdf-schema/.

[40] W3C. The semantic web activity. http://www.w3.org/2001/sw/, 2001.

[41] W3C. Skos specification development. http://www.w3.org/2004/02/skos/.

[42] W3C. Owl web ontology language overview. http://www.w3.org/TR/owl-features/, 2004.

[43] Amy J. Warner. A taxonomy primer. http://www.lexonomy.com/publications/aTaxonomyPrimer.html, 2002.

[44] Wikipedia. Inverted index. http://en.wikipedia.org/wiki/InvertedIndex.

[45] WORDNET. Wordnet - a lexical database for the english language. http://wordnet.princeton.edu/.

[46] Zeng and Salaba. Types of knowledge organisation system (kos). FRBR Workshop, OCLC 2005, 2005.

[47] Marcia Lei Zeng. Construction of controlled vocabularies a primer. Internet, 2005.