Information extraction from the web using a search engine

Information Extraction from the Web using a Search Engine

ISBN: 978-90-74445-85-6

Cover design by Paul Verspaget
Photo by Marianne Achterbergh

The work described in this thesis has been carried out at the Philips Research Laboratories in Eindhoven, the Netherlands, as part of the Philips Research programme.

Information Extraction from the Web using a Search Engine

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op maandag 8 december 2008 om 16.00 uur

door

Gijs Geleijnse

geboren te Breda

Dit proefschrift is goedgekeurd door de promotor:

prof.dr. E.H.L. Aarts

Copromotor:
dr.ir. J.H.M. Korst

Contents

1

Introduction

"All science is either physics or stamp collecting."
— Ernest Rutherford.

Whether we want to know the name of Canada's capital, or gather opinions on Philip Roth's new novel, the World Wide Web currently is the de-facto source to find an arbitrary piece of information. In an era where a community-based source as Wikipedia is found to be as accurate as the Encyclopaedia Britannica [Giles, 2005], the collective knowledge of the internet contributors is an unsurpassed collection of facts, analyses and opinions. This knowledge simplifies the process for people to gather knowledge, form an opinion or buy a cheap and reliable product.

With its rise in the late nineties, the web was intended as a medium to distribute content among an audience. Alike newspapers and magazines, the communication was merely one way. The content published on the web was presented in an often attractive format and lay-out, using a natural language (e.g. Dutch) we are most acquainted with.

Nowadays, only a few years later, the web is a place where people can easily contribute, share and reuse thoughts, stories or other expressions of creativity. The popularity of social web sites enriches the information available on the web. This mechanism turned the web into a place where people can form nuanced opinions about virtually any imaginable subject.

To enable people to share and reuse content, such as the location of that great Vietnamese restaurant in Avignon on *Google Maps*, the information on the web is currently not only presented in a human-friendly fashion, but also in formats that allow interpretation of information by machines. The so-called Social Web, or Web2.0, enables people to easily create and publish content. Moreover, content can be easily reused and combined.

A movement next to the social web is the *semantic web*. The semantic web community has created a dedicated formal language to express concepts, predicates and relations between concepts. Using this *mathematical language for general information*, knowledge can be expressed on every imaginable topic. The semantic web can be seen as a distributed knowledge base. Instead of browsing through web pages, the semantic web enables direct access to *information*.

The more information is already expressed in the semantic web languages, the easier it becomes to represent new information. For example, to model the concept of *First Lady of the United States*, it may be needed to first model the concepts *country*, *United States*, *person*, *president*, *married*, *time*, *period* and so on. The use of earlier defined notions makes the content of the semantic web richer, as content created by various parties can be linked and combined.

In the late sixties in Eindhoven, N.G. De Bruijn and his group developed the *mathematical language for mathematics* and system Automath [De Bruijn, 1968; Nederpelt, Geuvers, & De Vrijer, 2004]. Automath is a dedicated formal language to express mathematics. The project can be seen as an attempt to formulate and propagate a universal language for mathematics, that is checked by a system. Such languages serve two goals. On the one hand, it is a means to ensure mathematical correctness. If a theorem is provided with a proof in the mathematical language, and the well-designed system accepts this proof, then the theorem can be considered to be true. On the other hand, the language provides a means of clear and unambiguous communication.

Białystok, Poland, the home town of the constructed language *Esperanto*, is the base of one of the most active projects on formal mathematical languages. The Mizar system builds on a set of axioms. A collection of mathematics is formalized (i.e. derived from the set of axioms) through out the years. Although the Mizar team have succeeded to completely formalize a whole handbook on continuous lattices (by 16 authors in 8 years time), the formalization of an elementary theory in another mathematical subject (i.e. group theory) proved to be too ambitious [Geleijnse, 2004].

In spite of the work done by the semantic web and formal mathematics researchers, both mathematicians and web publishers prefer natural language over dedicated artificial languages to express their thoughts and findings. In mathematics, dedicated researchers are formalizing (or translating) definitions, theorems

and their proofs into formal languages. The translation of mathematics into formal languages was the topic of my 2004 master's thesis. In this thesis, I will discuss approaches to catch information on the web into a dedicated formalism. Although both topics may be closer to stamp collecting than to physics, I do hope that you will enjoy this work.

1.1 Information on the Web

In this thesis, we focus on information that is represented in natural language texts on the web. We make use of the text itself rather than of the formatting. Hence, we extract information from *unstructured texts* rather than from formatted tables or XML. Although some web sites may be more authoritative than others, we do not distinct between sources as such.

Suppose we are interested in a specific piece of information, for example the capital of Australia. Nowadays, the web is an obvious source to learn this and many other facts. The process of retrieving such information generally starts with the use of a search engine, for example *Google* or perhaps the search engine in Wikipedia. As we are unaware of the name of Australia's capital, we query for terms that can be expected to co-occur with this specific piece of information. The term *Australia* is of course a good candidate, but the combination of the words *Australia* and *capital* will more probably lead to relevant pages.

The everyday internet user has learned to formulate effective search engine queries. However, the fact *'Canberra is the capital of Australia'* still has to be identified within the search results. The search engine returns documents that are likely to reveal this information, but we have to search the retrieved documents for the fact itself.

To understand a text, we have to be able to parse the sentences, know the precise semantics of the words, recognize co-references, read between the lines, resolve ambiguities etc. Hence, for machines this is not a trivial task.

The study of information extraction addresses a subproblem of document (or, text) understanding: the identification of *instances* of classes (e.g. names of persons, locations or organizations) and their relations in a text (e.g. the expressed relation between *Canberra* and *Australia*). In this thesis we study how information extraction can be applied on a specific text corpus: the web.

In this thesis, we focus on the following problem. We are given a domain of interest, expressed using classes and relations. The goal is to extract information from unstructured texts on the web. We first find relevant texts on the web using a search engine. Having retrieved a collection of relevant texts, we focus on two information extraction tasks. On the one hand we are interested to discover and extract instances from the given classes, while on the other hand we extract rela-

tions between such instances. The extracted information is stored in a structured, machine-interpretable format.

With structured information available, we can easily find the information we are interested in. The extracted information can be used in intelligent applications, e.g. in recommender systems to acquire additional meta data. This meta data can be used to make meaningful recommendations for music or TV programs. For example, suppose a user has expressed a preference for TV programs relating to France. The recommender system may be able to recognize regions as *Languedoc* and *Midi-Pyrénées* and cities as *Cahors* and *Perpignan* using the extracted information. Likewise, if the user has expressed a preference for French music the system will be able to recognize the names of artists like *Carla Bruni* and *Charles Aznavour*.

1.1.1 Structured Information on the Web

Of course, not all information on the web is unstructured. As alternative sources for information, we distinguish the following three structured representations of information on the web.

- The semantic web and other XML-based languages. Pages written in these language are dedicated subparts of the web for machine interpretable information. Information represented in these formats can fairly easily be extracted.
- Web sites with a uniform lay-out. Large web sites, that make use of a database, typically present their content in a uniform lay-out. For example, the lay-out of the *Amazon* page for a CD by *Jan Smit* has a similar lay-out as the page for *Spice* by *The Spice Girls*. Hence, given a page within *Amazon*, we can easily identify the title, price, reviews and other information based on the lay-out.
- Tables and other formatted elements inside web pages. In columns in a table, typically similar elements are stored. For example, if multiple terms from one column are known to be soccer players, all other terms in the column can be expected to be soccer players as well.

When we are interested in information that is available on the web, from a practical point of view, the use of unambiguous structured information is always preferred over the extraction of information from unstructured texts. However, as not all information is available in such a manner, web information extraction – from unstructured texts – is a relevant research topic.

1.1.2 The Social Web and its Potential

The web as we know it today enables us to get a nuanced view on products, events, people and so on. The internet community can easily create content in the form of weblogs, comments, reviews, movies, images and so on. All this information can be used to form an opinion or help in for example selecting the right mattress to buy or book to read. Although the content provided by amateurs may undermine the influence of journalists, critics and other professionals [Keen, 2007], we can learn from the *collective knowledge* of the web contributors.

Where the semantic web merely focusses on representing the facts of life, the social web touches on a more vague or abstract representation of knowledge: the 'wisdom of the crowds'. This collective knowledge can be seen as a sign of the times, or a general opinion about a subject.

1.2 Information Extraction and Web Information Extraction

Web Information Extraction (WIE) is the task to identify, structure and combine information from natural language texts on the web. Given a domain of interest, we want to create a knowledge base on this topic.

As information gathering from structured sources is in general easier and more reliable than the use of unstructured texts, web information extraction is particularly interesting for the following information demands.

- The information that cannot be extracted from structured or semi-structured sources, such as XML documents, single web sites or tables, but is spread across various web pages.
- The information that is expected to be present on the web. Obviously, we cannot extract information that is not present in the studied corpus. Hence, we can say in general that web information extraction is suited for all topics that people write about.

1.2.1 A Comparison between Web Information Extraction and *Traditional* Information Extraction

Information extraction (IE) is the task of identifying instances (*named entities* and other terms of interest) and relations between those instances in a collection of texts, called a text corpus. In this work, instances can be terms and other linguistic entities (e.g. *twentieth president*, *guitarist*, *sexy*) as well as given names (e.g. *The Beatles*, *Eindhoven*, *John F. Kennedy*).

For example, consider the following two sentences.

> *George W. Bush is the current president of the United States. He was born in New Haven, CT.*

We may consider *George W. Bush*, *current president*, *the United States* and *New Haven, CT* to be instances in the presented example. A task in information extraction could be to isolate these terms and identify their *class*, or the other way around: when given a class (e.g. *Location*), find the instances.

As we deal with natural language, ambiguities and variations may occur. For example, one can argue that the sequence *president of the United States* is a profession rather than *the current president* or *current president of the United States*.

Apart from identifying such entities, a second information extraction task may be to identify relations between the entities. The verb 'is' reflects the relation 'has profession' in the first sentence. To identify the place of birth, we have to observe that 'he' is an anaphora referring to *George W. Bush*.

Traditional information extraction tasks focus on the identification of named entities in large text corpora such as collections of newspaper articles or biomedical texts. In this thesis however, we focus on the web as a corpus.

Suppose that we are interested in a list of all countries in the world with their capitals. When we extract information from a collection of newspaper articles (e.g. three months of the New York Times), we cannot expect all information to be present. At best, we can try to discover every country-capital combination that is expressed within the corpus. However, when we use the web as a corpus, we can expect that every country-capital combination is expressed at least once. Moreover each of the combinations is likely to be expressed on various pages with multiple formulations. For example, *'Amsterdam is the capital of the Netherlands'* and *'The Netherlands and its capital Amsterdam (...)'* are different formulations of the same fact. In principle, we have to be able to interpret only one of the formulations to extract the country-capital combination. Hence, in comparison with a 'traditional' newspaper corpus, we can both set different objectives and apply different methods to extract information from the web.

With respect to the task of information extraction, the nature of this corpus has implications for the method, potential objectives and evaluation. In Table 1.1 the most important differences between the two can be found.

NEWSPAPER CORPUS	WEB CORPUS
No or fewer redundancy. Especially for smaller corpora, we cannot expect that information is redundantly present.	**Redundancy.** Because of the size of the web, we can expect information to be duplicated, or formulated in various ways. If we are interested in a fact, we have to be able to identify just one of the formulations to extract it.
Constant and reliable. In corpus-based IE, it is assumed that the information in the corpus is correct.	**Temporal and unreliable.** The content of the web is created over several years by numerous contributors. The data is thus unreliable and may be out-dated. Statements that are correctly extracted are not necessarily true or can be outdated.
Often monolingual and homogeneous. If the author or nature (e.g. articles from the *Wall Street Journal*) of the corpus is known beforehand, it is easier to develop heuristics or to train named entity recognizers (NERs).	**Multilingual and heterogeneous.** The web is not restricted to a single language and the texts are produced by numerous authors for diverse audiences.
Annotated test corpora available. In order to train supervised learning based named entity recognizers, test corpora are available where instances of a limited number of classes are marked within the text.	**No representative annotated corpora.** As no representative annotated texts are available, the web as a corpus is currently less suited for supervised machine learning approaches.

Static. Experimental results are independent of time and place as the corpora are static.	**Dynamic.** The contents of the web changes continuously, results of experiments may thus also change over time.
Facts only. Information Extraction tasks on newspaper corpora mainly focus on the identification of facts.	**Facts and opinions.** As a multitude of users contributes to the web, its contents is also suited for opinion mining.
Corpus is Key. In traditional information extraction, the task is to identify all information that can be found in the corpus. The information extracted is expected to be as complete as possible with respect to the knowledge represented in the corpus.	**Information Demand is Key.** As for many information demands the web can be expected to contain all information required, the evaluation is based on the soundness and completeness of extracted information itself.

Table 1.1: Comparison between the Web as a corpus and 'traditional' corpora.

1.2.2 Three Information Demands

We separate the information that can be extracted from the web into three categories: facts, inferable information and community-based knowledge.

Fact Mining

The first and probably most obvious category of information that can be extracted from the web is factual information. In this category we focus on the extraction of factual statements (e.g. *'Tom Cruise stars in Top Gun'*, *Brussels is Belgium's capital*). Such statements can be expected to be expressed within a single document or even within a sentence. Hence, the extraction of factual information focusses on the identification of a collection of factual statements, each expressed within a single document.

In Chapter 4, we focus on the extraction of such factual information from the web. We use the extracted information to get insights in the performance of our algorithms, as a ground truth is often available for these information demands.

Mining Inferable Data

An application domain other than factual data, is the extraction of *inferable* data from the web. Inferable data is not present as such on the web, but when it is discovered it can be recognized by human judges as true or relevant. We create such information by combining data from multiple sources. For example, the average price of an 19 inch LCD television in shops in Eindhoven can be identified by combining data from multiple web sites.

In Chapter 5, we discuss two information demands, where the required information is inferred from data extracted from the web. First, we present a method to extract lyrics from the web. Although many dedicated websites exist on this topic, it is not trivial to return a correct version of the lyrics of a given song. As many typo's, mishearings and other errors occur in the lyrics present on the web, there is need to construct a correct version using the various versions available. Such a correct version may even not be present on the web. When a user is given such a version however, it is relatively easy to judge the correctness.

The second application focuses on an information demand from a Dutch audiovisual archive. The collection of audiovisual material are annotated using a dedicated thesaurus, a list of keywords and their relations. To retrieve a particular document, knowledge on the content of this thesaurus is crucial. However, both professional users and the general audience cannot be expected to know each and every word that is contained in the thesaurus. Using web information extraction techniques, we present a method to link a given keyword to the term in the thesaurus with the closest meaning.

Community-based Knowledge Mining

The web is not only a well-suited text corpus to mine factual information. As a large community of users contributes to the contents of the web, it can also be used to mine more subjective knowledge. For example, we call *Paul Gauguin* a post-impressionist and related to *Vincent van Gogh*, *Christina Aguilera* a pop artist similar to *Britney Spears*. Such qualifications may not all be facts, but rather thoughts shared by a large community.

In the last part of this thesis (Chapter 6) we focus on methods to automatically find such internet community-based information. On the one hand we classify instances (e.g. *pop artists*) into categories and on the other hand identifying a distance matrix of related instances. The information found can be used to create an automated folksonomy: a knowledge base where items are tagged using implicit input from multiple users.

In restricted domains (e.g. *Movies*) for fact mining, the use of information extraction techniques for semi-structured information may be well usable. The *In-*

ternet Movie Database[1] for example is a reliable, semi-structured source to extract data on movies. When we are interested in subjective data based on opinions of the web community however, we cannot restrict ourselves to a single source. We combine data from multiple web sites, and thus multiple contributors, to characterize instances. We can however use semi-structured data from social websites as as *last.fm* as a benchmark on restricted domains like music [Geleijnse, Schedl, & Knees, 2007].

1.3 Related Work

We first focus on research on the extraction of information from semi-structured sources on the web. While the problem addressed is similar to the one in this thesis (i.e. extracting and combining information from multiple documents into a structured machine interpretable format), the source and therefore the methods differ.

In the second subsection, we focus on related research fields. Finally, Section1.3.3 focusses on previous work specific to web information extraction.

1.3.1 Gathering Information from Structured Sources

Information extraction from structured sources is thoroughly described in for example [Chang, Kayed, Girgis, & Shaalan, 2006] and [Crescenzi & Mecca, 2004]. These methods, 'wrappers', make use of the homogeneous lay-out of large web sites with pages that are constructed using a data-base.

As discussed in Section 1.1.1, web sites such as *amazon.com* and *imdb.com* make use of a database and present automatically generated web pages. The lay-out is uniform over the whole site, but the relevant information changes from page to page. For example, within an online music store, the information related to a particular album is page dependent. The performing artist, the title of the album and other catalogue data can be found on the exact same place on the page. The HTML-source of the two pages will also only differ at these places. For pages within a large web site, a wrapper algorithm can be created the information of interest from an arbitrary page within the site. Agichtein and Gravano [2000] make use of the homogeneous lay-out of large websites to extract information by first annotating a number of pages using a training set of known instances. Etzioni and others [2005] combine the extraction of information from unstructured sources with the identification of instances within tables. Shchekotykhin et al. [2007] describe a method to recognize tables on a specific domain (digital cameras and notebooks) and extract the information represented in these tables. In [Auer et al., 2007] structured text from *Wikipedia* is used to create semantic web content.

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

1.3.2 Related Fields and Tasks

In this subsection, we mention several tasks are closely related to web information retrieval.

Information Retrieval Information retrieval is often referred to as the task to return an (ordered) list of relevant document for a given query [Van Rijsbergen, 1979]. Kraaij [2004] gives an overview of commonly used models and techniques as well as evaluation methods for information retrieval.

A high quality document retrieval system is an essential aspect of an information extraction system as the retrieval of relevant documents or fragments is the first step in any large scale information extraction task.

In this work, we use a web search engine that retrieves relevant documents using an indexed collection of web pages [Brin & Page, 1998]. These pages are used to extract the information from the domain of interest. On the other hand, extracted information, such as given names, can be used to index documents in an information retrieval system.

Named Entity Recognition In the nineties, the Message Understanding Conferences (MUC) focused on the recognition of named entities (such as names of persons and organizations) in a collection of texts [Chinchor, 1998]. Initially, this work was mostly based on rules on the syntax and context of such named entities. For example, two capitalized words preceded by the string '*mr.*' will denote the name of a male person. As the creation of such rules is a laborious task, approaches became popular where named entities were recognized using machine learning techniques [Mitchell, 1997], for example in [Zhou & Su, 2002; Brothwick, 1999; Finkel, Grenager, & Manning, 2005]. However, such approaches typically make use of annotated training sets where instances (e.g. '*Microsoft*') are labeled with their class ('*Organization*'). For tasks where instances are to be recognized of other classes (e.g. the class *Movie* or *Record Producer*) annotated data may not be at hand.

The identification of more complex entities is studied by Downey et al. [2007]. With statistical techniques based on the collocation of subsequent words, terms such as movie titles are identified. Alternative rule-based approaches also give convincing results using the web as a corpus [Sumida, Torisawa, & Shinzato, 2006]. Schutz and Buitelaar [2005] focus on the recognition of relations between named entities in the soccer domain by using dependency parse trees [Lin, 1998].

Question Answering Question Answering is a task where one is offered a question in natural language [Voorhees, 2004]. Using a large text corpus, an answer to this question is to be returned. Although many variations in this task occur, typically the question is to be parsed to determine the type of the answer. For example,

the type of the answer for *Who killed John F. Kennedy?* is *person.* Based on the content of the corpus, a person name is to be returned. Question Answering also focusses on other types of questions with a more difficult answer structure (e.g. *Why did Egyptians shave their eyebrows?*), the shortest possible text fragment is to be returned [Verberne, Boves, Oostdijk, & Coppen, 2007]. Dumais et al. [2002] use the redundancy of information in a large corpus in a question answering system. Statements can be found at different places in the text and in different formulations. Hence, answers to a given question can possibly be found at multiple parts in the corpus. Dumais et al. extract candidate answers to the questions at multiple places in the corpus and subsequently select the final answer from the set of candidate answers.

Information extraction can be used for a question-answering setting, as the answer is to be extracted from a corpus [Abney, Collins, & Singhal, 2000]. Unlike question-answering, we are not interested in finding a single statement (corresponding to a question), but in *all* statements in a pre-defined domain. Functional relations, where an instance is related to at most one other instance, in information extraction correspond to factoid questions. For example the question *In which country was Vincent van Gogh born?*, corresponds to finding instances of *Person* and *Country* and the 'was born in'-relation between the two. Non-functional relations, where instances can be related to multiple other instances, can be used to identify answers to list questions, for example "name all books written by Louis-Ferdinand Céline" or "which countries border Germany?" [Dumais et al., 2002; Schlobach, Ahn, Rijke, & Jijkoun, 2007].

1.3.3 Previous work on Web Information Extraction

Information extraction and ontology constructing are two closely related fields. For reliable information extraction, we need background information, e.g. an ontology. On the other hand, we need information extraction to generate broad and highly usable ontologies. A good overview on state-of-the-art ontology learning and populating from text can be found in [Cimiano, 2006].

McCallum [2005] gives a broad introduction to the field of information extraction. He concludes that the accuracy of information extraction systems does not only depend on the design of the system, but also on the regularity of the texts processed.

The topic of hyponym extraction is by far the most studied topic in web information extraction. The task is given a term to either find it broader term (i.e. its hypernym), or to find a list of hyponyms given a hypernym. Etzioni and colleagues have developed KnowItAll: a hybrid web information extraction system [2005] that finds lists of instances of a given class from the web using a search engine. It combines hyponym patterns [Hearst, 1992] and learned patterns for instances of the

class to identify and extract named-entities. Moreover, it uses adaptive wrapper algorithms [Crescenzi & Mecca, 2004] to extract information from html markup such as tables. KnowItAll is efficient in terms of the required amount of search engine queries as the instances are not used to formulate queries. In [Downey, Etzioni, & Soderland, 2005] the information extracted by KnowItAll is post-processed using a combinatorial model based on the redundancy of information on the web.

The extraction of general relations from texts on the web is recently studied in [Banko, Cafarella, Soderland, Broadhead, & Etzioni, 2007] and [Bunescu & Mooney, 2007]. Craven et al. manually labeled instances such as person names and names of institutions to identify relations between instances from university home pages. Recent systems use an unsupervised approach to extract relations from the web. Sazedj and Pinto [2006] map parse trees of sentences to the verb describing a relation to extract relations from text.

Cimiano and Staab [2004] describe a method to use a search engine to verify a hypothesis relation. For example, if we are interested in the 'is a' or hyponym relation and we have the instance *Nile*, we can use a search engine to query phrases expressing this relation (e.g. *"rivers such as the Nile"* and *"cities such as the Nile"*). The number of hits to such queries is used to determine the validity of the hypothesis. Per instance, the number of queries is linear in the number of classes (e.g. *city* and *river*) considered.

In [De Boer, Someren, & Wielinga, 2007] a number of documents on art styles are collected. Names of painters are identified within these documents. The documents are evaluated by counting the number of painters in a training set (of e.g. *expressionists*) that appear in the document. Painters appearing on the best ranked documents are then mapped to the style. De Boer et al. use a training set and page evaluation, where other methods simply observe co-occurrences [Cilibrasi & Vitanyi, 2007].

A document-based technique in artist clustering is described in [Knees, Pampalk, & Widmer, 2004]. For all music artists in a given set, a number of documents is collected using a search engine. For sets of related artists a number of discriminative terms is learned. These terms are used to cluster the artists using support vector machines.

The number of search engine *hits* for pairs of instances can be used to compute a semantic distance between the instances [Cilibrasi & Vitanyi, 2007]. The nature of the relation is not identified, but the technique can for example be used to cluster related instances. In [Zadel & Fujinaga, 2004] a similar method is used to cluster artists using search engine counts. In [Schedl, Knees, & Widmer, 2005], the number of search engine hits of combinations of artists is used in clustering artists. However, the total number of hits provided by the search engine is an estimate and not always reliable [Véronis, 2006].

In [Pang, Lee, & Vaithyanathan, 2002; Dave & Lawrence, 2003; Kim & Hovy, 2004; Pang & Lee, 2005] methods are discussed to identify opinions on reviewed products. For example, given is a set of reviews of some flat screen television mined from the web. The task is to assign a grade to the product or its specific features (e.g. the quality of the speakers).

The extraction of social networks using web data is a frequently addressed topic. For example, Mori et al. [2006] use *tf·idf* (see [Salton & Buckley, 1988; Manning & Schütze, 1999]) to identify relations between politicians and locations and Jin, Matsuo and Ishizuka [2006] use inner-sentence co-occurrences of company names to identify a network of related companies.

1.4 Outline

This thesis is organized as follows. In the next chapter, we formulate the problem and give an outline of the method to extract information from the web. This method gives rise to two subproblems, on the one hand the identification of relations in texts and on the other hand the identification of the terms and given names of interest. We will discuss these subproblems in Chapter 3. To obtain evidence for the applicability of the methods discussed in this thesis, in Chapter 4 we present a number of case-studies, where we extract factual information from the web. Chapter 5 focusses on two applications of web information extraction. Contrary to the case-studies in Chapter 4, the information extracted here cannot be found in structured sources. Chapter 6 handles the extraction of community-based data from the web, where we find *tags* for a set of instances. Finally, the conclusions can be found in Chapter 7.

2

A Pattern-Based Approach to Web Information Extraction

In this chapter we present a global outline for an approach to extract information from the web. Hereto we first define a formal model for the concept 'information'. Next, we discuss the design constraints that are specific for both the corpus, i.e. the web, and the use of a state-of-the-art search engine. Based on the design constraints, a global method to extract information from the web is presented.

2.1 Introduction

In this section, we first focus on a model to represent information. Using the definitions provided in Section 2.1.2, we formulate our problem definition in Section 2.1.3.

2.1.1 A Model for 'Information'

Finding a suitable representation of information is one of key tasks in computing science. We call data information, when it has a meaning. That is, when it can be used for some purpose, for example the answering of questions.

To represent the concept information, we let ourselves be inspired by the semantic web community. This community uses the concept *ontology*, which is defined by Gruber as 'a specification of a conceptualization' [1995]. Wikipedia pro-

vides a perhaps somewhat more practical definition: 'ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts'[1].

In the semantic web languages, an information unit or *statement* consists of a triplet of the form subject - predicate - object, for example *Amsterdam - is capital of - the Netherlands* or *the Netherlands - has capital - Amsterdam*. Analogous to the object-oriented programming paradigm, we speak of *classes* and their *instances*. Note that in this model *instances* are part of the ontology. This allows us to express knowledge on concepts such as *Amsterdam* and their domains (*City*), but also enables us to express relations between concepts. As the predicates can be as refined as required, this model can be used to express statements that are more complex.

The semantic web languages OWL and RDFS enable the formulation of properties of classes and relations. These languages are rich [Smith, Welty, & McGuinness, 2004], but complex [Ter Horst, 2005]. In this work, we opt for a simple formalization as the focus of this work is on the extraction of information, rather than on the use of the extracted information. We note that constructs that allow reasoning, such as axioms and temporal properties are not included in this formalism.

An *initial ontology* serves three purposes.

1. It is a specification of a domain of interest. Using the classes and relations, the concepts of interest are described. A domain is specified by defining the relevant classes (e.g. *City*, *Capital*) and relevant relations (e.g. *is located in* defined on classes *City* and *Country*).
2. The ontology is used to specify the inhabitants of the classes and relations: the formalizations of the statements describing the actual instances and their relations. For example, *Amsterdam* is an instance of the class *Capital* and the pair (*Amsterdam*, *the Netherlands*) may be a relation instance of *is located in*.
3. We use the ontology the specify an information demand. By defining classes and their instances as well as relations and relation instances, we model the domain and indicate the information that is to be extracted from the web.

Now suppose we are interested in a specific piece of information, for example: *the Capital of Australia*, *artists similar to Michael Jackson*, *the art movements associated with Pablo Picasso* or *the profession Leonardo da Vinci is best known for*. We assume that such information can easily be deduced from an ontology that contains all relevant data. The aim of this work is to automatically fill, or *populate*,

[1] `http://en.wikipedia.org/` article: Ontology (Computer Science), accessed December 2007.

an ontology that describes a domain of interest. We hence focus on populating an ontology on the one hand with instances and on the other hand with pairs of related instances.

2.1.2 Definitions and Problem Statement

The semantic web languages are created to describe information in a machine readable fashion, where each concept is given a unique, unambiguous descriptor, a universal resource identifier (e.g. `http://dbpedia.org/resource/-Information_extraction` is the URI for the research topic of *Information Extraction*). By reusing the defined URIs, distributed content can be linked and a connected knowledge base is built.

For reasons of simplicity we abstract from the semantic web notations. By keeping the definitions simple, the notations introduced in this thesis can be translated into the semantic web languages with fair ease, as we maintain the subject - predicate - object structure used in the semantic web languages.

We define an ontology O as follows.

Definition [Ontology]. An ontology O is a pair (C,R), with C the set of classes and R the set of relations. □

Definition [Class]. For ontology O, we define class $c_j \in C$ as $c_j = (n,I,b)$, where
n is the string giving the name of the class,
I gives the set of *instances* of the class, and
$b \in \{true, false\}$, a boolean indicating whether c_j is complete. □

Hence, each class is assigned a unique name (e.g. *Location*, *Person*) and a set of instances. As the initial ontology is used to specify the information demand, we use b to indicate whether we consider the class to be complete, i.e. whether *all* relevant instances in I are given. Note that a class c_j with $b \equiv$ true does not need to be complete in an absolute sense, but that the completeness of c_j indicates that there is no demand to find additional instances for the class. To refer to the set of instances of class c_j, we will use I_j as a shorthand notation.

Definition [Instance]. For a class c_j, an instance $i \in I$ is defined by the string representing the instance. □

We consider instance i to be an inhabitant of a class named n, if the statement *"i is a n"* (e.g. *Eindhoven is a city.*) is true. Hence, the name of the class defines its semantics. We assume that within a given class (e.g. *Person*), the string i uniquely identifies the instance.

Apart from classes, we also consider a set of relations R.

Definition [Relation]. For ontology O, a relation $r_a \in R$ as $r_a = (n, c_s, c_o, \varphi, J)$, with
n is the string representing the name of the relation,
c_s is the subject class, $c_s \in C$,
c_o is the object class, $c_o \in C$,
$\varphi \in \{true, false\}$, indicating whether the relation is functional, and
J is the set of *relation instances*, $J \subseteq I_s \times I_o$. □

A relation can be conveniently expressed as the triplet *[c_s] n [c_o]*. For example, *[person] was born in [city]* is instantiated with *[Vincent van Gogh] was born in [Zundert]*.

For *non-functional* relations (i.e. $\varphi \equiv false$), instances in the subject class can be related to multiple instances in the object class. For example, a person may have multiple professions, a painter can belong to more than one art movement and *Radiohead* can be considered to be related to various other musical artists. For some relations on the other hand, the number of instances in the object class related to a subject instance may be restricted. In practice, this distinction is viable for all relations considered in this work. We will return to the consequences of this choice in Chapter 3.

Finally, we define the relation instances.

Definition [Relation Instance]. For relation $r_a = (n, c_s, c_o, J)$ in ontology O, a relation instance $j \in J$ is a pair (i, i'), where
i is an instance of the subject class c_s, and
i' is an instance of the object class c_o. □

We consider relation instance j to be an inhabitant of a relation named n, if the statement "i_s n i_o" (e.g. *Eindhoven is located in the Netherlands*) is true.

In Figure 2.1 an example ontology is visualized. Relations are considered between instances in the central class *person* and instances in all the other classes.

2.1.3 The Ontology Population Problem

As stated in the introductory chapter, we restrict ourselves to using natural language texts on the web. Before we focus on the actual process of extracting information from such texts, the task is how to find potentially relevant texts. For information extraction tasks with a large collection of documents, the use of a document retrieval system is necessary to identify relevant texts.

As the web is a collection of billions of text documents, there is need to select

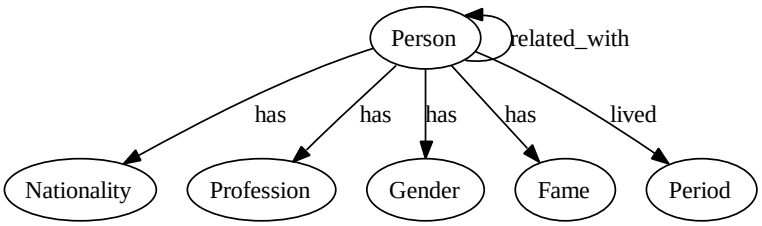

Figure 2.1. An example ontology on historical persons.

potentially relevant documents or document fragments. As we consider document retrieval a separate concern, we chose to use an off-the-shelf search engine. Using a search engine, we hence need to formulate queries that result in relevant documents. Having retrieved a relevant document, we can focus on the extraction of information, i.e. populating the initial ontology. We consider the following two subproblems in ontology population from texts on the web using a search engine.

The Class Instantiation Problem. Given an initial ontology O with class c_j, identify instances of c_j using texts found with a web search engine. □

The Relation Instantiation Problem. Given an initial ontology O, with relation $r = (n, c_s, c_o, \varphi, J)$ find relation instances $(i, i') \in I_s \times I_o$. □

These two subproblems in information extraction are combined in the ontology population problem.

The Ontology Population Problem (OP). Given an initial ontology O, instantiate the classes and relations by extracting information from texts on the web found with a search engine. □

Given an initial ontology O, we use O' to refer to the populated ontology.

Popular search engines currently only give access to a limited list of possibly interesting web pages. A user can get an idea of relevance of the pages presented by analyzing the title and a snippet presented. When a user has sent an accurate query to the search engine, the actual information required by the user can already be contained in the snippet.

If these snippets and titles are well usable for web information extraction purposes, the documents themselves do not have to be downloaded and processed. We hereto formulate the following alternative problem description.

The Snippet Ontology Population Problem (SOP). Given an initial ontol-

ogy O, instantiate the classes and relations by extracting information from search engine snippets. □

2.1.4 Evaluating a Populated Ontology

Having populated an ontology, we want to obtain insight in the quality of the information extracted in terms of soundness and completeness. That is, the extracted information on the one hand needs to be correct and on the other hand as complete as possible.

Hereto, we use the standard measures precision and recall. To measure precision and recall, we assume a ground truth ontology O_{ref} to be given.

For the set $O'(I_j)$ of instances of class c_j found in the populated ontology O', we define *precision* and *recall* as follows.

$$precision(c_j) = \frac{|O_{\text{ref}}(I_j) \cap O'(I_j)|}{|O'(I_j)|}$$

and

$$recall(c_j) = \frac{|O_{\text{ref}}(I_j) \cap O'(I_j|)}{|O_{\text{ref}}(I_j)|}.$$

We formulate similar measures for the relations r in R.

$$precision(r) = \frac{|O_{\text{ref}}(J) \cap O'(J)|}{|O'(J)|}$$

$$recall(r) = \frac{|O_{\text{ref}}(J) \cap O'(J)|}{|O_{\text{ref}}(J)|}.$$

The standard objection function in the field of information retrieval to combine precision and recall is the F-measure [Van Rijsbergen, 1979; Voorhees, 2005]. If precision and recall are equally weighted, i.e. considered to be of the same importance, F is defined as follows.

$$F(c_j) = \frac{2 \cdot \text{precision}(c_j) \cdot \text{recall}(c_j)}{\text{precision}(c_j) + \text{recall}(c_j)} \tag{2.1}$$

Using the parameter α, the F measure is generalized as F_α, where $F = F_1$.

$$F_\alpha(c_j) = \frac{(1+\alpha)\cdot \text{precision}(c_j)\cdot \text{recall}(c_j)}{\alpha\cdot \text{precision}(c_j) + \text{recall}(c_j)} \tag{2.2}$$

The F-measures for evaluating the populated relations are formulated similarly.

As discussed, to measure precision and recall a ground truth ontology is required. For some information demands, we can not expect such an ontology or any other form of structured data to exist. Moreover, information extraction tasks with a known ground truth are not very interesting from an application point of view.

In cases where no ground truth is available, precision is typically estimated by manually inspecting a sample subset of the instances found. Recall is estimated using an (incomplete) set of known instances of the class. For example, if we are interested in an ontology with musical artists, a complete list of such artists is not likely to be known. However, we can compute the recall using a set of known or relevant instances (e.g. famous musical artists extracted from structured sources such as *Last.fm* or *Wikipedia*) and express the recall using this list.

A separate aspect of the evaluation is the notion of *correctness*. We cannot assume that all correctly extracted statement are indeed true. However, based on the expected redundancy of information on the web, we expect factual information to be identifiable.

More complex to evaluate are subjective relations, such as the relation between a musical artist and a genre as regarded by the web community. Nevertheless may the use of web information extraction techniques be valuable for such information demands, as subjective information is less likely to be represented in a structured manner. We return to this topic in Chapter 6.

2.2 Extraction Information from the Web using Patterns

The ontology population problem can be split into two concerns.

- We need to compose a strategy to retrieve relevant text.
- We have to focus on a method to actually extract information from these texts.

We will argue that choosing a strategy to retrieve documents influences the process of extracting information from these documents. In this section, we will discuss a global method to populate an ontology (i.e. to extract information) using texts retrieved with a search engine. The strategy chosen to formulate search engine queries effects the method to extract information from the texts retrieved.

Before presenting an ontology population algorithm in Section 2.2.2, we first discuss the consequences of choosing a commercial search engine and the web as a corpus.

2.2.1 Design Constraints

The use of a commercial search engine and the nature of the texts on the web lead to requirements that constrain the design of a method to extract information from the web.

Search Engine Restrictions

In this thesis, we use a search engine that provides us with 'the most relevant' pages on the web for a given query. As the web is a collection of billions of changing, emerging and disappearing pages, it is infeasible to extract information from each and every one of them. As we hence need a reliable web document retrieval system, we use a state-of-the-art search engine to find relevant documents. The design of such a search engine is a separate concern and outside the scope of this thesis. Therefore, we choose to use such commercial search engines for our purposes. Using search engines like *Yahoo!* or *Google* also facilitates the reuse of the methods developed, as programmer's interfaces are provided.

The use of a (commercial) search engine also has important disadvantages.

- A query sent to the search engine from two different machines can give different search results, as the services of large search engines are distributed.
- The search results differ over time, as the web changes and the pages indexed and ranked are continuously updated.

Hence, an experiment making use of a distributed search engine can give different results when conducted at any other time or place. For this reason, the use of static corpora as test sets in information extraction are currently the only basis to objectively compare experimental results. Hence, experimental results of alternative approaches in web information extraction are hard to compare.

In the first chapter, we give a comparison between static corpora and the Web as a corpus. We choose not to test our methods on static corpora to benchmark the performance with other methods, as our method is specifically designed for the characteristics of the Web. However, where possible we do compare our web information extraction approach with work by others.

An initiative where a *snapshot* of the web is stored and indexed would be a stimulus for the field of web information extraction. Such a time and location independent search engine would facilitate a reliable comparison of alternative approaches in web information extraction.

Currently, both *Google* and *Yahoo!* allow a limited amount of automatic queries per day. At the moment of writing this thesis, *Google* allows only 1,000 queries a day, where each query returns at most 10 search results. Hence if for a given query expression the maximum of 1,000 search results are available, we need to formulate 100 queries using the *Google* API. *Yahoo!* currently is more generous, allowing 5,000 automated queries per day, where at most 100 search results are returned per query.

Hence, this search engine use restriction requires us to analyze our approach not only in terms of time and space complexity, but also in terms of the order of number of queries, which we termed the *Google Complexity*.

Definition [Google Complexity]. For a web information extraction algorithm using a search engine, we refer to the required number of queries as the Google complexity. □

In this thesis, we will analyze the Google Complexity in terms of the required number of queries for the populated ontology O'.

To restrict the Google complexity, we need accurate queries for which we can expect the search engine to return highly relevant information. The actual requirements depend on the application of the data. If the collection of information is a single time effort, a run time of a couple of days would be acceptable. However, for real-time or semi real-time applications, a more efficient approach is required.

Design Constraint. The Google complexity of the approach chosen to populate an ontology should be such that the algorithm terminates within days. □

In this chapter, we present a method with a Google complexity that is linear in the size of the output. In Chapter 5, we focus on two applications of web information extraction, with a constant Google complexity.

Limitations on Text Processing

Having retrieved a potentially relevant document from the web, the task is to identify relevant instances and their relations. Traditionally, approaches in information extraction (and natural language processing in general) can be split into data-oriented and knowledge-oriented ones.

In a data-oriented information extraction approach, instances and relations are typically recognized using an annotated training set. In a *representative* text corpus, relevant information such as part-of-speech tags, dependency parse trees and noun phrases are signaled. These annotated texts are used to train a machine learning

algorithm to classify a text without annotations, the test set. The assumption is used that instances of the same class appear in a similar context, are morphologically similar, or have the same role in the sentence.

In a knowledge-oriented approach on the other hand, we create a model to recognize instances and relations in texts. We hence use our own knowledge of language to create recognition rules. For example, we could state that two capitalized words preceded by *mr.* indicate the name of a male person.

Using either a data- or knowledge-oriented approach to populate an ontology, the approach is to be domain dependent. The annotations or rules that are used to recognize some class c_j (e.g. *Movie*, *Musical Artist*) cannot be used to recognize instances of some other class (e.g. *Person*). An additional problem for a data-oriented approach is the lack of available annotations.

Supervised data-oriented approaches in natural language processing make use of a representative training corpus. The text in this training corpus is annotated for the specific NLP task, for example part-of-speech tagging [Brill, 1992] or the identification of dependencies within a sentence [Lin, 1998; Marneffe, MacCartney, & Manning, 2006]. Such low level features are commonly used in information extraction methods [Collins, 2002; Etzioni et al., 2005]. The common annotations for information extraction in the available corpora focus on standard, restricted named entity recognition tasks, such as the recognition of person names, companies and – in the biomedical domain – protein names. The more regular a corpus is, the better a system performs on a given NLP task [McCallum, 2005].

The web texts found with a search engine and especially the snippets are irregular as they are multilingual and contain typo's and the broken sentences. Due to the irregularity of the texts and the lack of representative training data, it is therefore not likely that low level features like parts-of-speech can be identified reliably. An additional problem is that annotated training data is not available for all the class instantiation tasks we are interested in.

Given these considerations, we choose not to make use of manually annotated training data and off-the-shelf systems that are trained on such data. Hence, to opt for a generic approach in ontology population, we formulate the following constraint.

Design Constraint. To facilitate a generic approach, we do not make use of manually annotated training data. □

In Chapter 4 we return to this topic, where we evaluate the use of an off-the-shelf supervised named entity recognizer to identify person names in snippets.

In the next chapter, taking this design constraint into account, we discuss options in rule-based and unsupervised machine learning approaches in ontology pop-

ulation.

2.2.2 Sketch of the Approach

In this section, we present a global approach to populate an initial ontology O. As discussed earlier in this chapter, we are confronted with the design constraint that the availability of the search engines is limited. This enforces us to formulate precise queries, in order to obtain highly relevant search results.

Now, if an ontology with complete classes is given, the task is to only populate the relations, i.e. to find relation instances. In other words, we have to find and recognize natural language formulations of the *subject – relation – object* triplets.

If we are interested in the class instantiation problem, the tasks are quite similar. For a class named n, the task is to find terms t where the triplet *t is a n* is expressed. Hence, the class instantiation problem can easily be rewritten into a relation instantiation problem for incomplete classes.Suppose we are handed the following class instantiation problem: $O = (\{c_j\}, \emptyset)$ with $c_j = (n, I, b)$. We now can rewrite the problem into a relation instantiation problem for incomplete classes, by creating a new class c_j with the name of class c_j as the only instance. A relation r_k is introduced to express the original inhabits (or is-a) relation between the instances and the class itself. That is, $O = (\{c_j, c_i\}, r)$ with $c_j = (n', I, b)$, $c_j = (n'', \{n\}, \text{true})$ and $r = (\text{is a}, c_j, c_i, \text{true}, J)$, with $J = \{(a,b) | b = n \wedge a \in I_j\}$.

Without loss of generality we can thus focus on an approach to solve the incomplete relation instantiation problem here. We will focus on the identification of statements containing a subject – relation – object triplets.

A common approach in web information extraction is to formulate queries consisting of all pairs of the names of known instances of the subject and object classes. The number of hits is used by Cilibrasi and Vitanyi [2007] to compute a distance between instances, while Mika creates a network of related instances in a similar fashion [Mika, 2007]. Knees et al. [2004] use the total number of search results (i.e. the numbers of *hits*) of queries with two instances to classify musical artists. Gligorov et al. [2007] use the number of hits of combinations of instances from two separate ontologies as a distance measure used in ontology mapping. De Boer et al. [2006] use combinations of names of art styles and periods to create a mapping between the two.

Hence, if we are interested in the relation named *was born in* and the subject class c_s containing the instance *John F. Kennedy*, we can combine this instance with all instances in object class c_o into queries. The search results are then processed in some fashion to identify evidence for the *was born in* relation between the queried instances.

Although this approach is a straightforward method to collect relevant texts on the web, we observe the following drawbacks.

- *Large number of queries.* This approach leads to $|I_s| \cdot |I_o|$ queries and has therefore in general no Google complexity linear in the total number of instances.
- *Not generally applicable.* As such an approach assumes the classes to be complete, it cannot be used to solve the general ontology population problem for incomplete classes.
- *No solution for relation identification.* The co-occurrence of two instances in a document does not necessarily reflect the intended relation. Hence, either the query needs to be more specific [Cimiano & Staab, 2004] or the documents need to be processed [Knees et al., 2004].

As an alternative, we formulate queries containing *one* known instance. Such an approach would lead to a Google complexity linear in the number of instances in O', if we formulate a constant number of queries per instance. Having formulated a query containing an instance, the texts retrieved by the search engine are to be processed to recognize an instance of the other class and evidence for the relation between the two.

A very simple language model. The web as a corpus – and especially the collection of snippets returned by a search engine – is multi-lingual and contains typo's, broken sentences, slang, jokes, and other irregularities. As no representative annotations or reliable tools are available for such data, we choose to opt for a very simple language model to identify instances and their relations.

We focus on sentences where the instances of the subject and object class are related by a small text fragment. We ignore the rest of the context. Given a relation r_a, we use short text fragments that are commonly used to express the relation of interest. For example, the text fragment *was born in* is an often used expression to express the relation between a person and his place of birth. We refer to these frequently occurring text fragments as patterns.

Design Constraint. We recognize a relation between two instances if and only if the two instances are connected by one of the predefined text fragments. □

Of course, a relation between two instances can be formulated in numerous manners and such formulations can be found in various other ways, e.g. using anaphora, in multiple sentences etc. Hence, if we would be interested to find each and every occurrence of an expression of the intended relation, this method might not be the best possible choice. However, as we use the web as a corpus, we make use of the *redundancy of information*. We expect that important concepts and relations occur in various formulations on the Web. As we are interested to find *at least one*

formulation of a subject – relation – object triplet on the Web, we do not have to recognize *every* relevant statement encountered.

Making use of the redundancy of information, the chosen language model is a powerful mechanism to formulate precise and effective queries. By combining an instances and a pattern into a query (e.g. *John F. Kennedy was born in*, we generate very relevant search results. The locations extracted in the search results are used to simultaneously populate the class and the relation.

In related work, Etzioni et al. [2005] propose a method to combine patterns with class names into queries to populate the given classes. The identification of hyponyms using combined instance-pattern queries is discussed in [Tjong Kim Sang & Hofmann, 2007].

We combine a pattern and a known instance into a search engine query. The patterns are stored with placeholders for instances of the classes. For example, for the relation *born in* with classes *Person* and *Location*, the following subject - pattern - object triplets can be identified: *[Person] was born in [Location]* and *[Location] is the place of birth of [Person].* In the given examples, *[Location]* and *[Person]* serve as placeholders for the instances of the corresponding classes. When querying the pattern in combination with a subject instance, the object instance is to be recognized in the position of the object class placeholder and vice versa.

Hearst [1992] coined a simple technique to identify the relations between two terms in a text. She identified a number of frequently used text fragments – patterns – that connect a word and its hyponym. The running example in this paper is the following sentence.

> The bow lute, such as the Bambara ndag, is plucked and has an individual curved neck for each string.

From this example sentence, we learn that a *Bambara ndag* is a kind of *bow lute*. Hence, to *extract* the hyponym relation between *bow lute* and *Bambara ndag* no context is required but the text fragment in between the two terms. Moreover, no knowledge or any other background information on Bambara ndags or bow lutes is required to identify the relation between the two. Hearst identified the six patterns as given in Table 2.1.

The preselected patterns in [Hearst, 1992] are used in various web information extraction systems, for example [Ciravegna, Chapman, Dingli, & Wilks, 2004; Etzioni et al., 2005; Sumida et al., 2006; McDowell & Cafarella, 2006; Pantel & Pennacchiotti, 2006].

We expect information to occur redundantly on the web. Although we do not need to recognize every formulation of a given fact, we can expect to extract instances and relation instances from multiple different texts. We can use the redundancy of information on the web to filter the extracted data. Not all extracted

[hypernym] such as [hyponym]
such [hypernym] as [hyponym]
[hyponym] or other [hypernym]
[hyponym] and other [hypernym]
[hypernym] including [hyponym]
[hypernym] especially [hyponym]

Table 2.1. Patterns for instance-class relation.

data can be assumed to be correct. Extracted statements can be erroneous for two reasons. On the one hand because the context influences the semantics of the *instance - pattern - instance* phrase. For example, consider the sentence *Some people think that Sydney is the capital of Australia*, where the context suggests that the triple *Sydney - is the capital of - Australia* is not a true fact. On the other hand, the information provided can simply be false.

As a consequence of the redundancy of information on the web, we assume that a *instance - pattern - instance* phrase will most often express the corresponding relation in the ontology. However, as we ignore the context of the subject - pattern - object phrase, erroneous or misinterpreted data can be extracted. For example, suppose we would extract *Canberra* to be Australia's capital from 30 documents on the web, while *Sydney*, *Wellington* and *Canbera* are identified only a couple of times as such. Based on these figures, we filter out the erroneously extracted data.

Sketch of Algorithm. Given is an initial ontology describing the domain of interest. For each relation $r \in R$ in the ontology we assume given a non-empty set $\mathcal{P}(r)$ of patterns expressing r and a non-empty set of instances for either the object or the subject class. Using a known instance and a pattern, we can formulate queries that potentially lead to relevant texts.

Using an ontology O that meets the requirements, we populate O using the following approach. We iteratively select a relation r in R (e.g. *born in*) and a pattern $\mathcal{S}$ corresponding to this relation (e.g. '*was born in*'). We then select a class, i.e. either the subject or the object class for r, and take a known instance from this class (e.g. *Alan Turing* from the subject class *Person*). The selected instance and pattern are then combined into a search engine query (*Alan Turing was born in*). Subsequently, we extract instances of the unqueried class from the search results. This procedure is continued until no unqueried instance-pattern pairs exist. New patterns can be learned by analyzing texts containing newly extracted relation instances. Using newly learned patterns, the ontology population procedure can be

do ¬ stop criterion →
  do $\exists_{r,c_j}\exists_{i\in I_j,\ \mathcal{S}\in\mathcal{P}(r)}$ "$i-\mathcal{S}$ combination unqueried" →
    combine pattern $\mathcal{S}$ and instance i into query ;
    collect snippets or documents from the search results ;
    extract instances of the related class c_k from search results ;
    store the extracted instances i' in class c_k ;
    store the extracted relation instances $(i,i') \in I_j \times I_k$ in relation r;
  od
find new patterns for the relations in R ;
od

Table 2.2. Sketch of the ontology population algorithm.

repeated. In Table 2.2 we give an overview of the approach in pseudo-code.

When initially no patterns are provided, the algorithm can be used to identify patterns. However, in that case, non-empty sets of relation instances are required.

As a stop criterion we simply use a fixed number of iterations. The extraction of the instances in the texts as well as the identification of patterns can be studied in isolation and are the topics of the next chapter.

Google complexity. As extracted instances are used as queries, one can easily observe that the Google complexity of the approach cannot be expressed in terms of the size of the input, the initial ontology O. However, the Google complexity can be expressed in terms of the size of the populated ontology O'.

After the termination of the algorithm, each instance in the output ontology has been queried with every matching pattern. Suppose we have $\text{pat}(r_a)$ patterns for relation r_a, then the total number of queries N_q in the population phase can be expressed as follows.

$$N_q = \sum_{r_a \in R} \text{pat}(r_a) \cdot (|I_s| + |I_o|) \tag{2.3}$$

Hence, assuming that a constant number of queries is used, the Google complexity is linear in the sum of the sizes of the sets of instances in the populated ontology.

Bootstrapping. It is notable that the algorithm features multiple bootstrapping mechanisms. For an ontology with incomplete classes, the following bootstrapping

steps apply.

- The instances extracted for relation r_a are used as queries to populate other relations. For example, using the ontology in Figure 2.1, we can find instances in class Person after querying phrases containing instances of period. We use the persons found, to identify relations between person and profession and potentially find new professions.
- The extracted relation instances can be used to find new relation patterns. In Section 3.1 we present a mechanism to identify patterns using a set of relation instances. When we expand the set of relation instances, we can find other or more reliable relation patterns. These patterns can then be used to populate the ontology.
- The texts can be used to identify new instances. Using queried texts containing known instances, we can learn to recognize the morphology and context of instances. We focus on this task in Section 3.2.

Hence, we created a framework, where starting with only few instances we can populate a full ontology on a domain of interest. The approach as discussed in this section, leaves two issues unresolved: the identification of instances from text and the identification of patterns. These topics are the focus of the next chapter.

3

Two Subproblems in Extracting Information from the Web using Patterns

In the previous chapter, we proposed a pattern-based method to extract information from the web using a search engine. After having presented a global outline, we identified two subproblems to be resolved. In this chapter, we study these problems in isolation. We first focus on the automatic identification of relation patterns in Section 3.1. Section 3.2 focusses on several alternative approaches to identify instances from text.

3.1 Identifying Effective Patterns

Ravichandran and Hovy [2002] present a method to automatically identify *surface text patterns* expressing relations between pairs of terms using a search engine. Based on a training set of relation instances, their method identifies natural language patterns that express some relation between two instances. For example, "was born in" showed to be a one of the patterns expressing the relation between instances Mozart (of class *Person*) and 1756 (of class *Year*). This pattern proved to be precise as many of the search results for the query *Mozart was born in* showed to contain the instance *1756*.

Using the terminology defined in the previous chapter, the algorithm proposed by Ravichandran and Hovy can be sketched as follows. Given is an ontology $O =$

for $(i, i') \in J \ \rightarrow$
    combine the two terms as a search engine query: i , i' ;
    collect the sentences in the search results containing both i and i' ;
    replace instances by placeholders for the corresponding classes ;
    store the text fragments $\mathcal{S}$ in a set P ;
rof
initialize $c(\mathcal{S})$ and $n(\mathcal{S})$ to 0 for all $\mathcal{S} \in P$;
for $(i, i') \in J \ \rightarrow$
    query i ;
    collect all sentences in the search results that contain both instances ;
    for $\mathcal{S} \in P \ \rightarrow$
        $c(\mathcal{S}) = c(\mathcal{S}) +$ number of occurrences of $\mathcal{S}$ with i';
        $n(\mathcal{S}) = n(\mathcal{S}) +$ the total number of occurrences of $\mathcal{S}$;
    rof
rof
for $\mathcal{S} \in P \ \rightarrow$
    compute precision $f_{\text{pr}}(\mathcal{S}) = \frac{c(\mathcal{S})}{n(\mathcal{S})}$;
rof
select the most precise ones using the scores f_{pr} such that $c(\mathcal{S}) \geq 5$;

Table 3.1. Sketch of the pattern identification algorithm proposed by Ravichandran and Hovy.

$(\{c_s, c_o\}, \{r\})$, with the set of relation instances J non-empty. The identification of patterns is done in two phases: a collection and an evaluation phase.

In the collecting phase of Ravichandran and Hovy's algorithm, queries are formulated to identify potential patterns expressing the relation r. Subsequently, the collected text fragment are evaluated to select the most precise ones. A sketch of this algorithm is given in Table 3.1.

The algorithm presented is used in a question-answering setting for so-called *factoid* questions [Voorhees, 2004]. Using the terminology introduced in the previous chapter, such questions correspond to functional relations.

We address the issue of extracting patterns, since we observed a number of drawbacks of Ravichandran and Hovy's work with respect to the application of such patterns in a more general information extraction setting.

- Ravichandran and Hovy focus only on functional relations. In a general information extraction setting, we cannot assume that all relations are functional.
- The use of precise patterns can lead to a low recall of relevant search results. The criterion for selecting patterns, precision, is therefore not the only basis for a pattern to lead to relevant search results. Although Ravichandran and Hovy use a threshold to filter out rare phrases, for the more frequently occurring phrases, precision is the only selection criterion.
- When querying an arbitrary instance, the probability of retrieving sentences that both contain the unqueried instance as well as one of the predefined patterns is not very high.

Hence, we propose both different evaluation criteria as well as an adapted mechanism to collect and evaluate the patterns. We present a domain-independent method to identify *effective* rather than precise patterns representing relations. We call a pattern effective, if it links many instance-pairs in the snippets found with a search engine. Hence, the use of an effective pattern should lead to snippets containing instances in the related class with high levels of precision and recall. The identification of effective patterns is important, since we want to perform as few queries to a search engine as possible to limit the use of its services.

3.1.1 Problem Description

We are interested to identify effective patterns for a given relation between two classes. To discover such patterns, we require the set of relation instances J to be non-empty.

Hence, using the terminology as posed in Chapter 2, we consider an ontology O with one single relation, i.e. $O = (\{c_s, c_o\}, \{r\})$, with J non-empty. Here, $r = (n, c_s, c_o, J))$. We do not require that $c_s \neq c_o$.

The Effective Pattern Extraction Problem. Given is an ontology O with relation r and a non-empty set of relation instances for r. Identify effective patterns that express relation r. □

For example, we consider the classes with names *Author* and *Book Title* and the relation named *has written*. We assume that J contains some relation instances, e.g. (*Leo Tolstoy*, *War and Peace*) and (*Günter Grass*, *Die Blechtrommel*). The aim is then to find natural language phrases that relate authors to the titles of their books, such as the simple pattern *wrote*. Thus, if we query a pattern in combination with the name of an author (e.g. *Umberto Eco wrote*), we want the search results of this query to contain the books by this author.

3.1.2 The Effective Pattern Extraction Algorithm

We present an algorithm to identify effective patterns for relations. For reasons of simplicity we only focus on infix patterns, contrary to the approach by Ravichandran and Hovy. As we are interested in subject – relation – object triplets, we expect the relation to be expressed in text in between the two instances. Therefore, the pre- and postfix parts of the patterns are expected to mainly function as a means to detect the location of instances in the text. We consider this to be a separate concern and return to this topic in the next section.

From the set J of relation instances we select a set $T \subseteq J$ to identify patterns from text and a validation set $V \subseteq I_s$ that is used to check the identified patterns for effectiveness.

The set T should be chosen such the instance-pairs are typical for relation r. We do so by selecting the instance-pairs that are found most frequently in a previous iteration of the ontology population algorithm (Section 2.2).

To identify a (new) set of effective patterns that represent r, we first discover how relation r is expressed in natural language texts on the web. Subsequently we address the problem of selecting *effective* patterns from the total set of patterns found.

Identifying Relation Patterns

We first generate a set of patterns with the use of the following algorithm. For evaluation purposes, we also compute the frequency of each pattern found.

In the first part of the algorithm, the identification phase, we collect a set of patterns by querying both instances i and i' of the pairs in T. We query both " $i \star i'$ " and " $i' \star i$ ". The $\star$ is a regular expression operator, serving as a placeholder for zero or more words. Table 3.2 gives example search results.

Having collected the search results for the given queries, we collect the inner-sentence text fragments in between the two queried instances. The collected text fragments are subsequently normalized by removing all mark-up that is ignored by the search engine. Since popular search engines are case-insensitive and ignore punctuation, we translate all phrases found to a normal form: the simplest expression that we can query that leads to the document retrieved.

For each of the normalized phrases, we compute the frequency p_{freq}. A sketch of the identification phase of the effective pattern extraction algorithm can be found in Table 3.3.

We now have generated a set with patterns and their frequencies within the retrieved snippets.

Leo Tolstoy's masterpiece, War and Peace.
Leo Tolstoy, War and Peace - eSnips, share anything
Leo Tolstoy's major work, War and Peace, is
Leo Tolstoy: His Own War and Peace (Path I) By Ekaterina Chelpanova. Published: 1st June 05
Leo Tolstoy. Then novel War and Peace was written by a famous
Leo Tolstoy?s novel, War and Peace, contains three kinds of material, a historical
Leo Tolstoy/Tolstoi — Download War and Peace
Leo Tolstoy name his book "War and Peace" and not "Peace and War", when
Leo Tolstoy fictionalized him in "War and Peace'
Leo Tolstoy that is not War and Peace? Anna Karenin
Leo Tolstoy to devote a War and Peace to the period of the
Leo Tolstoy's most celebrated novel War and Peace, the vast epic of
Leo Tolstoy, author of "War and Peace" and "Anna Karenina.
Leo Tolstoy's classic work, War and Peace.
Leo Tolstoy's monumental epic War and Peace

Table 3.2. Example search results for the *allintext*-query *Leo Tolstoy* $*$ *War and Peace*.

for each $(i, i') \in T \;\; \rightarrow$
    query the expressions " $i * i'$ " and " $i' * i$ " ;
    extract all phrases $\mathcal{S}$ matching the queried expressions ;
    replace i and i' in $\mathcal{S}$ by placeholders for the classes ;
    normalize $\mathcal{S}$;
    store $\mathcal{S}$ and update its frequency $p_{\text{freq}}(\mathcal{S})$;
rof

Table 3.3. Sketch of the pattern identification phase.

Selecting Relation Patterns

From the list of relation patterns found, we are interested in the most effective ones. Precision is not the only criterion for effectiveness. For example, the retrieved pattern *född 30 mars 1853 i* proved to a 100% precise pattern expressing the relation between a person (*Vincent van Gogh*) and his place of birth (*Zundert*). Clearly, this rare phrase is unsuited to mine instance-pairs of this relation in general. On the other hand, high frequency of some pattern is no guarantee for effectiveness either. The frequently occurring pattern "was born in London" (found when querying for *Thomas Bayes* $\star$ *England*) is well-suited to be used to find London-born persons, but in general the pattern is unsuited – since too narrow – to express the relation between a person and his or her country of origin.

Taking these observations into account, we formulate three criteria for selecting effective relation patterns.

1. The patterns should *frequently* occur on the web, to increase the probability of getting any results when querying the pattern in combination with an instance.
2. The pattern should be *precise*. When we query a pattern in combination with an instance in I_s, we want to have many search results containing instances from c_o.
3. If relation r is not functional, the pattern should be *broad*, i.e. among the search results when querying a combination of the pattern and an instance in I_s there must be as many distinct r-related instances from c_o as possible.

Note that these criteria are language independent. To measure the three criteria, we use the validation set to combine the patterns found with instances i in the validation set V. Hereto we define the following variables.

1. The frequency of a pattern $\mathcal{S}$, $f_{\text{freq}}(\mathcal{S})$, the number of occurrences of $\mathcal{S}$ found in the identification phase.
2. The precision $f_{\text{prec}}(\mathcal{S})$ of a pattern $\mathcal{S}$, is given by

$$f_{\text{prec}}(\mathcal{S}) = \frac{\sum_{i \in V} P(\mathcal{S}, i)}{|V|},$$

for instances $i \in V$, where $P(\mathcal{S}, i)$ is defined as

$$P(\mathcal{S}, i) = \frac{F_I(\mathcal{S}, i)}{F_O(\mathcal{S}, i)}$$

where

$F_I(\mathcal{S},i)$ is the number of snippets
containing instances of $c_o$when querying $\mathcal{S}$ with i

and

$F_O(s,x)$is the total number of snippets found
when querying $\mathcal{S}$ with i.

3. The broadness of a pattern $\mathcal{S}$, $f_{\text{spr}}(\mathcal{S})$ where

$$f_{\text{spr}}(\mathcal{S}) = \sum_{i \in V} B(\mathcal{S},x),$$

with

$$B(s,x) = \begin{array}{l}\text{the number of } \textit{distinct} \text{ instances of class } c_o \text{ found} \\ \text{when querying } \mathcal{S} \text{ with } i.\end{array}$$

The larger we choose the validation set V, the more reliable the measures for precision and broadness.

We finally calculate the score of the patterns. For the non-functional relations, we do so by multiplying the individual scores:

$$\text{score}(\mathcal{S}) = f_{\text{freq}}(\mathcal{S}) \cdot f_{\text{prec}}(\mathcal{S}) \cdot f_{\text{spr}}(\mathcal{S}) \tag{3.1}$$

For the functional relations, the aim is to obtain results with low scores for $f_{\text{spr}}(\mathcal{S})$. For these cases we propose the following function to combine the three parameters.

$$\text{score}(\mathcal{S}) = f_{\text{freq}}(\mathcal{S}) \cdot f_{\text{prec}}(\mathcal{S}) \cdot \frac{1}{f_{\text{spr}}(\mathcal{S})} \tag{3.2}$$

For efficiency reasons, we only compute the scores of the patterns with the highest frequencies. We apply this heuristic, as we are in practice likely to obtain thousands of patterns in the identification phase. For the case of the computation of the scores for patterns expressing functional relations, $f_{\text{freq}}(\mathcal{S})$ is the upperbound for score$(\mathcal{S})$. Hence, if it holds that $f_{\text{freq}}(\mathcal{S}')$ is smaller than score$(\mathcal{S})$ for patterns $\mathcal{S}$ and $\mathcal{S}'$, then score$(\mathcal{S}')$ is also smaller than score$(\mathcal{S})$. Therefore not all patterns need to be evaluated to find the most effective ones. For the case of the non-functional relations, the maximum value for $f_{\text{spr}}(\mathcal{S})$ can be estimated to find an upperbound for score$(\mathcal{S})$.

We express the Google complexity of the effective pattern identification algorithm in terms of the sizes of the identification and validation sets T and V. The

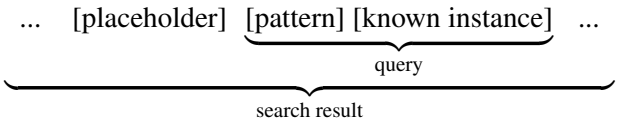

Figure 3.1. Query, pattern, search result and placeholder.

identification phase requires $2|T|$ queries, as each pair in T is queried twice. Each pattern evaluated in the second phase requires $|V|$ queries to determine the score. Hence, assuming that the m most frequently identified patterns are evaluated, the total Google complexity of the algorithm is $2|T| + m|V|$.

3.2 Identifying Instances

As discussed in the first chapters of this thesis, the ontology population task differs from the general information extraction task in a number of aspects. These differences also have their consequences in choosing a strategy in identifying instances from texts.

As we use the web as a corpus, we can assume that the instances are redundantly available. For our task it is not necessary to recognize each of the encountered occurrences [McDowell & Cafarella, 2006]. In recognizing instances, the focus should be on the precision. If we extract erroneous instances and use them in newly constructed queries, this will potentially lead to the extraction of more erroneous instances. Hence, we opt for a strategy with high precision, while the bootstrapping mechanisms should lead to a high recall for the ontology considered.

As we opt for a pattern-based approach, we know the context of the potential instance (i.e. the queried expression) and its placeholder (either preceding or following the search query). We define the maximal distance to the query in terms of the number of words. Figure 3.1 and Tables 3.4 and 3.5 illustrate this task. Within the search results in Table 3.4, the task is to identify professions like *scientist* and *chemist* at a placeholder following the queried expressing. In Table 3.5 a challenge is to recognize *Rachel Carson* as a *Person*, contrary to *Project Manager*.

Now the problem is to identify instances at the placeholder.

The Instance Identification Problem. Given is an initial ontology with relation r on the classes c_s and c_o. Using a query combining an instance in I_s and a pattern expressing r, we obtain query results Q. Given Q, identify instances of class c_o at the placeholder. □

A class in the given initial ontology may be either complete or incomplete. In the

Marie Curie was a world-renowned scientist who made many important discoveries,
Marie Curie was a dedicated humanitarian, eager to
Marie Curie was a two-time Nobel Prize winner and one of the first women ever to
Marie Curie was a lone genius who found new
Marie Curie was a famous scientist as you think. She was born in Poland 1867.
Marie Curie was a brilliant scientist who received two Nobel Prizes.
Marie Curie was a world-renowned scientist who made many important
Marie Curie was a physicist and chemist of Polish upbringing and, subsequently,
Marie Curie was a Polish-born physicist and chemist and one of the most famous scientists of her time.
Marie Curie was a Polish physicist and chemist who lived between 1867-1934.
Marie Curie was a Polish chemist and pioneer in the early field of radiology and
Marie Curie was a Polish-born physicist and chemist and one of the most famous
Marie Curie was a real hero as she

Table 3.4. Example search results for the pattern *[Person] was a [Profession]*, instantiated with *Marie Curie*.

first case, it may seem trivial to recognize instances in texts as we can match the text at the placeholder with the set of instances. However, as the terms representing the instances can be ambiguous (e.g. *Live*, *Madonna*), we present an approach to compensate for the ambiguity. For the task with incomplete classes, we focus on both knowledge-oriented and data-oriented approaches. The design constraint that no representative manually annotated texts are available hampers the latter.

3.2.1 Instance Identification for Complete Classes

Given is an ontology O with a complete class c_a, for example *Painter*. Note that for complete classes all relevant instances are included in the ontology. Hence, if a class is labeled to be complete, all *relevant* instances for the given setting are given.

What were some of Sir Isaac Newton's other jobs before he was a scientist
scientist, while Benjamin Franklin was a scientist
if Einstein was a scientist
Project Manager was a scientist
Poste on 2008-03-13 09:59 by Mike. Fucked Up. I met this new woman. Apparently she was a scientist
Charles W. Buggs (1906-1991) Charles W. Buggs was a scientist
describe an occupation: "My father was a scientist
therefore the proposition that Mahatma Gandhi was a scientist
Kurt Godel was a scientist
gathered to celebrate the centennial of Rachel Carson. She was a scientist
Rachel Carson was a scientist
Niels Bohr was a scientist
Werner Heisenberg was a scientist
Orange Research and Education, and was a scientist
his flaws and excesses (well depicted in the movie), Kinsey was a scientist

Table 3.5. Example search results for the pattern *[Person] was a [Profession]*, instantiated with *scientist*.

We defined the placeholder for instances of the unqueried class in terms of the maximum amount of words between the queried expression and the instance to be identified, given that these words are within the same sentence. We allow distances larger than 0 to compensate for variations in adjective, adverbs and the like. For example, for the pattern [*hyponym*] *is a* [*hypernym*] the words directly preceding the phrase *is a* are typically used to specify the hyponym (Chapter 5.1).

For simplicity, we use the full stop marker to detect sentence boundaries. Recently, Kiss and Strunk [2006] have proposed a more elaborate method to distinguish sentence boundaries from abbreviations in a multilingual corpus using an unsupervised approach.

Having defined the placeholder for instances of class c_a, we scan the search results for occurrences of the instances in the class. For larger sets of instances, a

suffix tree can be built to efficiently represent the instances and match them with the texts found [Gusfield, 1997].

To match the instances with the search results, we can simply opt for an exact string matching approach. As an alternative, we can allow small variations based on the edit distance (to compensate for encountered typo's) or ignore case-differences.

Compensating for Ambiguous Terms. Homonyms are a common phenomenon in natural language, and are one of the factors that complicate natural language processing. Homonyms can have meanings that are quite distant (for example the term *Boston* may refer to the city and the pop band) or more closely related (e.g. *Theo van Gogh* is the name of two different persons, *Groningen* is both the name of a Dutch province and its capital city, *Boston* is both a band and the title of their debut album).

Hence, when encountering an occurrence of one of the instances in a text, we are not guaranteed that the term indeed refers to the intended instance. Ideally, for each occurrence of an instance in a text we want to observe whether the occurrence indeed reflects the intended instance. However, the automatic parsing of texts is troublesome. Moreover if an instance is identified as a subject or object within a sentence, then we still do not know whether the term indeed reflects the instance.

For terms with only one meaning, we can be more confident that occurrences of these terms indeed refer to the intended instance. For term with numerous definitions however, this relation can be much less certain. Hereto, we propose a mechanism to estimate the likeness that a term indeed refers to the instance.

We use the *define* functionality in *Google* to obtain the number of senses of a term. For example, by querying `define: Tool`, we obtained a list of 31 definitions for the term *Tool*, collected from various online dictionaries and encyclopedias. This indicates that Tool is an ambiguous term. On the contrary, terms such as *Daft Punk*, *Fatboy Slim* and *Johannes Brahms* lead to precisely one definition. We define $n(i)$ to be the number of definitions for i that are returned by Google. If no definitions are returned, we consider $n(i)$ to be 1.

In Tables 3.6 and 3.7 examples are given for definitions for an ambiguous and an unambiguous instance of the class *Artist*.

Based on the number of definitions $n(i)$ returned by Google, we formulate an estimate $p(i)$ for the likeliness that i_a is the intended meaning of the term. We investigate the following two alternatives to estimate $p(i)$.

- **uniform.** As we do not know anything about the distributions of the use of the definitions for term i, we estimate that each definition has a equal probability to be used and that only one of the definitions reflects the instance.

Definitions of Boston on the Web:

• state capital and largest city of Massachusetts; a major center for banking and financial services
wordnet.princeton.edu/perl/webwn

• Boston is an American rock band that achieved its most notable successes during the 1970s and 1980s. Centered on guitarist, songwriter, and producer Tom Scholz, the band is a staple of classic rock radio playlists. ...
en.wikipedia.org/wiki/Boston(band)

• "Boston" is a song by Augustana.
en.wikipedia.org/wiki/Boston(song)

• Boston is the self-titled debut album by American rock band Boston. The album broke fast, with several blockbuster hard rock hits. All eight of the songs on the album still receive regular airplay on classic rock radio. ...
en.wikipedia.org/wiki/Boston(album)

• Boston (1833-1850), a chestnut with a white nose (and often called "Damn his eyes" because no one could beat him), was born in Richmond, Virginia. ...
en.wikipedia.org/wiki/Boston(horse)

• Boston is a local government district with borough status in Lincolnshire, England. Its council is based in the town of Boston. It lies around N530'0" W00'0".
en.wikipedia.org/wiki/Boston(borough)

Table 3.6. Top results (of 12 in total) for the *Google* query 'define: Boston' as retrieved on March 12, 2008.

Definitions of Right Said Fred on the Web:
• Right Said Fred is the name of a British pop band, which was founded in 1989 by brothers Richard Fairbrass and Fred Fairbrass from East Grinstead. ... `en.wikipedia.org/wiki/RightSaidFred`

Table 3.7. All results for the *Google* query 'define: Right Said Fred' as retrieved on March 12, 2008.

$$p_{\text{lin}}(i) = \frac{1}{n(i)} \tag{3.3}$$

- **square root.** Especially for terms with many definitions, we observe some overlap between the definitions. Moreover, two distinct definitions can be closely related. For example, *Red Hot Chili Peppers* is the name of a band and the name of their debut album. We therefore investigate a second method to estimate $p(a)$ by using the square root of the number of definitions found.

$$p_{\text{sqrt}}(i) = \frac{1}{\sqrt{n(i)}} \tag{3.4}$$

We determine a confidence score p_i for the class membership of an instance i of c_a, we multiply the estimates $p(i)$ with the number of occurrences $\text{oc}(i)$ of the instance in the search results.

$$p_i = \frac{\text{oc}(i) \cdot p(i)}{\sum_{i' \in I_a} \text{oc}(i') \cdot p(i')} \tag{3.5}$$

We use these estimates to obtain a sorted list of instances for the class c_a. In Chapter 6 we use these estimators to score relations between ambiguous terms.

3.2.2 Instance Identification for Incomplete Classes

To recognize instances in incomplete classes, the strategies described for the complete class case may be applied to recognize already known instances. In this part of the chapter, we focus on the recognition of instances that are not already included in the ontology. Here, we focus on both knowledge- and data-driven approaches.

Knowledge-Driven Approach

A commonly used strategy to recognize instances in a text, is to formulate recognition rules [Chinchor, 1995; Etzioni et al., 2005; Sumida et al., 2006; Schedl &

Class	*Regular Expression*	*examples*
Year	$(1\|2)\cdot(0-9)^3$	1992, 2345
Gender	he|she|son|..	male, female
Person	((A-Z)·(a-z)$^+$)^2	Johnny Cash, George Baker
Person	(A-Z)·(a-z)$^+$ (A-Z). (A-Z)·(a-z)$^+$	George W. Bush, Anton F. Philips

Table 3.8. Classes and possible recognition rules

Widmer, 2007]. It is notable that such a knowledge-driven approach to recognize instances is class-dependent. For example, recognizing instances of *Movie* is done differently from recognizing instances of the class *Year*.

When designing rules to recognize instances at the placeholders in the search results, we focus on the structure of the instances and their context [De Meulder & Daelemans, 2003].

- **Context.** The left and right context for a term can be expressed as regular expressions. For example, a term in an enumeration may have a comma as its left context and the word *and* as its right context.
- **Structure.** Rules describing the structure focus on the number of words and the use of capitals and punctuation marks. For example, a person's name can be recognized as two or three capitalized words.

  The rules describing the structure of instances can be described using a regular expression.

We formulate regular expressions and a maximum distance from the queried expression to identify instances from texts. Table 3.8 gives example regular expressions to recognize the structure of instances. Instances of the class *Year* is for example specified as a four digit term preceded by the name of a month. For instances of the class *Gender*, the instances are indirectly recognized. The text is for example scanned for the word *son*, which corresponds to the instance *male*.

The algorithm to identify instances using such a rule-based approach is sketched in Table 3.9. We first scan the text for an occurrence of the instance (described by M) encapsulated by a left context (c_l) and a right context (c_r). The $*$ is a wildcard symbol matching any string. If we encounter a substring that is described by $c_l \cdot M \cdot c_r$, we isolate the string of maximum length matching M.

Apart from the structure and context, in general information extraction tasks there is a third method that is used to identify instances in texts. Using a part-of-speech tagger [Brill, 1992] the roles (e.g. subject) in the sentence and word groups (e.g. noun phrases) can be identified. These techniques are useful to identify terms in text [Frantzi, Ananiado, & Mima, 2000; Etzioni et al., 2005]. An alternative

Q is the set of sentences containing the queried expression ;
"select the parts of the sentences matching the placeholder" ;
for all fragments $q \in Q \rightarrow$
  re $= c_l \cdot \mathrm{M} \cdot c_r \cdot *$;
  $b =$ match(re,q) ;
  do $(\neg b \wedge \mathrm{length}(q) > 0) \rightarrow$
    "remove *first* word or punctuation mark from q" ;
    $b =$ match(re,q) ;
  od
  if $(b) \rightarrow$
    re $= c_l \cdot \mathrm{M}$;
    $b =$ match(re,q) ;
    do $(\neg b) \rightarrow$
      "remove *last* word or punctuation mark from q" ;
      $b =$ match(re,q) ;
    od
    re $= \mathrm{M}$;
    $b =$ match(re,q) ;
    do $(\neg b) \rightarrow$
      "remove *first* word or punctuation mark from q" ;
      $b =$ match(re,q) ;
    od
  fi
rof

Table 3.9. Identifying instances using rules.

approach is the use of N-gram statistics to identify named entities [Downey et al., 2007].

Acceptance functions. After extracting a term, we can perform an additional check to find out whether the extracted term is really an instance of the concerning class. We perform this check with the use of a search engine. We query phrases that express the term-class relation. Again, these phrases can be constructed semi-automatically. Hyponym patterns are candidates as well for this purpose [Hearst, 1992, 1998; Cimiano & Staab, 2004]. A term is to be accepted as instance, when the number of hits of the queried phrase is at least a certain threshold. For example, we query the phrase *'Cities such as Eindhoven and'* to check whether *'Eindhoven'*

is indeed an instance of the class *City*.

Using a set of patterns R expressing relation r, we formulate the following acceptance function:

$$\text{accept}_{c_s}(t) = \begin{cases} \text{true} & \text{if } \sum_{\mathcal{S}\in R} h(\mathcal{S}, c_s, t) \geq n \\ \text{false} & \text{otherwise} \end{cases}$$

where $h(\mathcal{S}, c_s, t)$ is the number of hits for query with pattern $\mathcal{S}$ combined with term t and the plural form of the name of class c_s. The threshold n has to be chosen beforehand. We can do so, by calculating the sum of *hits* for queries with known instances of the class. Based on these figures, a threshold can be chosen e.g. the minimum of these sums. When the instances in the initial ontology are well-known, the sum of hits for these instances can be expected to be large. Hence, setting a threshold based on such instances will lead to a threshold (and acceptance function) that will filter out correct, but less well-known, instances.

When we use such an acceptance function, we can allow ourselves to formulate less strict recognition rules. That is, false instances that are at first accepted, are still rejected as an instance by the use of the acceptance function.

As an alternative, a term t can be checked using *Google*'s define functionality. If the name of c_s occurs in one of the definitions for t, then t is likely to be an instance of c_s. In Chapter 4 we will use this mechanism to evaluate a populated ontology.

Data-Driven Approach

In Chapter 2 we argued that we opt for an approach without manual annotations. Hence, in choosing a data-driven approach, we should opt for an unsupervised learning mechanism to recognize instances in texts.

Using a set of instances, we head for an approach where we create a training set of texts by automatic annotations. We illustrate the construction of a training set with the following example. Suppose we have an ontology with two classes, *Year* and *Person*. We assume both sets of instances to be non-empty. The relation *yob* between the two classes is expressed by the given pattern *[person] was born in [year]*.

Now, we select one instance and pattern combination, say *was born in 1854* (cf. Table 3.10). We can automatically annotate the search results for this query as *1854* is a known instance of the class *Year*. In the search results, the queried instances can be automatically labeled marked as members of the class. Other instances (e.g. 1875 in the first line of Table 3.10) are ignored as these instances are less likely to reflect a year of birth.

When we select an instance from the other class, i.e. *Person*, we are to scan the search results for instances of *Year*. Using the search results for the other query, we

Maria Paulina "Mary" Wittrock *was born in* **1854** and, in the year 1875, was the
At one point, she claimed she *was born in* **1854**.
Secretary Weir *was born in* **1854** in El Monte, California, and spent his childhood
(11.) Lucinda Crank *was born in* **1854**. 4. Mary Polly Crank was born in 1804 and died in 1883.
resident of Cherokee township, *was born in* **1854** in Pennsylvania, where he lived
Wilde, 1854-1900. Oscar Wilde *was born in* **1854** and grew up in an intellectually bustling Irish
of Wisconsin. Home Page. William Alexander Grimshaw *was born in* **1854** in New York.
4 iii. Thomas COLLINS *was born in* **1854** in Michigan. 5 iv.
Edith A. Curry *was born in* **1854** in Kentucky. She died on 15 Oct 1930 in Georgia. M
Elizabeth Youngblood *was born in* **1854**-5. She married Post.
Oscar Wilde *was born in* **1854** and grew up in an intellectually bustling Irish household. His mother was a poet who wrote under the pen name Speranza and who had a

Table 3.10. Example search results for the query *was born in 1854*. The instance *1854* is annotated.

can learn the structure and context of instances of this class. Hence, when querying for example *Alan Turing was born in*, the use of the search results (Table 3.11) is twofold.

1. The search results are used to identify instances of the class *Year* and relation instances expressing the relation between Turing and his year of birth.
2. The training set to learn instances of *Person* is expanded using the search results for the said query.

In the training set, we only annotate the queried instance. Having labeled the instances in the sentences in the training set, the task is to create training data for a classifier. Hereto, each word, number and punctuation mark, called *token*, is

in the town of Chatrapur, *Alan Turing was born in* a nursing home in Paddington, London.
Alan Turing was born in London, England, on June 23, 1912.
Alan Turing was born in 1912 and showed an early interest in the natural world. He studied mathematics at Cambridge University and established himself as a
Alan Turing was born in London, England, on June 23, 1912. Both his parents
Alan Turing was born in 1914 and
Alan Turing was born in London, England in 1912.
Alan Turing was born in Paddington London on 23 June 1912 and went on to study
Alan Turing was born in London on June 23, 1912.
Ir J Psych Med March 2003;Vol 20 No 1: 28-31. *Alan Turing was born in* Paddington, London on June 23, 1912. His family were middle-class and well-off.
Alan Turing was born in London, England, on June 23, 1912.
Known as the founder of Computer Science, *Alan Turing was born in* 1912 in Paddington, London.
Alan Turing was born in June of 1912 to Julius Mathison Turing, a member of the

Table 3.11. Example search results for the query *Alan Turing was born in*. The goal is to identify the year of birth.

labeled in the sentences. We distinguish three classes:

- *start*, signaling the start of an instance,
- *intern*, indicating all tokens within an instance following the begin label, and
- *not*, indicating all text that is not included in the instance.

Note that with these three labels are enough to distinguish separate instances in a text. An explicit *end* label is not required, as the last word or punctuation mark can be derived implicitly. The distinction between *begin* and *intern* is needed to separate and recognize two subsequent instances in one sentence.

Now, given an annotated sentence, we describe each of the tokens using a feature vector. Each vector describes a *focus word* in the sentence, i.e. the token to be labeled and its context. In each vector, the label is associated with the focus word. Table 3.12 gives a collection of vectors representing one sentence, i.e. one vector per focus word. We use a window of fixed size n (in the given example $n = 1$) to represent the left and right context of the focus word.

LEFT CONTEXT	RIGHT CONTEXT	FOCUS WORD	LABEL
	is	Afghanistan	*start*
Afghanistan	a	is	*not*
is	conservative	a	*not*
a	Islamic	conservative	*not*
conservative	country	Islamic	*not*
Islamic	and	country	*not*
country	99	and	*not*
and	per	99	*not*
99	cent	per	*not*

Table 3.12. Feature vectors for *... Afghanistan is a conservative Islamitic country and 90 per cent of its population is Muslim ...* representing the focus word and the context window of one token.

As we choose to opt for an approach solely based on the syntax, the features that can be extracted to describe the focus word are limited. We concentrate on the presence of capitals, as their use is common in many named entities. We distinguish the features *numeric* (to abstract from numbers), *no word* (to abstract from other tokens without letters), *capitalized* (for tokens starting with a capital) and *no caps*.

For each focus word, we create a vector of length $4n+3$. For a window size of n, we consider the n tokens preceding and following the focus word. Each of these tokens is represented by 2 features: the token itself and its abstraction. Hence, the context of the focus word is described by $4n$ features. The focus word itself, its abstraction and its class are the other three features in the vector.

Having constructed a set of training vectors, we translate query results into a set of test vectors in a similar fashion. The task is to classify the vectors into the classes *not*, *start* and *intern*. The goal is to recognize instances at the placeholder, by observing similarities in structure and context with respect to vectors in the training set.

We choose to use Memory-based learning (MBL) to classify the vectors [Daelemans & Bosch, 2005][1]. Contrary to other popular machine learning approaches, memory-based learning does not abstract from the data processed. This characteristic has proven to be successful in natural language processing, as irregularities and various exceptions are typical for natural language. MBL has also shown to be well usable for cases with a limited context. As we are interested in information extraction from snippets, this property is of high importance.

[1] In our experiments, we use TiMBL (version6.1, `http://ilk.uvt.nl/timbl/`) with the standard parameter settings.

MBL is based on k-nearest neighbors classification. Suppose we have a set T of vectors in the training set. A distance measure $\Delta(v,t)$ between two vectors is used. This measure is a weighted sum over the distances of the features. The weight is used to express the importance of a feature. Typically, features representing tokens with a large distance to the focus word are less important than features representing closer ones.

$$\Delta(v,t) = \sum_i w_i \cdot \delta(v_i,t_i)$$

As the features are non-numerical, $\delta(v_i,t_i)$ has been simply defined as follows.

$$\delta(v_i,t_i) = \begin{cases} 1 & \text{if the strings } v_i \text{ and } t_i \text{ are equal} \\ 0 & \text{otherwise} \end{cases}$$

The training set is used to compute weights for each of the features, based on the information gain of the features with respect to the class labels [Duda, Hart, & Stork, 2000]. For a given k, the label is selected using majority voting among the k closest vectors in the training set.

In cases with a small training set, it is likely that a high weight is assigned to the focus word itself. In such cases the classifier is overfitted. This may lead to a situation where only known instances are recognized. To avoid this situation, we can alternatively leave out the focus word itself from the feature vector. In that case, the classification is solely based on the context and the abstraction of the focus word. We will return to this topic in the next chapter.

Having classified the individual vectors, we have to extract the instances from the data. From each vector classified as *start*, the focus term is identified. If the focus term is obscured, we look it up from the original search results. For all following vectors classified as *intern*, we extract the focus terms as well. The extracted focus terms are combined into one term and added to the ontology as an instance. Note that we ignore vectors classified as *intern* that are not preceded by a vector labeled *start*.

Returning to the example of *Alan Turing*'s year of birth, to recognize instances of the class *Year* the choice for a data-oriented approach is less obvious. The instances can easily be described using a rule, and a complete set of instances can be generated. However, constructing rules to recognize instances of other classes (e.g. *Pop Artist*, *City*, *Movie*) can be less straightforward.

In the next chapter, we present case-studies where we compare methods using rule-based instance identification approaches with instance identification using memory-based learning.

4

Evaluation: Extracting Factual Information From the Web

In the previous chapters, we discussed methods to populate an ontology using texts found with a search engine, in this chapter we present case-studies to illustrate the applicability of these methods. We compare the various alternatives in instance identification. For evaluation purposes, we choose to populate ontologies on domains that are verifiable. For each of the populated ontologies, the precision can be determined and recall can be analyzed. In all case-studies we solely make use of the document titles and snippets returned by the search engine. The documents themselves are thus not accessed.

Section 4.1 focusses on the population of an ontology using manually identified patterns, where the instances are recognized using rules.

In Section 4.2, we focus on the identification of effective patterns. Using a small training set, we are interested whether the patterns found can be used in an information extraction task. In Section 4.3 we identify a list of effective hyponym patterns and compare this list with commonly used ones in the literature. The learned patterns are used in an experiment where we investigate the applicability of memory-based learning (MBL) in our web information extraction setting.

Section 4.4 focusses on the identification of instances and the effect of the bootstrapping mechanisms. We compare a rule-based approach with an approach

using MBL in finding all presidents of the US and their order.

Finally, in Section 4.5 we focus on an extensive case-study: the identification of a list of historical persons with their biographies.

4.1 Populating a Movie Ontology

For our first case study, we have constructed a small initial ontology on the movie domain. It is defined as $O = (C, R)$ where

$$
\begin{array}{lcl}
C & = & \{c_{\text{Director}}, c_{\text{Actor}}, c_{\text{Movie}}\}, \\
R & = & \{r_{\text{acts-in}}, r_{\text{directed}}\}, \\
I_{\text{Director}} & = & \{\ \textit{Steven Spielberg}\ ,\textit{Francis Ford Coppola}\} \\
I_{\text{Actor}} & = & \emptyset, \\
I_{\text{Movie}} & = & \emptyset, \\
r_{\text{acts-in}} & = & (\text{acts in}, c_{\text{Actor}}, c_{\text{Movie}}, \varphi, \emptyset) \\
r_{\text{directed}} & = & (\text{directed}, c_{\text{Director}}, c_{\text{Movie}}, \varphi, \emptyset)
\end{array}
$$

We thus only identify three classes, each of them are incomplete. The class *Director* is the only class where instances are provided. For the two relations, no relation instances are given. The goal is to identify movies directed by these directors using patterns expressing the *directed* relation. The movies found are used to find starring actors, where those actors are the basis of the search for other movies in which they played, etc. In this experiment, we focus on the population of O in one iteration, thus using a predefined set of patterns. We extract the information from the top 100 snippets returned by *Google*.

For the two relations considered, we have manually selected the following patterns and placeholders.

$$
\begin{array}{lcl}
P_{\text{acts-in}} & = & \{[\textit{Movie}]\ \textit{starring}\ [\textit{Actor}]\} \\
P_{\text{directed}} & = & \{[\textit{Director}]\textit{'s}\ [\textit{Movie}]\ ,\ [\textit{Movie}],\ \textit{director}\ [\textit{Director}]\ \}.
\end{array}
$$

Instance identification. For all three classes, as a placeholder, we use the remaining part of the sentence preceding or following the queried expression. We do so, as multiple instances of the same class are often enumerated (e.g. in the sentence *Titanic starring Leonardo Di Caprio and Kate Winslet.*). As actors and directors are generally both persons, we apply the same recognition rules for these two classes.

As the structure of instances of *Movie* is less regular, we focus on the context of such instances in the texts. We recognize an instance of *Movie* if it is placed between quotation marks. A person's name (instances of the classes *Director* and *Actor*) is recognized as the longest sequence of two or three capitalized words.

Another feature of the recognition function is the use of tabu words[1]. An ex-

[1] Also called *stop words* in literature.

tracted term is rejected as instance, if it contains one of the tabu words. We use a list of about 90 tabu words for the person names (containing words like 'DVD' and 'Biography'). For the movie titles we use a much shorter list, since movie titles can be much more diverse. We have constructed the tabu word lists based on the output of a first run of the algorithm.

We *check* each of the extracted candidate instances with the use of one of the following queries: "The movie [*Movie*]", "[*Actor*] plays", or "[*Director*] directed". A candidate is accepted, if the number of *hits* to the query exceeds a threshold. After some tests we choose 5 as a threshold value, since this threshold filtered out not only false instances but most of the common spelling errors in true instances as well.

Experimental results. We have found 7,000 instances of the class Actor, 3,300 of Director and 12,000 of Movie. The total number of retrieved instances increases with about 7% when 500 query results are used instead of 100.

We first ran the algorithm with the names of two (well-known) directors as input: *Francis Ford Coppola* and *Steven Spielberg*. Afterwards, we experimented with other less famous directors as input.

An interesting observation is that the outputs are independent of the input sets. That is, when we take a subset of the output of an experiment as the input of another experiment, the outputs are the same, modulo some small differences due to the changes in the Google query results over time.

When we analyze the *precision* of the results, we use the data from the Internet Movie Database (IMDb)[2] as a reference. An instance in the populated ontology is accepted as a correct one, if it can be found in IMDb. We have manually checked three sequences of 100 instances (at the beginning, middle and end of the generated file) of each class. Based on the exact matches with the entries in IMBb, we estimate a precision of 0.78. Most misclassified instances were misspellings or different formulations of the same entity (e.g. "Leo DiCaprio" and "Leonardo DiCaprio"). Other identified instances, like *James Bond* (found as an instance of both *Movie* and *Actor*) and *Mickey Mouse* (found as an actor in *Fantasia 2000*) were related to the movie domain, but are no instances of the intended classes. It is also notable that not all instances found relate to distinct concepts. For example, *the Karate Kid* as well as *Karate Kid* were identified as movies, likewise *Leo DiCaprio* and *Leonardo DiCaprio* were added to the class *Actor*, while *Hable con Ella* and *Talk to Her* are alternative titles for the same movie.

Likewise, we have also analyzed the precision of the relations, we estimate the precision of the relation between movie and director around 0.85, and between

[2] `http://www.imdb.com`

CATEGORY	RECALL
Best Actor	0.96
Best Actress	0.94
Best Director	0.98
Best Picture	0.87

Table 4.1. Recall of Academy Award Winners

movie and actor around 0.90.

With respect to the *recall* of the algorithm, we first observe that number of entries in IMDb exceeds our ontology by far. Although our algorithm performs especially well on recent productions, we also are interested how well it performs on classic movies, actors and directors. First, we made lists of all Academy Award winners (1927-2005) in a number of relevant categories, and checked the recall (Table 4.1).

IMDb has a top 250 of best movies ever, of which 85% were retrieved. We observe that results are strongly oriented towards Hollywood productions. We also made a list of all winners of the Cannes Film Festival, the 'Palme d'Or'. Alas, our algorithm only extracted 26 of the 58 winning movies in this category.

Sumida et al. [2006] used a large Japanese web corpus to identify a list of movie titles. The texts are scanned for hyponym patterns [Hearst, 1992], with phrases like *Movies such as*. Movie titles are extracted when by signaling text between the Japanese variant of quotation marks. They report a precision of 83%. KnowItAll [Etzioni et al., 2005] uses hyponym patterns to find actors and movie titles as well as the patterns *[Actor] stars in [Movie]* and *[Actor] star of [Movie]*. Noun phrases are extracted as candidate instances. These candidate instances are subsequently checked using additional queries. The focus of the evaluation is on the population of the classes, rather than on the identification of relation instances. Precision and recall are formulated in terms of the texts processed, rather than using ground truth ontology. For instances of the class *Actor* high precision is obtained for various levels of recall. The precision of instances *Movie* found with KnowItAll is less precise.

4.2 Identifying Burger King and its Empire

Inspired by one of the list questions in the 2004 Text Retrieval Conference (TREC) Question Answering track [Voorhees, 2004], (*'What countries is Burger King located in?'*), we are interested in populating an ontology with restaurants and the

PATTERN	PREC	SPR	FREQ
c_o restaurants of c_s	0.24	15	21
c_o restaurants in c_s	0.07	19	9
c_o hamburger chain that occupies villages throughout modern day c_s	1.0	1	7
c_o restaurant in c_s	0.06	16	6
c_o restaurants in the c_s	0.13	16	2
c_o hamburger restaurant in southern c_s	1.0	1	4
c_o en co in de verenigde staten c_s	1.0	1	3
c_s concurrençant kfc et c_o	1.0	1	3
c_o boycott over us c_s	0.92	1	3
c_o hamburger even in c_s	0.66	1	3
c_s we have mcdonald's burger king pizza hut and c_o	1.0	2	1
c_o hamburger spokesman throughout c_s	1.0	1	2
c_o new mcveggie burger in c_s	1.0	1	2
c_o has 1320 restaurants in c_s	1.0	1	2

Table 4.2. Top learned patterns for the restaurant (c_o) -country (c_s) relation.

countries in which they operate. We identify the classes c_s for *country* and c_o *restaurant* and the relation *located in* between the two classes.

In this case-study we aim for the population of an ontology with names of restaurant chains and their locations. For the given task, we define the ontology O as follows: $O = (\{c_s, c_o\}, \{r\})$. Here, c_s is the *complete* class with all countries in the world[3]. The relation r expresses the non-functional *is located in* relation. We have added no patterns to r, $r = (\text{is located in}, c_s, c_o, \text{false}, J)$, but instead included a small set of relation instances. The goal is therefore to first identify a set of effective patterns, that can subsequently be used to populate the ontology.

We assign the instances *McDonald's* and *KFC* to the *incomplete* class c_o, as well as a handful of relation instances: $J = \{$*(China,McDonald's)*, *(United States, McDonald's)*, *(Canada, McDonald's)*, *(France, McDonald's)*, *(Australia, McDonald's)*, *(Netherlands,McDonald's)*, *(Germany, KFC)*, *(Netherlands,KFC)* $\}$.

Identifying Patterns. The patterns are identified using the eight relation instances provided in the initial ontology. Using this small set, we identified a list of 170 patterns. These patterns are validated and ranked using the two instances of the restaurant class. The total number of queries used is thus very limited: 8 for the identification phase and 2 for the validation phase.

[3]The *conventional short forms* taken from the CIA World Factbook `http://www.cia.gov/cia/publications/factbook`

"restaurants including t and"
"restaurants for example t and"
"restaurants like t and"
"restaurants such as t and"

Table 4.3. Hyponym patterns for instance-class relation.

The learned patterns with the highest scores are given in Table 4.2. We note that the numbers for *spr* and *freq* are low due to the limited number of queries used. Many of the patterns are recognizable as typical for the given relation. However, the vast majority of the patterns has a value for $p_{\text{spr}} = 1$. Hence, the use of such a pattern in combination with either *KFC* or *Burger King* led only to a single country name. These patterns may therefore be too specific.

Recognizing Instances. The country names are recognized using the collected list of countries. We assume the country names to be unambiguous. We extract the longest terms t consisting of at most 4 capitalized words at the placeholder, directly preceding or following the queried expression.

As this extraction rule is likely to not only cover restaurants but a wide range of terms, we use a check function to filter out erroneously extracted terms. Hereto, we use the set H of hyponym patterns in table 4.3 in the following acceptance function

$$\text{accept}(t) = \sum_{p \in H} h(p,t) \geq n,$$

where $h(p,t)$ is the number of search engine *hits* for query with pattern p combined with term t. Based on experiments with the two known restaurant chains, we set the threshold to $n = 50$.

Evaluation We selected the 20 best scoring patterns and use them in the ontology population algorithm with the given ontology. The first task is to identify *Burger King* as a restaurant using the given patterns. If this instance is found, it can be used – in combination with the same patterns – to identify the countries it is located in.

Using the 20 learned patterns, 53 terms were accepted as instances of restaurant (Table 4.4)[4]. The reader may recognize a number of restaurant, coffee and fast food chains, among which *Burger King*. Less expected are the names of geographic locations and names of famous cuisines such as 'Chinese' and 'French'. The last category of false instances found that have not been filtered out, are a number of very common words (e.g. 'It' and 'There').

[4]Experiment conducted in November 2005 using Google.

Chinese	Bank	Outback Steakhouse
Denny's	Pizza Hut	Kentucky Fried Chicken
Subway	Taco Bell	Continental
Holywood	Wendy's	Long John Silver's
HOTEL OR	This	**Burger King**
Japanese	West	Keg Steakhouse
You	BP	Outback
World	Brazil	San Francisco
Leo	Victoria	New York
These	Lyons	Starbucks
FELIX	Roy	California Pizza Kitchen
Marks	Cities	Emperor
Friendly	Harvest	Friday
Tim Hortons	Vienna	Montana
Louis XV	Greens	Red Lobster
Good	It	There
That	Mark	Dunkin Donuts
Italia	French	

Table 4.4. Extracted instances for restaurant using Google, December 2005.

In Table 4.5 the 53 extracted terms can be found, together with the acceptance scores $\sum_{p \in H} h(p,t)$ as found with the *Yahoo!* API in June 2008. As *OR* is an special query operator, the hits for *HOTEL OR* will not reflect the actual number of occurrences for the corresponding phrases. The high scoring terms – except for *HOTEL OR* – correspond to large chains. Where *Keg Steakhouse* was accepted as an instance in 2005, no hits were found for any of the four acceptance function queries.

The algorithm returned 69 instance-pairs with countries related to *Burger King*. On the Burger King website[5] a list of the 65 countries can be found in which the hamburger chain operates. Of these 65 countries, we identified 55. This implies that our results have a precision of $\frac{55}{69} = 0.80$ and recall of $\frac{55}{65} = 0.85$. Many of the falsely related countries – mostly in Eastern Europe – are locations where Burger King is said to have plans to expand its 'empire'.

Using post-processing, we can filter out common words (e.g. *Good*, *It*) which is likely to improve the results. The geographic locations can be recognized and filtered out using a gazetteer [Cunningham, Maynard, Bontcheva, & Tablan, 2002; Zong, Wu, Sun, Lim, & Goh, 2005].

We hence can conclude that learning patterns using only a small set of known

[5] `http://www.whopper.com`

Outback Steakhouse	1748	Friday	25
HOTEL OR	821	Harvest	24
Burger King	506	Louis XV	23
Starbucks	489	Marks	22
Red Lobster	488	Tim Hortons	22
Pizza Hut	463	You	21
Taco Bell	392	West	18
Subway	375	New York	17
Denny's	332	Bank	12
Long John Silver's	302	Vienna	12
Chinese	288	Montana	12
Wendy's	240	Good	11
This	166	Lyons	10
Japanese	149	Continental	7
French	144	Mark	5
Outback	135	Italia	5
California Pizza Kitchen	111	Emperor	4
Dunkin Donuts	104	Cities	3
That	103	San Francisco	2
Roy	90	BP	1
These	48	Leo	1
Victoria	47	Holywood	0
Kentucky Fried Chicken	46	Keg Steakhouse	0
Greens	30	World	0
Friendly	29	Brazil	0
It	28	There	0
FELIX	25	*Hotel Or*	0

Table 4.5. The 53 instances found for restaurant and their scores for the acceptance function as found with the *Yahoo!* API in June 2008.

instances leads to good results in this case-study. The learned patterns are quite specific, but recognizable as strings relating the instances of the two classes. The extraction of the instances of the class *Restaurant* is done using simple rules. A number of irrelevant terms are falsely identified, but additional filtering steps may lead to improvements.

4.3 Identifying Countries

In this case-study, we focus on two tasks. First, we automatically identify hyponym patterns and compare the results with the patterns identified by Hearst [1992]. Contrary to the previous case-study, we will use a relatively large training set of relation

instances to identify a set of effective patterns. The learned patterns are used in the second part of this experiment to extract instances using memory-based learning.

4.3.1 Learning Effective Hyponym Patterns

We are interested whether the effective text patterns are indeed intuitive formulations of the given relation. As a test-case, we compute the most effective patterns for the hyponym relation using a test set with names of all countries. Taking the terms *country* and *countries* as hypernyms, we are interested which text fragments connect the names of countries with these words. Much pattern-based information extraction research (e.g. [Caraballo, 1999; Cimiano & Staab, 2004; Etzioni et al., 2005; Snow, Jurafsky, & Ng, 2006; Tjong Kim Sang & Hofmann, 2007]) is based on hyponym patterns manually identified by Hearst in [1992]. We are interested in the overlap of the automatically found hyponym patterns with the commonly used ones.

This experiment was set up as follows. We again use the collected list of countries (see Section 4.2). Let I_o be this set of countries, and let I_s be the set $\{$ (*countries*, *country* $\}$. The set of relation instances J consists of all instance combinations (*countries*, a) and (*country*, a) , for $a \in I_o$. We apply the text pattern learning algorithm as discussed in Section 3.1 on this set of relation instances.

Using the proposed pattern learning algorithm, we identified almost 40,000 patterns. We computed f_{spr} and f_{prec} for the 1,000 most frequently found patterns. In Table 4.6, we give the 25 most effective patterns found by the algorithm.

The common hyponym patterns 'like' and 'such as' show to be the most effective. This observation is useful, when we want to minimize the amount of queries for hyponym patterns. Other commonly hyponym patterns with high scores are *including*, *and other* and *namely*. All infix patterns identified by Hearst ([1992], Table 2.1 on page 28) are identified here as well.

Apart from hyponym patterns that can be generally used, we also find patterns that are specific for the given setting. Patterns like *code for*, *flag of* are very usable to identify the studied relation. Such phrases are directly recognizable as usable patterns, but may not be straightforward to identify manually. Other patterns containing an adjective (e.g. *is a sovereign*) are perhaps over-specific, but well-usable. The combinations of *is a*, *is an* or *is the* with an adjective occur in total 2,400 times in the list.

We conclude that the commonly used hyponym patterns are indeed also identified as effective patterns. Moreover, some patterns that are very typical for this setting (i.e. all countries and their hypernyms) are identified as well. Although these patterns are intuitive formulations of the studied relation, they are less straightforward to find manually.

PATTERN	FREQ	PREC	SPR
(countries) like	645	0.66	134
(countries) such as	537	0.54	126
is a small (country)	142	0.69	110
(country) code for	342	0.36	84
(country) map of	345	0.34	78
(countries) including	430	0.21	93
is the only (country)	138	0.55	102
is a (country)	339	0.22	99
(country) flag of	251	0.63	46
and other (countries)	279	0.34	72
and neighboring (countries)	164	0.43	92
(country) name republic of	83	0.93	76
(country) book of	59	0.77	118
is a poor (country)	63	0.73	106
is the first (country)	53	0.70	112
(countries) except	146	0.37	76
(country) code for calling	157	0.95	26
is an independent (country)	62	0.55	114
and surrounding (countries)	84	0.40	107
is one of the poorest (countries)	61	0.75	78
and several other (countries)	65	0.59	90
among other (countries)	84	0.38	97
is a sovereign (country)	48	0.69	89
or any other (countries)	87	0.58	58
(countries) namely	58	0.44	109

Table 4.6. Learned hyponym patterns and their scores.

4.3.2 Recognizing Instances using Memory-Based Learning

Contrary to the case-study in Section 4.2, we now focus on a data-driven approach to identify instances. We apply the method discussed in Section 3.1.2 to extract instances by classifying feature vectors derived from the search results.

As the focus is on the instance identification task, we use the best scoring hyponym patterns found with the complete list of countries. We selected the 11 most effective patterns to express the relation (cf. Table 4.6). Contrary to the pattern learning set-up, the class c_o is incomplete. We now only assume given the country names starting with A – D, i.e. from *Afghanistan* to *the Dominican Republic*. These names of countries are used to annotate search results and train the classifier as described in Chapter 2. No relation instances are provided. The complete class with instances *country* and *countries* remains unchanged.

I am deeply grateful to the Secretary General for the opportunity of working ... and destitute state - such as **Afghanistan** – provides fertile ground for ...
In an uninhabited region such as **Antarctica**, ...
with vaunted ski industries such as **Austria** and Switzerland insist
they aren't ...

Table 4.7. Example search results used to train the classifier.

We run the ontology population algorithm using the described initial ontology. Using the known names of countries, we construct a training set with all known instance-pattern combinations, i.e. all the snippets found with *like* (from *like Afghanistan* up to *like the Dominican Republic*) and the ten other patterns.

In Table 4.7 some example sentences are shown that are used in a training set of the classifier. Note that *Switzerland* will not be annotated as a country name, as the term is no instance in the initial ontology.

The annotated search results are used to identify instances of c_{country} when querying instances of the other class. We recognize instances in the search results for the 22 (11 patterns, 2 instances) queries.

We compare two approaches in the data-oriented identification of instances: one where we use the *focus word* in the vectors to be classified, and one where the actual focus word is left out. Approaches that use the focus word in the feature vector may be biased to this feature. In the worst case, only the instances in the training set will be recognized.

To describe the context of the focus word, we construct a vector with the 5 tokens preceding and following. Moreover, for each of these context features, we add one of the more general feature discussed in Chapter 3. The general feature describing the focus word is maintained for both approaches.

The vectors in the training set are labeled with one of the three following classes: *start*, *intern*, *not*. From the classified feature vectors in the test set, all focus words labeled *start* and possible subsequent focus words labeled *intern* are extracted.

We evaluate the instances identified using the complete list of countries from the *CIA factbook*. A term is thus considered a country when its exact string representation is found in this list, and incorrect in all other cases.

We use the number of times the instances are identified as a confidence measure p for the correctness and sort all instances found by decreasing occurrence. In

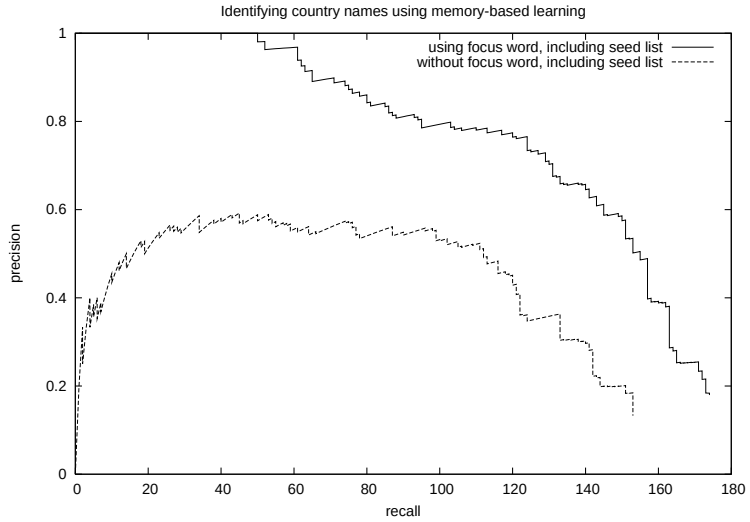

Figure 4.1. Recall (# countries identified, including the countries in the seed list) and precision for the instance identification.

Figure 4.1 the results of the two alternative classification approaches are compared: the approach with the focus word in the feature vector and the one without. We plot the precision of the n most frequently found terms against the absolute recall.

The figure shows that the 50 most frequently found instances are all correct if we use the approach with the focus word in the vector. The precision drops steeply for recall above 120 instances. For the other approach, errors occur among the best rated instances, but the precision is quite constant for recall levels between 20 and 100.

In Table 4.8 we give the most frequent instances that are evaluated to be incorrect. The table shows that many of these instances are geographic locations (regions, nations, continents) or variations on country names (e.g. *The UK*, *Myanmar*). *Taiwan* is not recognized as a sovereign country by the USA and not included in the CIA factbook.

Other errors include common words such as *The* and *What* and wind directions. While for example the multi-word terms *United States*, *Saudi Arabia* and *South Korea* are recognized, we also identify parts of these names (e.g. *Arabia*, *United*) as a separate instance.

As the approach using the focus word in the vector may be biased towards the instances that occur in the training set, we also compare the results by leaving out the country names starting with A – D in the evaluation set.

In Figure 4.2 the results are compared for the instances found that were not

The	Countries	North	Zealand
Europe	Thai	African	That
America	How	London	Shop
United	American	Saudi	They
States	Which	International	National
This	South	Asia	Taiwan
USA	There	California	Sea
What	Flutter	The UK	Gold
Country	However	Western	Rib
Korea	England	People	Napa
Sri	Arabia	Saturday	Cape
European	BALI	Our	Myanmar

Table 4.8. The best-ranked incorrect terms found using the classification without the focus word.

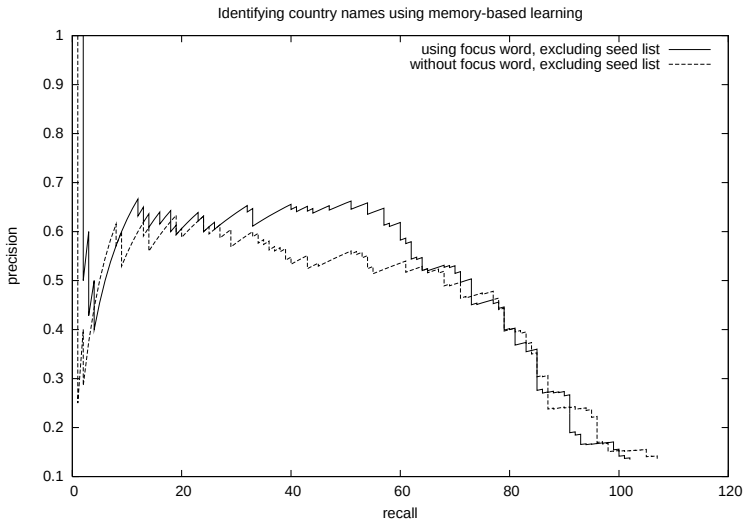

Figure 4.2. Recall (# countries identified, excluding the countries in the seed list) and precision for the instance identification.

included in the mentioned seed list. This figure shows that none of the two methods is clearly outperformed by the other. This is an expected result, as for this evaluation set the focus word itself does not contain relevant information.

As the results of this experiment cannot be compared with previous work, the quality of the results is hard to judge. We conclude that at precision levels of

around 0.5 up to 80 new instances can be identified. Among the false positive are many synonyms of country names and other geographic entities. The total set of countries has a size of 270 instances, while 73 countries were used in the annotation phase. Given the limited number of queries and the simplicity of the method, we are encouraged by these results. Analysis of the erroneous results shows that many false positives are geographic locations or parts of instances identified more frequently (e.g. apart from *Sri Lanka* also *Sri* and *Lanka* are extracted).

By adding check functions, the precision can be improved. Alternatively, using Cimiano and Staab's method [2004], we can distinguish between the various geographic locations found. For example, if more evidence is found that *London* is a city or *Asia* is a continent, the hypothesis that *London* and *Asia* are countries can be rejected.

4.4 The Presidents of the United States of America

In this case-study, we focus on an ontology describing the presidents of the US. We choose this setting as the required information can be expected to be redundantly available and the evaluation of the extracted information is relatively easy as an undisputable complete list of presidents is available as ground truth.

We want to identify a complete list of all past presidents (from *George Washington* to *George W. Bush*). We define the initial ontology as follows. Given are the classes named *US President* and *Rank* and the relations *succeeded* (with subject and object class *US President*) and *order* (on *US President* and *Rank*). The instances of the complete class *rank* (*first president*, *second president* .. *50th president*) are given beforehand. Note that the current president, George W. Bush, is the 43rd in line and the last instances are added for evaluation purposes.

Using this set-up, we focus on two tasks. First we populate the relations with a given complete class *US President*. The second task is to populate the ontology where *US President* is incomplete. We compare a rule-based approach with instance identification using memory-based learning.

4.4.1 Identifying Relations

We apply the ontology population on the US president ontology with complete classes starting with pattern *[US President] was the [Rank]* for *order*-relation and *[US President] succeeded [US President]* for the *succeeded* relation.

Again, we use the snippets and page titles found with the search engine. The most frequent relations found are used to identify new patterns.

In Section 3.1 we argued that for non-functional relations *broad* patterns are to be selected. For functional relations on the other hand, patterns are to be selected that connect the subject instance to few object instances. As the two relations are

	NON-FUNCTIONAL		FUNCTIONAL	
iteration	succeeded	order	succeeded	order
1	0.88	0.89	0.88	0.89
2	0.67	0.94	0.76	0.94
3	0.71	0.94	0.81	0.94
3	0.57	0.94	0.79	0.94
4	0.52	0.94	0.81	0.94

Table 4.9. F-Measure per iteration for the given relations.

in general functional[6], we expect the performance of the ontology population algorithm with the functional-setting for the two relations to have the best performance.

We evaluate the populated ontologies after each iteration. For the *order* relation, we focus on the president that is most often found for a given rank. For example, we take instance *26th* and extract the most frequent relation instance containing *26th*. The *succeeded* relation is evaluated in a similar manner. As *Grover Cleveland* succeeded both *Chester Arthur* and *Benjamin Harrison*, we focus on the *two* most frequently occurring relation instances with *Grover Cleveland* as subject instance.

In Table 4.9 the F-measures (combining precision and recall, page 20) of the extracted relation instances is given for the first four iterations.

As the ontology populated after the first iteration is based on the two manually selected patterns, the results do not differ for this iteration. For the subsequent iterations, we observe differences between the functional and non-functional approach in the *succeeded* relation. The F-measure for both approach is less than the one for the first iteration. The non-functional results deteriorate as more and more patterns are added (e.g. *'(president) and (president)'*) that do not express the intended relation. No differences are observed for the results for the *order* relation. This can be explained by the fact that few other relations are imaginable that combine the instances of the two classes. Moreover, as the vast majority of the instances were added in the first iterations, the ranking is stabilized after iteration 2.

We conclude that the ontology population method using the search engine snippets gives good results. The results for the *succeeded* relation show the effect of the distinction between functional and non-functional relations.

4.4.2 Identifying Instances

Having focussed on a task with complete classes above, the class *president* is now empty. Hence, the instances of *US President* are initially to be found using the

[6] Grover Cleveland was the only president in two non-subsequent terms.

order-relation as no queries can be formulated for the *succeeded*-relation.

In two alternative runs of the algorithm, the instances of *US President* are identified as follows.

- *Rule-based approach.* We have formulated a regular expression describing person names. We accept the longest sequence with either two or three capitalized words, or a sequence with two capitalized words and a capital and period in between (e.g. *John F. Kennedy*).
- *Using Memory-based learning.* Like the previous experiment in Section 4.3, we classify vectors describing the search results and extract instances from these vectors. We use the ten most recent presidents in the training set for the first iteration. In the feature vectors generated from the corresponding search results, we do not include the focus words in the classification. Hence, the ten presidents in the training set are not instantly added when populating the ontology but have to be extracted from the search results.

Compared to the experiment in Section 4.3, the training set for the memory-based learning approach is small. We are interested whether we can find a long list of instances using the bootstrapping mechanisms.

The evaluation of the extracted instances is not straightforward, as many alternatives may refer to the same president (e.g. *President Clinton*, Bill Clinton, *William Jefferson Clinton*). We therefore decided to automatically evaluate the instances found using *Google*'s define functionality.

We consider an extracted instance to be a president of the United States if:

- indeed definitions are found for the given term, *and*
- the word *president* is found in at least one of the definitions, *and*
- the terms *United States* or *US* are found in at least one of the definitions.

When we inspect the results for the populated ontology, we encountered many terms that refer to vice-presidents, presidents of other countries and other statesmen. Although these instances are no correct instances of *US President*, one could argue that they are instances in some superclass *politician*. We therefore propose a second evaluation where we focus on politicians in general. Using *Google define*, we consider an instance to be a politician if:

- definitions are found for the term, and
- at least one of the words *president*, *minister*, *leader*, *statesmen* or *politician* is found in the definitions.

Naturally, US presidents are included in this definition.

Figure 4.3 gives the precision and absolute recall for the first iteration. It shows that using the rule-based method in total 120 terms were found that refer to US

	RULE-BASED		USING MBL	
	succeeded	order	succeeded	order
iteration 1.	0.95	0.94	0.25	0.27
iteration 2.	0.98	1.00	0.25	0.27

Table 4.10. F-Measure for the first iteration for the given relations.

presidents, while history only has known 42 distinct presidents. Using the rule-based approach all presidents were identified, the last one – James Buchanan – at a precision level of 0.47. Hillary Clinton is said to be the 44th president of the United States[7].

Using the rule-based approach, in total 120 correct distinct variations of the names of the 42 presidents were found. Apart from these 120, we also found 61 names of presidents of other countries and 42 other politicians.

The performance for the approach using memory-based learning is disappointing. As only few names were included in the training set, classification of the context is biased towards these names. The only presidents identified are also present in the initial training set. As we do not abstract from the context, but only from the focus word, words in the context like *Bill* and *Clinton* signal the presence of an *intern* or *start* vector. For example, a focus word is only classified as *intern* when it is preceded by one of the first names of the presidents in the training set (e.g. *Bill* or *Ronald*). As no new president names are learned, the performance does not improve in the next iterations, when newly identified patterns are applied.

With respect to the relation instances found, the F-measures for the first two iterations are given in Table 4.10. As only few names were identified using memory-based learning, also the F-measures for the found relations are not convincing. The relations using the rule-based approach are precisely identified. With respect to the second iteration using the rule-based approach, only the succeeded relations between *James Polk* and *John Tyler* and between *James Monroe* and *James Madison* were not extracted, while the precision for the *succeeded*-related was 1.

We conclude that for the given task, the simple rule-based method gives convincing results. On the other hand, the approach using automatically annotated training data is disappointing. Especially, as the search results for the *rank* have shown to contain all relevant data to populate the ontology.

[7]The experiment was conducted in February 2008.

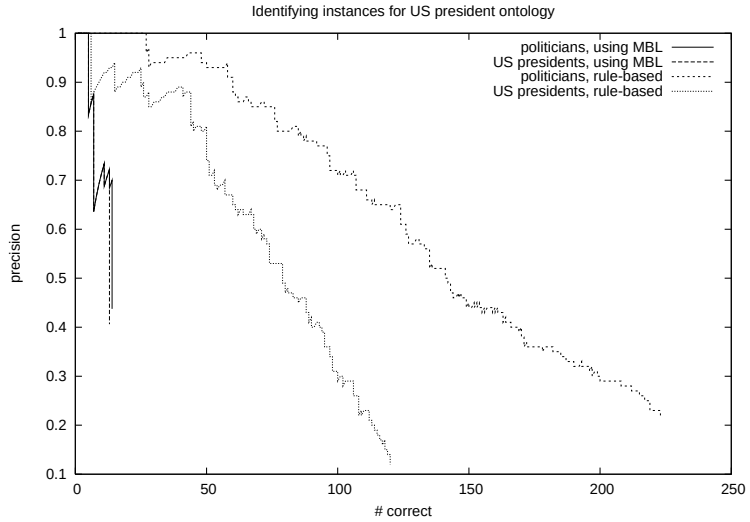

Figure 4.3. Recall and precision for the instance identification.

4.5 Extracting Historical Persons from the Web

In this last section of this chapter we focus on the population of an ontology on historical persons. Such information is currently not present as such on the Web. By combining and structuring the knowledge on diverse web pages, we are able to find answers to questions as *Who are important novelists from Ireland?*, *Which notable people were born in 1648?*, *Who are popular female composers?*. To present the information in an attractive manner, we compute a *fame* rank for each person based on the presence on the web.

In the given initial ontology (cf. Figure 4.4) all classes but *Person* are complete, while *Person* is empty. The class *Period* contains all combinations of years that are likely to denote the periods of life of a historical person. For example *'1345 - 1397'* is an instance of *Period*. The class *Nationality* contains all names of countries in the world. We identify derivatives of country names as well and use them as synonyms (e.g. *American* for *United States*, *English* for *United Kingdom* and *Flemish* for *Belgium*). A hierarchy among the instances of *Nationality* is defined using the names of the continent, such that we can for example select a list of historical persons from Europe. Likewise, the instances of *Profession* reflect 88 professions. For the instances *male* and *female* we have added a list of derivatives to be used as synonyms, namely the terms *he*, *his*, *son of*, *brother of*, *father of*, *man* and *men* for *male* and the analogous words for *female*. We use the class *Fame* to rank the retrieved instances of *Person* according to their presence on the web. Hence the task is to identify a collection of biographies of historical persons and to identify a social network between the persons found. As persons may have more than

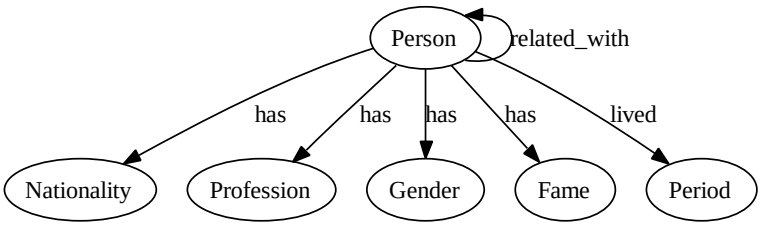

Figure 4.4. The ontology on historical persons to be populated.

one profession and can be related to multiple other people, we are interested in a ranked list of professions and related persons for each historical person found. For efficiency reasons, we only extract information from the snippets returned by the search engine.

We use the given instances in the ontology O to populate the class *Person*, i.e. to find a list of names of historical persons. We again use the instances available in the given ontology and combine these with patterns to formulate queries and hence create a corpus to extract information from.

Suppose we use instances in the class *Profession* to extract the persons. When querying for the instance *composer*, it is likely that few well-known composers dominate the search results. As we are interested in a rich ontology of historical persons, this is thus a less-suited approach.

The class *Period* contains all combinations of years that are likely to denote the periods of life of a historical person. Hence, the number of instances known for the class *Period* is by far the largest for all complete classes in O. As it is unlikely that many important historical persons share both a date of birth and a date of death, the use of this class is best suited to obtain a long and diverse list of persons. The names of historical persons are often followed in texts by a period in years (e.g. *'Vincent van Gogh (1853 - 1890)'*). As this period is likely to denote the period he or she lived in, we choose the pattern "(year of birth – year of death)" to collect snippets to identify the names of historical persons.

4.5.1 Identifying Person Names in Snippets

Having obtained a collection of snippets, the next problem is to extract instances from the texts, in this case person names. We choose to identify the names within the snippets using a rule-based approach.

First we extract all terms directly preceding the queried expressing that match a regular expression similar to the approach in the previous case study. That is, we extract terms of two or three capitalized words and compensate for initials, inversions (e.g. *'Bach, Johann Sebastian'*), middle names, Scottish names (e.g. *McCloud*) and the like.

Subsequently, we remove extracted terms that contain a word in a tabu list (e.g. *'Biography'*) and names that only occur once within the snippets. Having filtered out a set of potential names of persons, we use a string matching among the extracted names to remove typos and names extracted for the wrong period.

Using the 80,665 periods identified, we obtain a list of 28,862 terms to be added as instance to the class *Person*. Simultaneously, we extract the relations between the periods queried and the extracted instances.

In the evaluation we analyze the quality of the extracted instances and compare the rule-based approach with a state-of-the-art named entity recognizer using a hidden markov model [Duda et al., 2000].

4.5.2 Using Mined Names to Find Additional Biographical Information

Having found a list of instances of the class *Person*, we first determine a ranking of the instances extracted.

Finding a Rank. To present the extracted information in an entertaining manner, we determined the number of hits for each identified person. As names are not always unique descriptors, we queried for the combination of the last name and period (e.g. *'Rubens (1577 - 1640)'*). Although the number of hits returned a search engine is an estimate and irregularities may occur [Véronis, 2006], we consider this simple and efficient technique to be well suited for this purpose.

Now we use the names of these instances in a similar fashion to acquire biographical information for the 10,000 best ranked persons. To limit the number of queries per instance, we select the pattern 'was' to reflect the relation between *Person* on the one hand and *Nationality*, *Gender* and *Profession* on the other hand. By querying phrases such as *'Napoleon Bonaparte was'* we thus expect to acquire sentences containing the biographical information. Table 4.11 contains examples of the sentences used to determine biographical information. We scan these sentences for occurrences of the instances (and their synonyms) of the related classes.

Relating persons to a gender. We simply counted instances and their synonyms within the snippets that refer to the gender *'male'* the opposite words that refer *'female'*. We simply related each instance of *Person* to the gender with the highest count.

Relating persons to a nationality. We assigned the nationality with the highest count.

Relating persons to professions. For each person, we assigned the profession p that most frequently occurred within the snippets retrieved. Moreover, as persons may have multiple professions, all other professions with a count at least half of the count of p were added.

Hence, using one query per instance of *Person*, we identify basic biographical information.

Napoleon Bonaparte was the greatest military genius of the 19th century
Napoleon Bonaparte was born of lower noble status in Ajaccio, Corsica on August 15, 1769
Napoleon Bonaparte was effectively dictator of France beginning in 1799 and
Napoleon Bonaparte was the emperor of France in the early 1800s
Napoleon Bonaparte was a bully, rude and insulting
Napoleon Bonaparte was in Egypt and was not enjoying his tour
Napoleon Bonaparte was a great warrior and a notable conqueror
Napoleon Bonaparte was born on August 15, 1769 to Carlo and Letizia Bonaparte
Napoleon Bonaparte was defeated at Waterloo

Table 4.11. Example search results for the query 'Napoleon Bonaparte was'.

philosopher	1275	designer	222
composer	804	scientist	215
mathematician	773	musician	213
poet	668	historian	210
physicist	501	inventor	208
writer	478	essayist	201
playwright	469	engineer	199
novelist	429	singer	198
sculptor	362	dramatist	186
author	352	theorist	175
critic	346	illustrator	171
astronomer	343	journalist	166
painter	329	statesman	138
politician	323	teacher	138
artist	286	mystic	133
architect	284	educator	132
director	270	theologian	127
conductor	267	physician	125
actor	261	printmaker	124
pianist	224	scholar	112

Table 4.12. The professions that were found most often.

4.5.3 Evaluating the Identified Biographical Information

The rank assigned to each of the persons in the list provides a mechanism to present the extracted data in an attractive manner. Table 4.13 gives the list of the 25 best ranked persons and the identified biographical information. Using the criterion defined in Section 4.5, Johann Sebastian Bach is thus the best known historical figure.

As the data is structured, we can also perform queries to select subsets of the

Johann Sebastian Bach (1685-1750)	Germany	composer,organist
Wolfgang Amadeus Mozart (1756-1791)	Austria	composer,musician
Ludwig van Beethoven (1770-1827)	Germany	composer
Albert Einstein (1879-1955)	Germany	scientist,physicist
Franz Schubert (1797-1828)	Austria	composer
Johannes Brahms (1833-1897)	Germany	composer
William Shakespeare (1564-1616)	United Kingdom	author,poet
Joseph Haydn (1732-1809)	Austria	composer
Johann Wolfgang Goethe (1749-1832)	Germany	philosopher,director,poet..
Charles Darwin (1809-1882)	United Kingdom	naturalist
Robert Schumann (1810-1856)	Germany	composer
Leonardo da Vinci (1452-1519)	Italy	artist,scientist,inventor
Giuseppe Verdi (1813-1901)	Italy	composer
Frederic Chopin (1810-1849)	Poland	composer,pianist,poet
Antonio Vivaldi (1678-1741)	Italy	composer
Richard Wagner (1813-1883)	Germany	composer
Ronald Reagan (1911-2004)	United States	president
Franz Liszt (1811-1886)	Hungary	pianist,composer
Claude Debussy (1862-1918)	France	composer
Henry Purcell (1659-1695)	United Kingdom	composer
Immanuel Kant (1724-1804)	Germany	philosopher
James Joyce (1882-1941)	Ireland	author
Friedrich Schiller (1759-1805)	Germany	poet,dramatist
Georg Philipp Telemann (1681-1767)	Germany	composer
Antonin Dvorak (1841-1904)	Czech Republic	composer

Table 4.13. The 25 historical persons with the highest rank.

full ranked list of persons. For example, we can create a list of best ranked artists (Table 4.14), or a 'society' of poets (Table 4.15). We note that Frédéric Chopin is often referred to as 'the poet of the piano'. Table 4.16 shows that Vincent van Gogh is the best ranked Dutch painter.

In Table 4.14 we give the top-40 persons that have as first profession either artist or painter. Persons that also have as one of their professions artist or painter, but not as their highest-scoring profession are Sir Winston Churchill, John Ruskin and Kahlil Gibran. Their highest-scoring professions are politician, author and poet, respectively.

The reader can verify that the given list of extracted persons are highly accurate. However, lacking a benchmark set of *the* best known historical persons, we manually evaluated samples of the extracted ontology to estimate precision and recall.

Precision. To estimate the precision of the class *Person*, we selected three decennia, namely 1220-1229, 1550-1559 and 1880-1889, and analyzed for each the candidate persons that were found to be born in this decennium. For the first two decennia we analyzed the complete list, for decennium 1880-1889 we analyzed only the first 1000 as well as the last 1000 names. This resulted in a precision of

Leonardo da Vinci (1452 - 1519)	Italy	artist, scientist,...
Pablo Picasso (1881 - 1973)	Spain	artist
Vincent van Gogh (1853 - 1890)	Netherlands	artist, painter
Claude Monet (1840 - 1926)	France	artist, painter,...
Pierre-Auguste Renoir (1841 - 1919)	France	painter
Paul Gauguin (1848 - 1903)	France	painter
Edgar Degas (1834 - 1917)	France	artist, painter,...
Paul Cezanne (1839 - 1906)	France	painter, artist
Salvador Dali (1904 - 1989)	Spain	artist
Henri Michaux (1899 - 1984)	Belgium	artist, poet
Gustav Klimt (1862 - 1918)	Austria	painter, artist
Peter Paul Rubens (1577 - 1640)	Belgium	artist, painter
Katsushika Hokusai (1760 - 1849)	Japan	painter
Amedeo Modigliani (1884 - 1920)	Italy	artist, painter
JMW Turner (1775 - 1851)	United Kingdom	artist, painter
James Mcneill Whistler (1834 - 1903)	United States	artist
Rene Magritte (1898 - 1967)	Belgium	artist, painter
Henri Matisse (1869 - 1954)	France	artist
Rembrandt van Rijn (1606 - 1669)	Netherlands	artist, painter
Edouard Manet (1832 - 1883)	France	artist, painter
Herm Albright (1876 - 1944)	-	artist, engraver,...
Marc Chagall (1887 - 1985)	Russia	painter, artist
Edvard Munch (1863 - 1944)	Norway	painter, artist
Wassily Kandinsky (1866 - 1944)	Russia	artist, painter
Francisco Goya (1746 - 1828)	Spain	artist, painter

Table 4.14. The 25 artists with the highest rank.

0.94, 0.95, and 0.98, respectively. As the decennium of 1880-1889 resulted in considerably more names, we take a weighted average of these results. This yields an estimated precision for the complete list of 0.98.

We compare the precision of the rule-based approach with a state-of-the-art machine-learning-based algorithm, the Stanford Named Entity Recognizer (SNER [Finkel et al., 2005]), trained on the CoNLL 2003 English training data. Focussing on persons born in the year 1882, using the rule-based approach we extracted 1,211 terms. SNER identified 24,652 unique terms as person names in the same snippets. When we apply the same post-processing on SNER extracted data (i.e. removing typos by string matching, single-word names and names extracted for different periods), 2,760 terms remain, of which 842 overlap with the terms extracted using the rule-based approach.

We manually inspected each of these 2,760 terms, resulting in a precision of only 62%. Around half of the correctly extracted names are not recognized by the rule-based approach, most of them due to the fact that these names did not directly preceded the queried period.

To estimate the precision of the extracted biographical relations, we inspected randomly selected sublists of the top 2500 persons. When we focus on the best

William Shakespeare (1564-1616)	United Kingdom	author,poet
Johann Wolfgang Goethe (1749-1832)	Germany	poet, psychologist, philosopher..
Frederic Chopin (1810-1849)	Poland	composer,pianist,poet
Friedrich Schiller (1759-1805)	Germany	poet,dramatist
Oscar Wilde (1854-1900)	Ireland	author,poet
Jorge Luis Borges (1899-1986)	Argentina	author,poet
Victor Hugo (1802-1885)	France	author,poet,novelist
Ralph Waldo Emerson (1803-1882)	United States	poet,philosopher,author
William Blake (1757-1827)	United Kingdom	poet
Dante Alighieri (1265-1321)	Italy	poet
Robert Frost (1874-1963)	United States	poet
Heinrich Heine (1797-1856)	Germany	poet
Robert Louis Stevenson (1850-1894)	Samoa	engineer,author,poet
Alexander Pope (1688-1744)	United Kingdom	poet
Hildegard von Bingen (1098-1179)	Germany	composer,scientist,poet
Lord Byron (1788-1824)	Greece	poet
John Donne (1572-1631)	United Kingdom	poet,author
Henri Michaux (1899-1984)	Belgium	poet
Walt Whitman (1819-1892)	United States	poet
Robert Burns (1759-1796)	United Kingdom	poet

Table 4.15. The 20 best ranked poets.

Vincent van Gogh (1853-1890)	Kees van Dongen (1877-1968)
Rembrandt van Rijn (1606-1669)	Willem de Kooning (1904-1997)
Johannes Vermeer (1632-1675)	Pieter de Hooch (1629-1684)
Piet Mondrian (1872-1944)	Jan Steen (1626-1679)
Carel Fabritius (1622-1654)	Adriaen van Ostade (1610-1685)

Table 4.16. The 10 best ranked painters from the Netherlands.

scoring professions for the 2500 persons, we estimate the precision of this relation to be 96%. We did not encounter erroneously assigned genders, while we found 98% of the cases the right *Nationality*, if one is found.

Hence, we conclude that the ontology populated using the rule-based approach is precise.

Recall. We estimate the recall of the instances found for *Person* by choosing a diverse set of six books containing short biographies of historical persons. Of the 1049 persons named in the books, 1033 were present in our list, which gives a recall of 0.98 (Table 4.18).

From Wikipedia, we extracted a list of important 1882-born people[8]. The list contains 44 persons. Of these 44 persons, 34 are indeed mentioned in the Google snippets found with the queried patterns. Using the rule-based approach, we iden-

[8] `http://en.wikipedia.org/wiki/1882`, visited January 2007

Cesar Franck (1822 - 1890, B)	organist, composer, pianist
Vincent van Gogh (1853 - 1890, NL)	artist, painter
Roland de Lassus (1532 - 1594, B)	composer
Abraham Kuyper (1837 - 1920, NL)	theologian, politician
Henri Michaux (1899 - 1984, B)	artist, poet
Peter Paul Rubens (1577 - 1640, B)	artist, painter
Baruch Spinoza (1632 - 1677, NL)	philosopher
Rene Magritte (1898 - 1967, B)	artist, painter
Christiaan Huygens (1629 - 1695, NL)	astronomer, scientist,...
Rembrandt van Rijn (1606 - 1669, NL)	artist, painter
Johannes Vermeer (1632 - 1675, NL)	painter, artist
Edsger Wybe Dijkstra (1930 - 2002, NL)	computer scientist
Anthony van Dyck (1599 - 1641, B)	painter
MC Escher (1898 - 1972, NL)	artist
Antony van Leeuwenhoek (1632 - 1723, NL)	scientist
Piet Mondrian (1872 - 1944, NL)	artist, painter
Hugo Grotius (1583 - 1645, NL)	lawyer, philosopher,...
Jan Pieterszoon Sweelinck (1562 - 1621, NL)	composer, organist,...
Andreas Vesalius (1514 - 1564, B)	physician
Hieronymus Bosch (1450 - 1516, NL)	painter
Audrey Hepburn (1929 - 1993, B)	actress, princess
Ferdinand Verbiest (1623 - 1688, B)	astronomer
Desiderius Erasmus (1466 - 1536, NL)	philosopher, reformer,...
Theo van Gogh (1957 - 2004, NL)	judge, artist
Gerard Dou (1613 - 1675, NL)	painter, artist
Nicolaas Beets (1814 - 1903, NL)	king, poet, writer
Carel Fabritius (1622 - 1654, NL)	painter
Georges Simenon (1903 - 1989, B)	author
Kees van Dongen (1877 - 1968, NL)	painter
Gerardus Mercator (1512 - 1594, B)	cartographer
Emile Verhaeren (1855 - 1916, B)	poet, dramatist
Abel Janszoon Tasman (1603 - 1659, NL)	explorer
Pieter de Hooch (1629 - 1684, NL)	painter
Jan van Goyen (1596 - 1656, NL)	artist
Hendrick Goltzius (1558 - 1617, NL)	artist
Simon Stevin (1548 - 1620, NL)	mathematician
Jacob Jordaens (1593 - 1678, B)	artist, painter
Jan Steen (1626 - 1679, NL)	artist, painter,...
Jacobus Arminius (1560 - 1609, NL)	theologian
Guillaume Dufay (1400 - 1474, B)	composer

Table 4.17. The Belgian/Dutch persons with the highest rank.

tified 24 of these persons within the snippets. The other ones were only mentioned once (and hence not recognized) or found in different places in the snippets, i.e. not directly preceding the queried period. Using SNER, we identified 27 persons from the Wikipedia list.

For the recall of the identified biographical relations, we observe that for the 10,000 persons that we considered all were given a gender, 77% were given a nationality, and 95% were given one or more professions.

Hence, we conclude that using simple methods we have extracted reliable in-

BOOK	TOTAL	FOUND	RECALL
The Science Book	156	147	0.94
The Art Book	358	353	0.99
The Dutch Painters: 100 Seventeenth Century Masters	108	106	0.98
Philosophy: 100 Essential Thinkers	78	78	1.00
Herinneringen in Steen	195	195	1.00
Scientists and Inventions	154	154	1.00

Table 4.18. Recall for six popular scientific editions.

formation on historical persons and their biographies with good recall.

4.6 Conclusions

We evaluated the method to populate an ontology using a web search engine using a number of case-studies. We show that simple web information extraction techniques can be used to precisely populate ontologies. For all studied cases, the snippets showed to be a sufficient corpus to extract the information from. By combining and structuring information from the Web, we create a valuable surplus to the knowledge already available.

The use of the pattern-instance combinations in queries is an effective approach to access relevant search results. We have shown that the redundancy of information on the web enables us to precisely identify instances using the rule-based approach. For the data-oriented approach, the use of a large and representative set of known instances is crucial.

The relation instances in the various case-studies were precisely identified using the pattern-based approach. Both with manually constructed patterns as well as with learned patterns good results were achieved in the studied cases.

5

Application: Extracting Inferable Information From the Web

In the previous chapter, we have focused on the extraction of factual information that is easily verifiable using structured content on the web. The evaluations give confidence in the quality of the output of the method.

Now we focus on the extraction of information that is not present as such on the web, but can be inferred by combining data extracted from multiple websites. We do so by presenting two case studies. In Section 5.1 we focus on an information demand from the *Nederlands instituut voor Beeld en Geluid*. Having defined a thesaurus (or, an ontology) of keywords, the question is how to link a user-input term to a keyword that is semantically closest. We use methods developed in the previous chapters to address this task. Section 5.1 is based on [Geleijnse & Korst, 2007].

Section 5.2 focusses on the extraction of lyrics from the web. With numerous fanpages, legal and less legal websites on lyrics, it is not straightforward to find a reliable version on the lyrics for a given song. Section 5.2 is based on [Geleijnse & Korst, 2006a] and [Korst & Geleijnse, 2006] as well as the related patent applications (page 166). We use a rule-based approach to identify lyrics within pages found with a search engine and combine all versions into a most plausible version for the given song.

5.1 Improving the Accessibility of a Thesaurus-Based Catalog

To consistently annotate items and to facilitate their retrieval, cultural heritage institutions often use controlled vocabularies for indexing their collections. For the *Nederlands instituut voor Beeld en Geluid*[1] (B&G), the annotations are currently the only basis to retrieve the audiovisual content. The maintenance of the annotations as well as the addition of new annotated material is a costly and laborious task. Many of the hundreds of thousands of documents are therefore only sparsely annotated.

B&G uses a dedicated thesaurus, the GTAA (Gemeenschappelijke Thesaurus Audiovisuele Archieven (*Common Thesaurus Audiovisual Archives*), as a controlled vocabulary. Especially for the items that are briefly annotated, e.g. where a summary of the content is missing, searching for GTAA terms is the most effective mechanism for retrieval.

Although the use of a controlled vocabulary such as the GTAA provides a uniform annotation over the whole collection, it also gives rise to two problems. On the one hand, the retrieval of items – both for professionals and for the general public – depends on the knowledge of the content of the GTAA. Proper use of the terms in the GTAA is crucial for both indexing and retrieval. On the other hand, the controlled vocabulary is updated from time to time as new terms become relevant. B&G choose to limit the size of their controlled vocabulary. Therefore, all annotations that contain terms that are removed from the vocabulary have to be updated, as *expired terms* are mapped to terms within the latest version of the GTAA.

As proper use of the GTAA is of value for the accessibility of the B&G catalog, we focus on an assistant to identify proper terms within the thesaurus for a search demand. Given an arbitrary search term, we want to identify GTAA terms with a similar meaning. Such a mapping between the term and the GTAA can be of assistance for those who want to search the catalog as it will provide more effective search results. For those annotating an audiovisual production it can also be of use, as it can help to find the closest terms within the GTAA.

For many languages, such as Dutch, no structured knowledge is available to derive a mapping between an arbitrary term and the thesaurus. We therefore use unstructured texts to extract such a mapping, by deploying techniques developed in the fields of ontology mapping and web content mining. We derive semantic relations between a query term and the thesaurus using search engine snippets.

We illustrate that the method presented is domain and language independent by evaluating mappings of terms both to the Dutch GTAA and to the Agricultural Thesaurus[2] (NALT) of the United States National Agricultural Library.

[1] The Netherlands Institute for Sound & Vision, `http://www.beeldengeluid.nl`

[2] `http://agclass.nal.usda.gov/agt/agt.shtml`

5.1.1 Related Work

Together with the development of the semantic web, the research topic of ontology matching arose [Shvaiko et al., 2006]. In ontology matching, the task is to combine or create relations between two separately designed ontologies. Although most approaches are based on the structures of the ontologies combined with lexical matches (e.g. [Shvaiko & Euzenat, 2005; Meilicke, Stuckenschmidt, & Tamilin, 2007]), the use of web content mining has recently been deployed for this task [Gligorov et al., 2007; Van Hage, Kolb, & Schreiber, 2006].

Web information extraction applied to the cultural heritage domain is addressed by De Boer et al. [2007]. Here, ontologies of painters and art movements are linked by analyzing web pages on art movements. The numbers of search engine hits is used in [De Boer et al., 2006] to identify the periods corresponding to art styles. In [Cilibrasi & Vitanyi, 2007] such numbers are used to identify relatedness between Dutch 17th century painters. In [Navigli & Velardi, 2006] a method is presented to create structured knowledge on the arts domain using the definitions in a glossary. Patterns in the glosses are used to identify relations. These relations link the concept to a named entity, extracted using a NER.

As an alternative approach to the use of web content mining to improve the accessibility of the catalog, Malaisé et al. [2007] created a method to link the Dutch GTAA thesaurus to the English WordNet [Fellbaum, 1998] via a bilingual online dictionary. As the GTAA contains many multi-word terms and compounds, such a mapping can not always be found. Moreover, it is not trivial to link an arbitrary given term via WordNet to the GTAA.

5.1.2 Problem Description and Outline

Given is a thesaurus, i.e. a list of terms and their semantic relations. Typical relations are the *broader term* relation (BT) between a term and a more general term (e.g. *herring gull* and *seagull*), its counterpart the *narrower term* relation (NT), and the *related term* RT relation for terms that are associated with one another. Moreover, a thesaurus can contain preferred and non-preferred terms. The latter refer to the first via the *use* relation (US), *used for* (UF) is its inverse. As a thesaurus solely consists of a set of terms and their mutual relations, it can easily be described using the terminology posed in Chapter 2. Van Assem et al. [2004] proposed a mechanism to convert a thesaurus to semantic web format.

Apart from the standard thesaurus relations, the GTAA also distinguishes one or more categories for each preferred term. These categories are subdivided into 15 main categories (e.g. *sports and leisure*) and each containing 3 to 7 subcategories (e.g. *recreation*). The terms in the GTAA are mostly in plural, but the singular forms are added as well.

Example terms from both the GTAA and the NALT are given in Tables 5.1 and

bioscooppersoneel	*(cinema personnel)*
1D05.03	*economy – trades, services*
1D12.01	*arts and culture – general*
1D13.02	*sports and leisure – recreation*
BT	personeel *(personnel)*
BT	werknemers *(employees)*
NT	filmoperateurs *(film operators)*
RT	bioscopen *(cinemas)*
RT	film *(film)*
UF	explicateurs (± *silent film commentator)*

Table 5.1. Example terms from the GTAA

earthworms	
BT	invertebrates
BT	soil invertebrates
RT	earthworm burrows
RT	Lumbricidae
RT	vermiculture
RT	worm casts

Table 5.2. Example term from the the NALT

5.2. For the GTAA term the translations in English are given. For example, the entry shows that *invertebrates* is a broader term (BT) for *earthworms*.

Currently, detailed knowledge of the content of the GTAA thesaurus T is crucial for describing (and redescribing) items within the catalog. Moreover, the recall of briefly described items will improve when using search terms within the GTAA. Hence, an assistant is desired that suggests terms from the GTAA for a given query term. The problem addressed in this section is the following.

Thesaurus Mapping Problem. Given a term v and a thesaurus T, find the term $t \in T$ that is semantically closest. □

Ideally, for a given term v we are interested in a synonym of v within T. As the GTAA only consists of about 5,000 terms, it is not likely that a synonym is present for v. We therefore focus on finding the narrowest broader term t for v. For

example, if we are interested in a mapping for the term *albatross*, the terms *bird* and *animal* are indeed broader terms, but are too broad. *Seabird* however would be the narrowest broader term for *albatross*.

The algorithm presented is to be used as an assistant. For a given query v, we therefore present multiple candidate terms with hyperlinks to the GTAA. Even if t is not identified by the assistant, t can easily be found if the method does return terms that are semantically close (e.g. by the RT relation) and hence links to t. Instead of navigating through a thesaurus with 5,000 terms, the user is now only presented a handful of alternatives. Hence, the user can select a term at a single glace.

As no suitable structured information is available for this task, we again use a pattern-based method to determine a mapping from v to the thesaurus. Using the *Yahoo!* API for our experiments, we are allowed to perform 5,000 automated queries a day. Approaches as discussed in e.g. [Cimiano & Staab, 2004; Cilibrasi & Vitanyi, 2007] have a query complexity of the order of the number of terms in T per query term v. We therefore aim at an approach more efficient in the number of queries per term.

As a first approximation, we start with determining the most relevant categories (Table 5.1) for v. We use the computed categories in the next steps, where we present three alternative approaches in mapping v to T.

5.1.3 Determining Categories

A commonly used paradigm in natural language processing is that the semantics of a term can be determined by its context [Manning & Schütze, 1999]. We use this assumption to first determine the subcategory – and thus main category – for the term v. For each subcategory r, we compute a score $s_v(r)$.

We collect the 100 snippets for the query `"v"` and we scan them for terms in T. Each term in T that occurs in the snippets contributes to the scores of its subcategories [Fleischman & Hovy, 2002]. Hence, if the term *bioscooppersoneel* (see Table 5.1) occurs in the snippets found with v, this occurrence contributes to the scores of the subcategories 1D05.03, 1D12.01 and 1D13.02.

As infrequent terms are more discriminative than frequent ones, the occurrences of the terms in T are weighted by their estimated total frequency on the web. Words such as *haar* (either *hair* or *her* in English) that appear frequently in Dutch texts get a lower score than infrequent terms such as *1 mei-vieringen* (May 1 celebrations).

The score $s_v(r)$ for subcategory r is given by a *tf.idf*-based weighting scheme [Manning & Schütze, 1999]

$$s_v(r) = \sum_{t \in r} \mathrm{oc}(t) \cdot \log \frac{C}{f(t)}, \tag{5.1}$$

where

$\mathrm{oc}(t)$ denotes the number of times term t (or its singular form) occurs ,

$f(t)$ gives the number of search engine hits for the query "t", and

$C = \sum_{t \in T} f(t)$ gives the sum of all hits.

After having computed the scores for each of the subcategories, we assign the subcategory $r_{\max}$ with the highest score to v. As a term can be within multiple categories, we also add the subcategories with at least half the score of $r_{\max}$. Hence, we add each subcategory r_i for which the following holds

$$s_v(r_i) \geq 0.5 \cdot s_v(r_{\max}). \tag{5.2}$$

We will use the subcategories in the mapping techniques described in the next three subsections.

5.1.4 Term Mapping using Hyponym Patterns

We assume a set of patterns to be given that relate Dutch terms with their hypernyms. [Hearst, 1992]. IJzereef [2004] manually constructed such a set.

Having such a set of patterns, we combine the term v with each of the patterns into queries. We query an expression (e.g. *such as puffins*) and scan the returned snippets for terms in T preceding the search term. Hence, the aim is to find phrases (like *seabirds such as puffins*) to determine broader terms for v.

For a term $t \in T$ found within the snippets for query term v, we compute its score $s_v(t)$ as follows.

$$s_v(t) = q(t,v) \cdot \mathrm{oc}(t) \cdot \log \frac{C}{f(t)} \tag{5.3}$$

We use $q(t,v)$ as a penalty score for terms outside the subcategories found in the previous section.

$$q(t,v) = \begin{cases} 1.0 & \text{if } t \text{ and } v \text{ share a subcategory} \\ 0.3 & \text{if } t \text{ and } v \text{ share a main category} \\ 0.1 & \text{if } t \text{ and } v \text{ share no category} \end{cases}$$

The values for $q(t,v)$ are chosen in a somewhat arbitrary way. We will return to these choices when discussing the experimental results.

Using the scores, we compute a ranked list for the potential hypernym terms for v found using this method.

5.1.5 Term Mapping using Enumeration Patterns

Snow et al. [2006; 2005] observe that related terms (or *siblings*) tend to co-occur in enumerations. We thus can state that enumerated items share a broader term. Hence, if we can observe which terms within T are siblings of v, we can use the structure of the thesaurus to compute the broader term for v.

Similar to the approach described in the previous section, we select a number of patterns expressing the RT relation. Again, we scan the snippets for terms within the thesaurus. However, we do not score the terms found, but (all) their broader terms. Hence, the presence of the term *aalscholvers* (cormorants) contributes to the scores for *watervogels* (water birds), *vogels* (birds), and *dieren* (animals).

A term t is hence scored using the presence of all its narrower terms $\text{NT}^*(t)$ in the snippets.

$$s_v(t) = \sum_{s \in \text{NT}^*(t)} q(s,v) \cdot \text{oc}(s) \cdot \log \frac{C}{f(s)} \tag{5.4}$$

We assume that the broadest concepts (e.g. *dieren*, animals – 26,000,000 hits) are in general more present on the web than narrower concepts (e.g. *watervogels*, waterbirds – 230,000 hits). Hence, we do take the distance of s to t into account as the factor $\frac{C}{f(s)}$ penalizes common concepts. Again, we compute a ranked list of potential hypernym terms using this enumeration-based approach.

5.1.6 Term Mapping using a Lexical Approach

We observe that hyponym-hypernym pairs that are lexically similar (e.g. *dienstverlenende beroepen* and *beroepen*, *earthworms* and *worms*) occur infrequently within the same sentence. Next to the two approaches based on web information extraction, we therefore adopt an approach using the morphology of the terms.

If some term t in T is a suffix of v, then v may be a hypernym of t (e.g. if v contains a preceding adjective). However, not all t that match with a suffix of v are indeed hypernyms of v. For example, the GTAA term *ogen* (eyes) is a suffix of *psychologen* (psychologists).

However, if the computed categories for v do not overlap with the categories for suffix t, it is not likely that the two are related. We therefore use the subcategories as computed in Section 5.1.3 to filter out erroneous lexical mappings.

We construct a list of thesaurus terms that are suffixes of v and share a subcategory with v. If no such terms exist, we create such a list of terms that share a main

category with v. The list is sorted by increasing length.

5.1.7 Presenting the Results

Having independently found three lists of potentially relevant terms for the query v, the task is to identify a mapping, i.e. the most relevant term in T. We search the three lists to select the '*best of three*'.

This *best of three* term is selected as follows. We select the term with the highest average rank that is found by all methods. If no such term exists, we select the term with the highest average rank over two of the three methods. We leave this '*best of three*' field blank if no term is identified by more than one method.

Finally, the *set of winners* is identified, consisting of at most four terms: the best scoring terms for the individual methods plus the *best of three*.

As the algorithm presented is intended to be an assistant for users of the catalog rather than a fully automatic mapping, we also present the outputs of the individual methods. An HTML page is generated where the terms are linked with the corresponding entries in the thesaurus. Hence, even if the best suited term is not found, the user can navigate to this term by clicking a closely related term.

As the number of queries is linear in the number of patterns for a given input term, a real-time application is possible. With an (inefficient) implementation where 21 queries (1 for the categories and 10 for both the hyponyms and enumeration patterns) per term are performed sequentially, the method returns the results within a minute.

5.1.8 Experimental Results

In this section we present experiments with two thesauri. We evaluate the performance when mapping terms to the Dutch GTAA for the audiovisual archives, where in the second part of this subsection we use the United States NALT agricultural thesaurus.

To be of assistance, the relevant terms within the results of the method should be observable at a single glance. We therefore not only analyze the performance of the *best of three* term, but also the precision of the three individual methods and the average ranking of the terms in the benchmark set.

Experiments with the GTAA

We performed two experiments with the GTAA. In the first experiment, we map a set of 'expired terms' to the thesaurus, where in the second we use a 'leave-one-out' strategy to evaluate the recall and precision of the method. That is, we remove a term t from the thesaurus T and map this term to $T \setminus \{t\}$.

Mapping expired terms to the GTAA. As novel items are constantly added to the archive of B&G, the GTAA is updated from time to time as well. A major

[query] [keyword]	*[query] [keyword]*
[query] en [keyword]	*[keyword] en [query]*
[keyword] en [query]	*[query] en [keyword]*
[keyword] [query]	*[keyword] [query]*
[keyword] zoals [query]	*[query] of [keyword]*
[query] en andere [keyword]	*[query] en de [keyword]*
[keyword] als [query]	*[query] de [keyword]*
[keyword] of [query]	*[keyword] of [query]*
[query] of [keyword]	*[query] in [keyword]*
[keyword] van [query]	*[query] van [keyword]*

Table 5.3. The 10 hyponym patterns (left) and enumeration patterns (right) used for the GTAA.

problem is replacement of 'expired terms' with terms within the latest version of the GTAA.

In this experiment we discuss the applicability of our method to resolve this problem. As a benchmark set, B&G provided us a list of 78 pairs of such expired terms and the terms within the (current) GTAA to replace them.

We have automatically learned the patterns (Chapter 3) for the hyponym and enumeration relations by selecting all terms in the thesaurus starting with a – e and their BT or RT respectively. For each of the two methods, we use the 10 patterns that are found to be most effective (Table 5.3). We use `[query]` as a placeholder for the term to be queried (thus outside the thesaurus) and `[keyword]` denotes the place in the snippets where we search for terms within the thesaurus. It is notable that there is an overlap between the patterns for the two relations. The patterns *zoals* and *en andere* are Dutch translations of the patterns first identified by Hearst [1992] and translated by IJzereef in [2004].

The results for the test with the expired terms can be found in Tables 5.4 and 5.5. Table 5.4 shows the accuracy of the highest ranked terms for the three individual methods as well as the accuracy for the *best of three* and *set of winners*. It shows that the *best of three* provides the correct term – i.e. the benchmark – in 13 cases, while the highest scored terms with the hyponym, enumeration and lexical methods are correct in only 12, 4, and 7 cases respectively. For 15 terms, no *best of three* could be identified.

We have also analyzed the recall of the terms that are *1 click away* from the benchmark term by either the US, BT, NT or RT relation. Hence, these terms found have a closely related meaning. For example, given the term *bioscooppersoneel*

	WITHOUT CATEGORIES		USING CATEGORIES	
	benchmark	1 click away	benchmark	1 click away
#1 hyponyms	13	24	12	22
#1 enumerations	2	13	4	16
#1 lexical	6	11	7	15
best of three	14	21	13	24
set of winners	20	41	18	41

Table 5.4. Performance for the 78 expired terms. We compare the precision using the computed categories and subcategories (l.) with the approach where no (sub)categories were assigned to the expired terms.

(see Table 5.1), the terms *personeel*, *werknemers*, *filmoperateurs*, *bioscopen*, *film* and *explicateurs* are all linked to this term. If the method select either one of these terms, the user can navigate in one step to the term *bioscooppersoneel*. We analyze the number of times one of the *one click away* terms is among the search results. The average rank is computed using the *one click away* term with the highest rank.

If we analyze the term selected as the *best of three*, then 24 out of 78 are *one click away* from the term in the benchmark set. The *set of winners* contains such a term in 41 out of the 78 cases.

Table 5.4 shows that the performance of the lexical method improves when we take the category information into account. Contrary to our assumptions, the results of the method using hyponym patterns does not improve when using the category information. For the given benchmark set, the ranking of the enumeration patterns is slightly improved when using the category information.

Table 5.5 focusses on presence of the benchmark term within the *full* ordered lists. Depicted is the number of times the actual benchmark term is identified and their average rank. Hence, the benchmark term is identified in 46 out of the 78 cases using the hyponym patterns. However, the average rank of both the benchmark term and of the terms with distance 1 is better using the enumeration patterns. Here we see that the use of category information has a positive effect on the ranking of the method using the hyponym patterns.

When analyzing the results of the methods, we encounter numerous terms found that are intuitively correct. In Table 5.6 we give a number of examples of expired terms, the *best of three* alternative found and the benchmark as given by B&G.

For example, for the term *tabakswinkels* (tobacco shop) the term in the benchmark set, *detailhandel* (retail trade), was not found. However, the found sugges-

	WITHOUT CATEGORIES		USING CATEGORIES	
	benchmark	1 click away	benchmark	1 click away
recall hyponyms	46	70	46	70
average ranking	14.84	8.14	9.45	7.72
recall enumerations	16	44	16	44
average ranking	7.43	3.63	7.31	3.47

Table 5.5. Recall and average rank for the 78 expired terms.

term	*best of three*	benchmark
tabaksplanten tobacco plants	*planten* plants	*tabak* tobacco
tabakswinkels tobacco shops	*winkels* shops	*detailhandel* retail trade
tegelzetters tiler	*bouwvakkers* construction workers	*ambachten* crafts
titanium titanium	*metalen* metals	*chemische elementen* chemical elements
toxicologie toxicology	*geneeskunde* medical science	*vergiftigingen* poisonings
troepen troops	*militairen* soldiers	*krijgsmacht* armed forces
tweeverdieners two-earner family	*gezinnen* families	*inkomens* incomes

Table 5.6. Example terms and their English translations with their 'best of three' and benchmark mapping.

tions *kiosk*, *sigaretten* (cigarettes) and *winkels* (shops) seem valid alternatives as well. For *treinongelukken* (train accidents), we find *ongelukken* (accidents), *rampen* (disasters) and *verkeersongelukken* (traffic accidents), where the correct term was *spoorwegongelukken* (railway accidents).

To test the effect of the choice of the values for $q(t,v)$, we varied the value for the main category. We compute the ranks for the three individual methods by incrementally increasing the value for the main category, from 0.1 (the value for *sharing*

	benchmark		benchmark
#1 lexical	138	recall hyponyms:	343
#1 hyponyms	71	average ranking:	13.66
#1 enumerations	87		
best of three	159	recall enumerations:	146
set of winners	239	average ranking:	2.12

Table 5.7. Performance: recall and average rank for the 573 GTAA terms

no category) to 1 (the value for *sharing a subcategory*). For this experiments, the differences encountered were negligible. Hence, $q(t,v)$ can be simplified by only distinguishing two cases: either v and t share a subcategory or not. The use of the subcategory information is of particular use for the lexical- and enumeration-based methods.

Leave one out. In this second experiment with the GTAA, we use the thesaurus itself as a benchmark set. We proceed as follows. We select a term t within the GTAA that has a broader term b. We then remove t from the thesaurus and use this thesaurus $T \setminus t$ as a reference. The task is now to find the mapping of the term outside the thesaurus (i.e. t) to b.

We use the same patterns as in the previous experiment. For fairness, we therefore will only evaluate with terms in the thesaurus starting with the letters f to z that have a broader term. This resulted in a test with 573 terms. When a term has multiple broader terms, in the evaluations we focus on the best scoring one.

The results for these tests are given in Table 5.7. It shows that in 239 out of the 573 terms (i.e. 42%) the correct term is among the set of winners (of size at most 4). Table 5.7 shows again that the recall using the hyponym patterns is larger, but the ranking of the enumeration-based method is more precise. The lower recall using the enumerations can be explained by the structure of the thesaurus. As the GTAA is quite flat, for many terms found within the snippets no broader term is defined.

As an example, in Table 5.8 we give the best scoring output for the query term *fietspaden* (bicycle tracks). The broader term in the thesaurus is *infrastructuur* (infrastructure).

Given the difficulty of the tasks, and the fact that the mappings chosen in the benchmarks are sometimes debatable, we consider the results of the experiments convincing. As the correct answer is present in the majority of the cases (as the hyponym pattern method found 343 out of 573 correct mappings), the method

best of three:	paden	
lexical:	paden	*paths/tracks*
hyponyms:	wegen	*roads*
hyponyms:	trottoirs	*sidewalks*
hyponyms:	paden	
hyponyms:	meren	*lakes*
hyponyms:	padden	*toads*
enumeration:	infrastructuur	*infrastructure*
enumeration:	paden	
enumeration:	openbare voorzieningen	*public services*
enumeration:	wegen	
enumeration:	beroepen	*professions*

Table 5.8. Best scoring output for *fietspaden* (bicycle tracks).

can be of value as an assistant for those working with the GTAA or the catalog of B&G. The experiment with the expired term showed that the determination of the categories improves the performance of the lexical approach. The results suggest that this preliminary step can be omitted for the other two approaches.

Experiment with NALT

To illustrate that the methods used are suited for English as well, we perform the last experiment with the Agricultural Thesaurus (NALT) of the US National Agricultural Library.

The NALT contains Latin names for animals and plants, names for molecules and bacteria, but also product names such as *Brie Cheese*, *champagne* and *fish steaks*.

We learn the patterns using the terms starting with the letter *a*. The patterns found for NALT are given in Table 5.9. As the NALT does not categorize the terms, we omit the step as described in Section 5.1.3. As the NALT consists of 68581 terms, we also leave out the collection of the number of search engine hits for each thesaurus term, as this would require 14 days using the *Yahoo!* API.

We test the performance of the methods on the 3321 NALT terms starting with the letters *b* to *d* that have a broader term. The results are given in Table 5.10. As an illustration Table 5.11 gives the output for the term *dietary cation anion difference*, where *feed formulation* is its broader term in the NALT. The right mapping for the (more common term) *bitterness* indeed was found (Table 5.12).

For the hyponym-pattern based approach, typically a long list of terms is found that all co-occur once with the queried term. As no frequency information is avail-

[query] [keyword]	*[query] [keyword]*
[query] and [keyword]	*[query] and [keyword]*
[query] are [keyword]	*[keyword] and [query]*
[query] or [keyword]	*[query] or [keyword]*
[keyword] and [query]	*[query] are [keyword]*
[query] the [keyword]	*[query] such as [keyword]*
[query] and other [keyword]	*[query] of [keyword]*
[keyword] such as [query]	*[keyword] or [query]*
[keyword] including [query]	*[query] as [keyword]*
[keyword] or [query]	*[query] for [keyword]*

Table 5.9. The 10 hyponym (l.) and enumeration (r.) patterns used for NALT.

	benchmark		benchmark
#1 lexical	169	recall hyponyms:	468
#1 hyponyms	10	average ranking:	41.02
#1 enumerations	100		
best of three	192	recall enumerations:	301
set of winners	286	average ranking:	8.39

Table 5.10. Performance for the 3321 NALT terms

able, the ranking is just alphabetic. Using the enumeration method however, less terms are found. Moreover, as multiple hyponyms contribute to the score of their hypernym, the scores for the terms found tend to differ. Hence, although the number of correct mappings found with the enumeration patterns is smaller, the ranking of the method is much more reliable than the hyponym-based method.

It immediately shows that the results for NALT are far more modest. However, given the nature of the NALT and the fact that we did not correct for the frequencies of the terms, we consider the results as a proof of concept that this method is also applicable to another domain and language.

5.1.9 Conclusions

We have developed an algorithm to assist people to find alternative terms within a thesaurus for a given query term.

The algorithm developed combines three approaches to map a term to a term within a thesaurus. We use both texts found with *Yahoo!* as well as a simple lexical matching technique to make the mapping. The algorithm is constructed in-

hyponyms:	ammonia
hyponyms:	anions
hyponyms:	buffering capacity
hyponyms:	fever
hyponyms:	literature
hyponyms:	milk
hyponyms:	milk fever
hyponyms:	placenta
hyponyms:	retained placenta
hyponyms:	salts
hyponyms:	species differences
hyponyms:	urea
hyponyms:	viscosity
enumeration:	periparturient diseases and disorders
enumeration:	pregnancy complications

Table 5.11. The output for *dietary cation anion difference*.

best of three:	flavor
	...
hyponyms:	face
hyponyms:	families
hyponyms:	fear
hyponyms:	flavor
hyponyms:	food choices
hyponyms:	garlic
	...
enumeration:	flavor
enumeration:	ketones
enumeration:	thermodynamics
enumeration:	physics
enumeration:	light
enumeration:	grapes

Table 5.12. Part of the output for *bitterness*; 116 alternatives with the same score were found using the hyponym patterns.

dependently from the content of the thesaurus and can easily be mapped to another language. The combination of independent methods lead to considerably better performances than any of the individual methods.

The method can facilitate searching a catalog with the use of index terms, since

the algorithm can present a small number of alternative thesaurus terms for a given term. This reduces the number of alternatives from the 5,000 GTAA terms to only a handful. The experiments with the GTAA show that the algorithm indeed can be usable as an assistant to find the right terms within the thesaurus.

5.2 Extracting Lyrics from the Web

In the second section of this chapter, we focus on a rather different application of information extraction: the identification and construction of lyrics from the web.

We present an approach to automatically retrieve and extract lyrics of arbitrary popular songs from the web. An increasing amount of music is distributed via the Internet without the lyrics being included. Our approach offers the possibility to retrieve the lyrics of popular songs with little or no user effort allowing them to be read or sung during playback.

Lyrics are also used in karaoke-like settings. Applications that synchronize the music with its lyrics are the focus of ongoing research [Y. Wang, Kan, Nwe, Shenoy, & Yin, 2004; Iskander, Wang, Kan, & Li, 2006; Chen, Gao, Zhu, & Sun, 2006; Geleijnse et al., 2008]. These methods take both the audio-file and the lyrics as input. In this section we show that the retrieval of lyrics can be done automatically.

The lyrics of a song can also be used to extract additional information on the corresponding song. For example, if we want to create a playlist with, say 60% Christmas songs, the use of lyrics is the most obvious method to achieve this goal. Earlier work by Logan et.al. [Logan, Kositsky, & Moreno, 2004] focussed on the semantic analysis of lyrics. A set of songs was classified by genre using either features extracted from audio or from the lyrics. Their results indicate that the combination of the two could result in a better classification. Where Logan et.al. used a model where genres were associated with frequently occurring words in lyrics of the genre, other directions can be taken as well to extract information from the lyrics. The lyrics can be used to detect the topic of a song (e.g. Christmas, New York, lost love, summer holiday), its mood [Balog, Mishne, & De Rijke, 2006], the song's structure or the language in which it is sung [Mahedero, Martínez, Cano, Koppenberger, & Gouyon, 2005]. Hence such meta-data extracted from the lyrics can be used to find similar songs. Feature extracting from lyrics may thus be a valuable addition next to acoustic features extracted from audio (e.g. [McKinney & Breebaart, 2003]) and external sources such as reviews [Dave & Lawrence, 2003; Whitman & Ellis, 2004] or arbitrary web pages on music [Knees et al., 2004; Geleijnse & Korst, 2006c].

On the web several lyrics sites offer the lyrics of large collections of songs. So, users could access one of these lyrics sites. Apart from having to extract the lyrics

from the pages manually, these sites have two disadvantages. Each of the sites offers only a relatively small part of the total amount of available songs. In addition, the lyrics they offer are rather error prone. The lyrics sites usually depend on lyrics that have been uploaded by end users. Detailed analysis reveals considerable differences in the lyrics offered by the various sites for the same song.

In this section we present an approach to retrieve multiple lyrics versions of a given song using a search engine. This also offers access to the lyrics of songs that can be found on the web but do not appear at popular lyrics sites. Furthermore, by aligning the multiple lyrics versions we give direct insight into the differences in these lyrics versions.

Our approach consists of the following steps. Using only the song title and artist name, web pages are extracted that potentially contain lyrics of the song. Usually such web pages contain, in addition to the lyrics, other material, including surrounding text, advertisements, etc. From each of these web pages, the text fragment that is expected to comprise the lyrics is identified and extracted. Next, the multiple text fragments are compared to remove outliers. In order to compute a 'correct' version of a song's lyrics, the remaining text fragments are aligned on a word-by-word basis, aiming to maximize the number of matching words.

To the best of our knowledge, Knees, Schedl & Widmer [2005] are the only ones that consider web-based lyrics extraction. Our approach differs from theirs in the following aspects. Although they use a similar method to acquire web pages containing the lyrics, the subsequent steps of our approach are far more efficient. Before carrying out the actual alignment, we reduce the number of text fragments by removing outliers (i.e., text fragments that do not relate to the lyrics) by using a fingerprinting method. In addition, Knees et al. perform an approximate multiple sequence alignment of n web documents by aligning each pair of texts in a hierarchical fashion, requiring a total of $n^2 \log n$ alignments of web document pairs. In contrast, we select one reference text fragment and align each of the other text fragments with this reference text, requiring only n alignments of text fragment pairs, while retaining the same quality of solutions.

The outline of this section is as follows. First, we discuss an approach to identify a collection of *documents* that are likely to contain the intended lyrics (Section 5.2.1). In Section 5.2.2 we present an algorithm to *extract* the lyrics from these documents. The collected set of lyrics is likely to contain other texts than solely the lyrics of the intended song. We present an efficient method to *remove outliers* in Section 5.2.3. Having identified a set of lyrics that are expected to reflect the intended song, we present a multiple sequence alignment algorithm to *construct* a single version of the lyrics in Section 5.2.4. In Section 5.2.5 we discuss the computational complexity of the proposed algorithms and make a detailed comparison with the algorithm of Knees et al. We present experimental results in Section 5.2.6

`allinanchor`: *"[Artist]"*, *"[Title]"*, *lyrics*
`allintitle`: *"[Artist]"*, *"[Title]"*, *lyrics*
`allinanchor`: *"[Title]"*, *lyrics*
`allintitle`: *"[Title]"*, *lyrics*
"[Title]", *lyrics*

Table 5.13. The query templates used to retrieve pages containing relevant lyrics.

and end with concluding remarks in Section 5.2.7.

5.2.1 Website-independent Lyrics Extraction

In this section we describe how we retrieve and select a number of texts that each potentially constitute the lyrics of a given song. We assume that we are only given the title of the song and the name of the performing artist. The lyrics extraction algorithm consists of two steps. First we collect URLs of documents that potentially contain the lyrics of the song of interest (Section 5.2.1). Since we are interested in the lyrics themselves – and not in the documents as a whole – we then extract the lyrics from these documents using a website independent approach (Section 5.2.2).

Collecting URLs of Documents

We first retrieve the URLs of documents that potentially contain the lyrics of the intended song. We focus on heterogeneous sources and opt for an approach that is website independent.

To obtain potentially relevant documents we use a list of queries with placeholders. For our experiments we heuristically constructed the list of query templates in Table 5.13. The expressions `[Artist]` and `[Title]` are placeholders for the artist name and song title.

The first query returns web pages that contain the exact strings of the song title and artist name (by putting them between quotes) and the word *lyrics* in the anchor text surrounding the links to these pages (by adding the `allinanchor` clause). The anchor text is the text that directly relates to a link and can be considered as a condensed summary of the content of the page.

As an alternative to the `allinanchor` option we also use the `allintext` option. For this application, the allinanchor-option usually offers more hits than the allintitle-option, where the allintitle-option requires that the terms in the query appear in the title of the web page. The allinanchor-option is especially worthwhile for rare songs, for which the allintitle-option may give no or only a small number

of results. For the more well-known songs, the top-ranked results for both options seem to be of equal quality.

We store the URLs of the search results of the first query in a list $\mathcal{L}$ in the order as they are presented by the search engine. The results of the following queries are added to the tail of $\mathcal{L}$, where we omit double URLs.

5.2.2 Extracting Lyrics from the Documents

After retrieving the list of relevant URLs, we want to extract the lyrics from the corresponding documents.

We make use of the textual structure of lyrics. Within a book, we can observe at a single glance whether the text is prose or poetry. Like poetry, lyrics typically consist of stanzas separated by blank lines. Each stanza consists of one or more lines, where each line has a maximum number of characters. We opt for a rule-based approach to recognize the lyrics in text (Chapter 3.2.2).

In hypertext, the end of a line is marked with `<br>`-tag. Each line within the lyrics will thus end with such a tag. We assume the markup within the lyrics to be constant. Hence, the only tags within the lyrics part of the HTML source file of the page will be the end-of-line markers.

We use these characteristics to identify lyrics in a hypertext as follows. Let D be a document containing r end-of-line markers. Then, D can be considered as a sequence $(t_0, t_1, \ldots, t_r)$ with $r+1$ fragments, where the ith end-of-line marker separates t_{i-1} and t_i.

Since we opt for a time-efficient algorithm, we scan the full document only once. We therefore map the string of $r+1$ fragments t_i, $0 \leq i \leq r$, to a string L of the form $(\boldsymbol{b}|\boldsymbol{l}|\boldsymbol{n})^{r+1}$, i.e. a string of length $r+1$ containing only $\boldsymbol{b}$s, $\boldsymbol{l}$s and $\boldsymbol{n}$s. The mapping m of the fragments t_i meets the following criteria.

- $m(t_i) = \boldsymbol{b}$ whenever t_i is empty or only consists of white space characters (blank line),
- $m(t_i) = \boldsymbol{l}$ (a lyrics line) whenever t_i
  (a) does not contain any HTML-tags,
  (b) contains between 3 and 80 characters and
  (c) half of the characters are letters,

  and
- $m(t_i) = \boldsymbol{n}$ otherwise (non lyrics).

In terms of $\boldsymbol{b}$s, $\boldsymbol{l}$s and $\boldsymbol{n}$s, lyrics can now be described by a regular expression. As we provided the algorithm with a list of query templates, we also input a list of regular expressions that describe lyrics (Table 5.14).

$R_0 = (\boldsymbol{l}^{1-20} \cdot \boldsymbol{b})^{1-12} \cdot \boldsymbol{l}^{1-20}$
$R_1 = \boldsymbol{l}^{3-40}$

Table 5.14. Regular expressions R_0 and R_1 expressing lyrics within a page.

Here, $\boldsymbol{a}^{i-j}$ denotes a sequence of $\boldsymbol{a}$s of a length l, $i \leq l \leq j$. The second expression accepts a wider variety of texts. This is useful where there no stanzas are indicated, e.g. in lyrics of rap songs.

We match the string L to a regular expression as follows. As we want to extract the full lyrics (and for example not omit the last stanza), we want to identify the first longest substring that is described by the regular expression.

We match the string L with the regular expression $R' = R_0 \cdot (\boldsymbol{l} + \boldsymbol{b} + \boldsymbol{n})^*$ to find a prefix of L that matches R_0. If the first character of L is either a $\boldsymbol{b}$ or a $\boldsymbol{l}$ this is a $\mathcal{O}(1)$ operation. If not, the matching has an upperbound of the maximum number of lines in the lyrics as described by R_0, hence $21 \cdot 12 + 20 = 273$, or the length of L if $|L| \leq 273$. If no such match is found we remove the first character of L and repeat the procedure until either a match is found or L is the empty string. The search for a substring matching R' thus has a time complexity of $O(|L|^2)$.

After having found a substring of L that matches R', we want to find the longest prefix that matches R_0. We use a bounded linear search to determine the length of the longest prefix that matches R_0. We remove the last character from the string until the full string matches R_0. Hence, the identification of the longest substring matching a regular expression has a total time complexity of $O(|L|^2)$.

Some web sites contain the lyrics within preformatted text (put between tags `<pre>` and `</pre>`). As a preprocessing step, we first extract such fragments from the document. The preformatted text is then mapped to a sequence L with the same rules (using the new line character instead of the `<br>`-tag).

We match the sequence L to the regular expressions R_0 and R_1 and extract the fragment as described above. We use the corresponding substring in the document to obtain the lyrics within it. Finally, we check the text that directly precedes and succeeds the resulting substring, since the first and last lines of the lyrics may not be included in this substring.

Again for efficiency reasons, not all documents in the list $\mathcal{L}$ are downloaded. From the list of URLs, the top element is taken and the corresponding document is retrieved. The above extraction algorithm is applied to the retrieved document. The texts retrieved with either one of the regular expressions are added to a set $\mathcal{P}$ of potential lyrics. We continue this process until lyrics have been extracted from 40 documents, or the list $\mathcal{L}$ is empty.

5.2.3 Selecting a Subset of Lyrics

After having collected a number of URLs of documents and identified a number of potential lyrics (the set $\mathcal{P}$) from the documents collected, the task remains to remove the texts that are not the lyrics of the intended song s.

The set $\mathcal{P}$ of text fragments that is extracted as described in the previous section is likely to also contain text fragments that do no relate to the lyrics of the intended song, for a number of reasons, such as the following ones.

- A text fragment can be the lyrics of another song by the same artist (especially if the title of the intended song is a subsequence of the title of the other song).
- A text fragment can be the listing of an album's songs in which the intended song appears (especially if the song title is identical to the album title).
- A text fragment can be a listing of a playlist.

In this stage we want to remove these so-called outliers, since they do not reflect the intended lyrics. We use the assumption that the majority of the extracted text fragments constitutes the lyrics of the intended song.

We cluster the text fragments on the basis of similarity and retain only the text fragments in the largest cluster. As variations frequently occur in representations of lyrics to the same song, exact string matching is unsuited for this purpose. Approximate string matching techniques of strings of lengths s_0 and s_1 are in general $O(s_0 \cdot s_1)$. Since in worst case we need to compare each pair in $\mathcal{P}$, we consider this as computationally too expensive.

Creating a Fingerprint for the Lyrics

Instead of comparing the full strings in the set $\mathcal{P}$, we compare the *fingerprint* of the strings.

We use the assumption that longer words occur less frequently in texts than shorter ones [Manning & Schütze, 1999; Sigurd, Eeg-Olofsson, & Van Weijer, 2004]. We hence define the fingerprint $f(t) = (w_{t1}, w_{t2}, \ldots, w_{tm})$ as the m longest – and most typical – words for a text t.

For these purposes we define the order $\prec$ on words as follows.

$w_0 \prec w_1$ if and only if

- The length of w_0 is smaller than the length of w_1, or
- w_0 and w_1 have equal length and w_0 is lexicographical smaller than w_1.

The last criterion assures the selection of fingerprints from the texts to be consistent.

The Beatles - *Penny Lane*	photographs roundabout meanwhile hourglass suburban
Bob Dylan - *A Hard Rain's Gonna Fall*	executioner's whisperin mountains graveyard forgotten
O-Zone - *Dragostea Din Tei*	fericirea dragostea primeste amintesc Picasso
The Police - *Roxanne*	wouldn't tonight streets Roxanne another
Prince - *Purple Rain*	underneath friendship something laughing changing
Procol Harum - *A Whiter Shade of Pale*	straightway cartwheels cardboard wandered straight
Queen - *Bohemian Rhapsody*	Thunderbolt Scaramouche monstrosity frightening silhouetto
Britney Spears - *Baby One More Time*	loneliness something shouldn't wouldn't supposed
Emily Brontë - *Wuthering Heights*	intercommunication mispronunciations incomprehensible unsatisfactorily unconsciousness
Charles Dickens - *Oliver Twist*	pockethandkerchief chimbleysweeper's stauncherhearted unconstitutional Northamptonshire
Euclid - *The Elements (book 1)*	parallelogrammic quadrilaterals Parallelograms quadrilateral perpendicular
E.W. Dijkstra - *Goto statement considered harmful*	superfluousness undesirability specifications recommendation correspondence

Table 5.15. The fingerprints, with $f = 5$, of a number of texts.

As a fingerprint for t we thus select the m longest words using ordering $\prec$. The fingerprint of t can be computed in time linear in the length of t. Table 5.15 gives examples of the fingerprints for a number of lyrics and other texts.

Comparing Lyrics using Fingerprints

Having computed the fingerprints $f(t)$ for each text fragment t, we use them to select a subset $\mathcal{P}'$ of $\mathcal{P}$ such that each t in $\mathcal{P}'$ is a representation of the lyrics of the song queried. As an example, Figure 5.1 gives the fingerprints of 12 texts gathered for the song *Silver and Gold* by *U2*. The fingerprint of text 8 contains indeed terms that occur in the lyrics of the song. However, as the fingerprint words are shorter than the ones of e.g. text 1, text 8 will be an incomplete version of the lyrics to this song. Texts 1 and 6 share three out of five fingerprint words, where text 6 will not contain the word *silence*. The fingerprint for text 11 reflects a different song as it

```
 1 temperature something prisoners daylight silence
 2 temperature something prisoners daylight silence
 3 temperature something prisoners daylight silence
 4 satisfied important seriously caution foolish
 5 temperature something prisoners daylight silence
 6 temperatur something prisoners daylight someone
 7 temperature something prisoners daylight silence
 8 shotgun praying hunter hunted house
 9 rivers lights silver these shine
10 temperature something prisoners daylight silence
11 punctures waterless disappear carnival tonight
12 temperature something prisoners daylight silence
```

Figure 5.1. The fingerprints of 12 lyrics versions of the song *Silver and Gold* by *U2*.

shares no words with any other text.

To compute the subset $\mathcal{P}'$ of $\mathcal{P}$, we pairwise compare the fingerprints of the text fragments. We assume that if the fingerprints of two text fragments share at least k words, then they relate to the same text. We note that different lyrics versions of a song will have similar fingerprints, irrespective of whether repeating parts are included explicitly or not. For our experiments, we chose $k = 3$, while $m = 5$.

We construct a graph, where each node in the graph corresponds to an extracted text fragment. There is an edge between two nodes if the fingerprints of the corresponding text fragments share at least k words. Now, the connected components of the graph determine the clusters of the extracted text fragments. Two extracted text fragments are in the same cluster if there is a path in the graph from one to the other. We only retain the text fragments in the largest cluster.

Since a fingerprint consists of an ordered list of m words, we can determine whether two fingerprints share at least k words in at most $2m$ word comparisons. Hence, constructing the graph and determining the largest connected component requires $O(n^2m)$ word comparisons.

Occasionally, the extracted text fragments are so diverse that there is no clear winning subset. In that case the previous step is redone with the alternative regular expression, or, if that does not work, the document retrieval step is redone using a broader query.

5.2.4 Aligning Multiple Lyrics

If the lyrics retrieval and selection have successfully completed, then we have extracted a number of similar text fragments, that are all expected to be lyrics versions of the intended song. Even if this is the case, then there may still be a large variety

```
that life's a bore so full of the superficial
that life support of all of the superficial

A thousand lights had made me colder
A thousand lies have made me colder

Hear him with the women just around midnight
Hear him whip the women just around midnight

Burned out dealer to the teachers pet
Burnouts deal it to the teacher's pet

Cause waiting at the answer to his questions is a definite blow
Persuade him that the answer to his questions is a definite no!
```

Figure 5.2. A list of pairs of transcriptions. Each pair gives two different transcriptions of the same part of a song that we encountered in the retrieved lyrics.

in the extracted text fragments. Varieties occur as a result of mishearings, typo's and the use of abbreviations such as *repeat chorus*.

In Figure 5.2 we give a number of examples of transcriptions we encountered.

We next want to align the extracted text fragments to easily visualize the differences and to come up with a most probable version of the lyrics. This version is constructed using the lyrics identified on the web. The final version thus does not need to occur as such on the web.

Aligning multiple sequences is known to be an NP-hard problem [L. Wang & Jiang, 2004], for many sensible choices of the objective function such as the sum-of-pairs objective function. For a given alignment of n sequences the sum-of-pairs objective function simply sums up the score of all sequence pairs in the alignment. Several approximation algorithms have been proposed in the literature, e.g. [L. Wang & Gusfield, 1997; L. Wang, Jiang, & Lawler, 1996].

We choose the following approach. We first select a reference sequence and optimally align each of the other sequences with this reference sequence. Next we combine all these individual alignments into a single alignment of all sequences. As reference sequence we simply choose a sequence of maximum length, as we expect this sequence to give a complete transcription of the intended song. Shorter sequences may not include repeating parts explicitly or miss the beginning or end of the song.

Aligning a Pair of Lyrics

We align a pair of lyrics on the word level. To realize this, we opt for a dynamic programming approach where we align a pair of strings S_1 and S_2 in a $2 \times l$ matrix $A = [a_{ij}]$ of words. For A we have $\max(l_1, l_2) \leq l \leq l_1 + l_2$. Of this matrix, the ith

row is denoted by $A_{i\circ}$ and the jth column is denoted by $A_{\circ j}$. The upper row $A_{1\circ}$ consists of the source string S_1 with possibly empty strings inserted and the lower row $A_{2\circ}$ consists of the destination string S_2 with possibly empty strings inserted. An empty string or gap in an alignment is denoted by an asterisk ($\star$), assuming that this character does not appear as a word in the lyrics.

Each column $A_{\circ j}$ in the alignment corresponds to either a deletion, an insertion, a substitution, or a match. Gaps in the upper row $A_{1\circ}$ correspond to insertions in S_1, and gaps in the lower row $A_{2\circ}$ correspond to deletions in S_1. There is at most one gap in each column. If there are no gaps in a given column, then this position either corresponds to a match (if both characters in that column are identical) or a substitution (otherwise).

As a primary objective, the goal is to maximize the number of matches, and, as a secondary objective, to minimize the number of mismatches. To realize this, we use the following recurrence relation. We compute the dynamic programming table to align S_1 and S_2 using the following recurrence relation.

$$D(i,j) = \begin{cases} -j & \text{if } i = 0 \\ -i & \text{if } j = 0 \\ \max\ \{\ D(i-1,j)-1, D(i,j-1)-1, & \\ \quad D(i-1,j-1)+t(i,j)\ \} & \text{otherwise} \end{cases} \tag{5.5}$$

with

$$t(i,j) = \begin{cases} M & \text{if } S_1(i) = S_2(j) \\ -1 & \text{if } S_1(i) \neq S_2(j), \end{cases}$$

where $S_i(j)$ now denotes the jth word in the ith text fragment. Hence, all insertions, deletions and substitutions receive a weight -1, while a match receives a weight M. Now, M must be chosen large enough such that each alignment with a maximum score has a maximum number of matches. In a worst-case situation, an alignment with a single match in the lower-left or upper-right corner of the dynamic programming table must still get a larger score than an alignment with no matches and $\max(l_1,l_2)$ mismatches. Consequently, it must hold that $M-(l_1-1)-(l_2-1) > -\max(l_1,l_2)$, which is equivalent to $M > \min(l_1,l_2)-2$. Hence, choosing $M = \min(l_1,l_2)$ gives the desired result.

Multiple alignments with maximum $D(l_1,l_2)$ may occur. To get the matches as much as possible in the first part of the alignment, we give preference to insertions and deletions when tracing back.

Combining Single Alignments

We use the above dynamic programming approach to align the different text fragments on a word-by-word basis. Suppose that the remaining set of text fragments is

given by $\mathcal{P}' = (S_0, S_1, \ldots, S_m)$, and suppose that these word-sequences are ordered by decreasing length, i.e., $|S_i| \geq |S_{i+1}|$ for all $i = 0, 1, \cdots, m-1$. Then, we select S_0 as reference sequence and construct an optimal alignment between the reference sequence and each of the other sequences, resulting in alignments $A^1, A^2, \ldots, A^m$.

The resulting m alignments $A^1, A^2, \ldots, A^m$ are next combined into a single multiple alignment $A = [a_{ij}]$ containing $m+1$ rows. We could simply take $A_{1\circ} = A^1_{1\circ}$, and $A_{i+1\circ} = A^i_{2\circ}$ for $i = \{1, \ldots, m\}$. However, the upper rows in the different alignments (i.e. the rows corresponding to S_0) will generally have a different number of gaps, and these gaps will generally be at different positions. Let us consider aligning the following three character sequences as an example: 'california dreaming' and 'californian dream' and 'calif. dreaming', then the first will be chosen as reference sequence, resulting in the alignments A^1 and A^2 given as follows.

```
california* dreaming          california dreaming
californian dream***          calif.**** dreaming
```

Just taking the rows, as suggested above, from these two alignments would result in an alignment A given as follows.

```
california* dreaming

californian dream***

calif.***** dreaming
```

To obtain a better alignment, a gap can be inserted right after the tenth character in the third row, i.e., at the position where there is a gap in the upper row of A^1.

We propose the following simple strategy to account for differences in gaps in the upper rows of the given alignments $A^1, A^2, \ldots, A^m$. We simultaneously scan the upper rows of the given alignments from beginning to end. If at a given position, there is a gap in any of these upper rows, then for each alignment A^i that does not have a gap at this position, we insert a gap at that position both in the upper and lower rows of A^i.

A small example of the multiple sequence alignments that we obtain in this fashion is given in Figure 5.3. For easy visual inspection the words in the same column are appended with spaces such that they are all of equal length. The character | denotes a line break, which is considered as a single word. In addition, an extra row has been added to underline the columns where there is disagreement.

```
cause I know the dreams that you keep |  that's  where we meet
Cause I know the dreams that you keep |  That's  where we meet
Cause I know the dreams that you keep |  That's  where we meet
Cos   i know the dreams that you keep is wearing me    *  *
Cos   i know the dreams that you keep is wearing me    *  *
Cos   i know the dreams that you keep is wearing me    *  *
Cos   i know the dreams that you keep is wearing me    *  *
Cos   I know the dreams that you keep is wearing me    *  *
Cos   I know the dreams that you keep is wearing me    *  *
cause I know the dreams that you keep |  that's  where we meet
----- -                                -- ------- ----- -- ----
```

Figure 5.3. A fragment of a multiple sequence alignment of 10 lyrics versions of the song *No distance left to run* by *Blur*.

5.2.5 Computational Complexity

In this section we analyze the computational complexity of the lyrics retrieval and alignment algorithms presented in the previous sections, and we give a detailed comparison with the computational complexity of the algorithms presented by Knees et al. [2005]. In this analysis we do not take into account the time required for issuing queries to Google and for retrieving the documents from the web. The time required for these activities are assumed to be identical for both approaches.

Algorithm by Knees et al. The document collection approach of Knees et al. is very similar to our approach. However, Knees et al. do not extract the lyrics from the documents before aligning. They only remove HTML-tags and corresponding links and preprocess the documents to handle annotations. For example, they replace annotations such as '*repeat chorus*' by the actual chorus. Knees et al. also do not consider removing outliers before aligning. In conclusion, if both algorithms start with the same collection of documents, but our approach will retain fewer and shorter text fragments for the actual alignment.

Let us assume, nevertheless, that both multiple alignment algorithms have the same number of text fragments as input, i.e., that there are no outliers in the set of n documents. For ease of analysis, let us assume that we have n text fragments each with a length of l words, and let us assume n to be a power of two. Furthermore, let us assume that all alignments remain of $O(l)$ length.

The multiple sequence alignment proposed by Knees et al. follows an iterative hierarchical approach [Corpet, 1988]. For ease of reference, an alignment consisting of n rows is called an n-alignment. In the first iteration the algorithm pairs the n text fragments to $n/2$ 2-alignments. It does so by repeatedly selecting the best aligning pair of text fragments, until it obtains $n/2$ 2-alignments. To determine the

best aligning pair, an $l \times l$ dynamic programming table is constructed for each pair of text fragments. This results in a total number of word comparisons of $O(n^2 l^2)$ in the first iteration.

In the second iteration, it pairs the $n/2$ 2-alignments to $n/4$ 4-alignments, again by repeatedly selecting the best aligning pair. To determine the best aligning pair, a dynamic programming table is constructed for each pair of the set of $n/2$ 2-alignments, where each entry in the table requires 4 word comparisons. This results in a total number of word comparisons of $O(n^2 l^2)$ in the second iteration.

Analogously, in the ith iteration the algorithm pairs the $n/2^{i-1}$ 2^{i-1}-alignments to $n/2^i$ 2^i-alignments, where each entry in the dynamic programming tables requires $2^{2(i-1)}$ word comparisons. Again, this results in a total of $O(n^2 l^2)$ word comparisons in the ith iteration.

Since the total alignment procedure will consist of $\log n$ iterations, we obtain a total number of $O(n^2 \log n\ l^2)$ word comparisons.

Our algorithm. In comparison, our algorithm first extracts text fragments from the n documents by using the regular expression in $O(nl)$ time, where l denotes the maximum number of words in a collected document. Next, it selects a subset of text fragments using the fingerprints in $O(n\,lm) + O(n^2 m)$ time, where m is the number of words that are used in a fingerprint. Constructing a fingerprint for each of the n documents requires $O(n\,lm)$ time. Constructing the graph and determining the largest connected component requires $O(n^2 m)$ time. Next, a multiple sequence alignment is constructed, by only aligning each of the text fragments to a single reference text fragment. Assuming that there are no outliers in the collected documents, this requires $O(nl^2)$ word comparisons. In addition, the insertion of additional gaps to construct the final multiple sequence alignment requires $O(nl)$ time.

Hence, the total time complexity of our algorithm amounts to $O(n\,lm + n^2 m + nl^2)$. Since $m < l$, we can simplify this total time complexity to $O(n^2 m + nl^2)$. Furthermore, in practice one may assume that $nm < l^2$. In that case, the total time complexity can be further simplified to $O(nl^2)$.

As both multiple sequence alignment approaches give approximate results, it remains to be evaluated whether our algorithm is able to obtain results of at least the same quality. This is the subject of the next section.

5.2.6 Experimental Results

We tested the algorithms on two test sets. The first set was also used by Knees, Schedl and Widmer in their lyrics extraction method [2005]. We tested the performance using the lyrics of the songs as found in the CD booklets.

We conducted a second experiment on a set of 608 songs. We compare the

retrieval of our algorithm with the retrieval of the *Yahoo!* lyrics service, which uses the *GraceNote* lyrics collection.

The 258 Song Test Set

Our lyrics extraction and alignment algorithms were tested on the set of 258 songs from Knees, Schedl & Widmer [2005], of which we obtained the ground truth versions from these authors. The ground truth versions are the versions as they exactly appear in the CD booklets. We next give experimental results for the successive steps in the algorithm.

Collecting documents. To give the reader an idea of the number of documents that are expected to contain the lyrics of the various songs, for the 258 songs Google reported an average of 507 hits for the first query (containing the allinanchor-option). However, using this first query, for 6 songs no hits were found.

Extracting lyrics. By extracting the lyrics from the documents, we get a substantial reduction. On average, the size of the extracted lyrics is only 7% of the original document size. However, the reduction is rather modest in comparison to the size of the documents after they have been stripped from HTML-tags and corresponding links. On average, the size of the extracted lyrics is 79% of the stripped document size.

Removing outliers. On average, 38% of the extracted text fragments were found to be outliers. Comparing the size of the largest cluster with the size of the second largest cluster, we obtain that on average the first is four times as large as the second one. Hence, on average, there is a clear winner among the clusters.

Multiple sequence alignment. For the 258 songs we derived the following results. To compare the results of the multiple sequence alignment with that of the ground truth, we transform the multiple sequence alignment into a final version, by applying simple majority voting on a word-by-word level. For each column in the m-alignment, we select the word that occurs most often, where a gap is handled as follows. When for a given column the different words are being counted, then a gap (a_{ij} = '*') is counted as a word, unless it is succeeded or preceded in its row $A_{i\circ}$ by only gaps. Otherwise, it is just skipped. If the most-occurring word is not a gap, then this word is added to the final version. If the most-occurring word is a gap, then this is just skipped for the final version.

To determine the similarity between the resulting final version and the ground truth version we simply construct an optimal 2-alignment of these versions. Each column of this 2-alignment can be associated with one of the following four cases, namely a match, a substitution, a gap in the final version, or a gap in the ground truth. The fraction of columns relating to these four cases are denotes by r_{ma}, r_{su},

r_{gf}, and r_{gg}, respectively. Clearly, these fractions are between 0 and 1 and sum up to 1. Analogously to Knees et al., we define the recall as

$$rec = r_{ma} + r_{gf} \tag{5.6}$$

and precision as

$$pre = r_{ma} + r_{gg}. \tag{5.7}$$

For the total set of 258 songs our algorithm obtains an average recall of 0.93 and an average precision of 0.86. This result is very similar to the best results obtained by Knees et al. Occasionally, recall is considerably lower than the average recall due to the fact that in the ground truth version a chorus is repeated explicitly while it is not in the extracted final version. Likewise, for some songs, the precision is considerably lower than the average precision due to the fact that in the final version a chorus is repeated explicitly while it is not in the ground truth version. Since these differences cannot really be considered as errors, we also determined the average value of r_{su}. This is only 0.02. In other words, in the alignments of the extracted version with the ground truth version, only 2 out of 100 words correspond to a substitution. We note that these substitutions still contain many pairs as *(movin', moving)*, *(yeah, yea)*, *('re, are)* that cannot really be considered as wrong.

The above results are averaged over all 258 songs. However, for 7 of the 258 songs the recall and precision are substantially below the above averages because the algorithm selected the lyrics of another song. In all seven cases, the selected song was by the same group or artist. In addition, in four of these seven cases the song title of the intended song appears in the lyrics or even in the title of the extracted song. For example, when searching for the lyrics of *A Long Way From Home* by The Kinks the lyrics of *Long Distance* was found. This lyrics contains the string 'a long way from home'. For three of the seven cases, the intended song was found as a second largest cluster. For the four other cases, the clustering resulted in many small clusters, with an average fraction of outliers of 0.70, which could be used as an indication that something is wrong. Furthermore, when extracting the lyrics of all songs of a given artist, it can be easily checked whether the extracted lyrics for different songs incidentally are (very) similar. Hence, these errors can be detected automatically.

Comparison with the Yahoo! Music Collection

Since April 2007, *Yahoo! Music* provides access to song lyrics for "hundreds of thousands of songs", being "the largest database of high quality lyrics"[3,4] To

[3] http://www.gracenote.com/corporate/press/article.html/date=2007042400

[4] It is notable that the (growing) content of *Yahoo! Music* is restricted to material where the copyright is granted. The experiment with *Yahoo! Music* was conducted on August 3, 2007.

Cathy Dennis	-	*Touch Me (All Night Long)*
Floyd Cramer	-	*On the Rebound*
Frank Mills	-	*Music Box Dancer*
Groove Theory	-	*Tell Me*
Horst Jankowski	-	*A Walk in The Black Forest*
Inner Circle	-	*Bad Boys*

Table 5.16. Examples of songs that were not retrieved by the algorithm.

investigate whether the algorithm we presented is now obsolete, we compare the results of our algorithm with the *Yahoo! Music* lyrics collection.

An external company handed us a set of 609 song titles. We test our algorithm on this collection and compare the results of our method with the content of the *Yahoo! Music* lyrics database. The set mainly contains well-known artists and songs from various genres.

For each song, we query *Yahoo! Music* at most three times. Contrary to the experiment with the 258 songs, this collection contains a number of song titles and artist names containing parentheses. If the query `lyrics, [songtitle], [artist]` fails, we remove the texts between parentheses in both the song title and the artist name (e.g. *Blowin' Me Up (With Her Love)* is now queried as *Blowin' Me Up*). If no results are found after the adaptation, we leave out the artist name in the query.

As no ground truth is available for this set, we manually inspect the output of the algorithm. We consider the retrieved lyrics to be correct if we recognize them as the lyrics corresponding to the queried song.

The results of this experiment are as follows. Using the algorithm as described in this article, we find lyrics of 577 songs. Of only 32 songs we did not find any lyrics, or the lyrics retrieved did not correspond to the song, see Table 5.16 for some examples.

When we query the *Yahoo! Music* lyrics collection, we only find 191 correct lyrics for the given song-artist combinations. Additionally, 12 lyrics of different versions of the queried song were found. For example, for *Strangers in the Night* by *Frank Sinatra*, the version of the song by *Bette Midler* was finally retrieved using the third query. For 108 songs, lyrics to a different song were found. Hence, no results were found for 301 out of the 609 songs in the collection.

Of the songs that were not found by the algorithm presented, three could be found in *Yahoo! Music* (viz. Table 5.18). Table 5.17 contains some example songs that were not found in *Yahoo! Music*, but were indeed retrieved using our algorithm.

Beck	-	*Loser*
Crosby, Stills, Nash & Young	-	*Woodstock*
Aerosmith	-	*Angel*
Anita Baker	-	*Giving You the Best That I Got*
Bob Marley & The Wailers	-	*Who Is Mr. Brown*

Table 5.17. Examples of songs that were retrieved by the algorithm, but were not found in Yahoo! Music.

Inner Circle	-	*Bad Boys*
Chitty Chitty Bang Bang Original Cast	-	*Chitty Takes Flight (Finale to Act One)*
Groove Theory	-	*Tell Me*

Table 5.18. The three songs that were found in Yahoo! Music, but could not be retrieved using the algorithm.

Although the lyrics provided by *Yahoo! Music* may be of high quality, this experiment shows that some well known songs are not included. As no complete and reliable web site is available for collecting lyrics, the algorithm described remains a valuable tool for music research.

5.2.7 Concluding remarks

We have presented an approach to retrieve lyrics versions from the web using a search engine, and to efficiently align them. In comparison to the approach by Knees et al., our approach is much more efficient but nevertheless gives comparable results. A second experiment illustrated that the algorithm is also able to find lyrics of songs that are not stored on the large lyrics *Yahoo! Music*. The algorithm as presented in this article can be a valuable tool for those researching lyrics-based music information retrieval [Kleedorfer, 2008; Mahedero et al., 2005]. Moreover, the lyrics found can be used as a basis for automatic lyrics synchronization [Chen et al., 2006; Y. Wang et al., 2004] and creating visual effects using images and colored lights [Sekulovski et al., 2008; Geleijnse et al., 2008].

6

Discovering Information by Extracting Community Data

Apart from factual information, the web also is a valuable source to gather community-based data as people with numerous backgrounds, interests and ideas contribute to the content of the web. Hence the web is also a valuable source to extract opinions, characterizations and perceived relatedness between items.

We extract and combine information from diverse sources on the web to characterize items such as *Madonna* and *The Great Gatsby* using community-based data. By combining information from various sources such as fan pages, newspaper reviews, gossip magazines and music websites, the aim is to create a characterization of for example *Madonna* as expressed on the web. By combining this data, we create new information that may not be verifiable as it is not available as such.

This chapter is organized as follows. In Section 6.1, we discuss the problem definition and two alternative methods to extract information from the web. In Section 6.2 we present a method to process extracted data into characterizations. Section 6.3 focuses on the evaluation of the extracted community-based data using data from a social website, while in Section 6.4 we present a number of case studies followed by conclusions in Section 6.5.

6.1 Extracting Subjective Information from the Web

In this chapter, we are interested in the characterization of an item or concept by the web community. For example, given the latest novel by *Philip Roth* or *Madonna*'s new single, we want to know the way people describe such items. Moreover given a book or an artist, which other books or artists are considered to be related?

Users of so-called *social websites* – or *folksonomies* – such as *Flickr.com*, *YouTube.com* and *Last.fm* are invited to label the items described on these sites. Unlike the thesauri studied in Chapter 5.1, the tags applied to the items have no formally defined semantics, while the vocabulary is uncontrolled. However, in practice tagging has shown to be an effective mechanism to describe and retrieve content.

For a collection of items to be well searchable, a large and active community is required who *explicitly* labels the items with tags. Items that are not labeled or labeled with less intuitive tags may thus not be retrievable. However, such community websites describe items that are often also described on many other web pages. Users are thus invited to enter knowledge that is potentially already available on the web.

Current ontology population methods based on texts on the web (e.g. [Etzioni et al., 2005; McDowell & Cafarella, 2006]) focus on factual information rather than on more subjective, community-based descriptors of items. Here we present methods to efficiently identify and structure information as can be found on social websites. We focus on the labeling of items, such as musical artists, with tags from unstructured web sources. Hence, we propose a method where the tagging of artists is done *implicitly* by the web community. We thus compute the semantics of an item (e.g. a musical artist) in terms of tags as perceived by the web community. Previous methods (e.g. [Mika, 2007; Cilibrasi & Vitanyi, 2007; Schedl, Pohle, Knees, & Widmer, 2006]) use a quadratic number of queries to a search engine. In this chapter, we compare efficient techniques as discussed in chapter 2 with such approaches.

A method to automatically label items with tags can on the one hand be used as an alternative to the labeling of items by a community. On the other hand the computed tags can be used in support of a community web site. For example, computed tags can be presented as a suggestion to the user or can be used to avoid a 'cold start' problem for items in a collection that have not been labeled yet.

6.1.1 Relatedness, Categories and Tags

To describe instances using the collective knowledge of the web community, we thus adopt the notion of tags. In this chapter, we assume that a collection of instances (books, popular artists, painters, etc.) is given as well as a list of relevant

descriptors or tags. This leads to the ontology population problem with complete classes (Chapter 2).

Problem Definitions. Given is an ontology O with two *complete* classes c_a and c_g, with sets of instances I_a of size n and I_g of size m. We will again use the shorthand notation I_a and I_g to refer to these sets. The class c_a consists of objects or concepts i (pop artists, books, painters), while the instances j of c_g are relevant descriptions (labels, tags, genres). We are interested in the perceived relatedness between the instances in I_a, as well as the applicability of the tags in I_g to these instances.

In this chapter, we focus on the following three problems.

The Instance Relatedness Problem. Given $O = (\{c_a\}, \{r\})$, where r expresses the *is related to* relation with c_a as subject and object class. Populate O, where for each relation instance pair $(i, i') \in I_a \times I_a$ we are interested in the *degree of relatedness* $t(i, i')$ of instance i' with respect to i. □

Definition. Given is a set of tags I_g and an instance $i \in I_a$. We then call a tag $j \in I_g$ *most appropriate* for i if a domain expert would select j from the set I_g as the label best applicable to $i \in I_a$. □

The Instance Categorization Problem. Given $O = (\{c_a, c_g\}, \{r\})$, where r expresses the applicability relation with subject c_a and object c_g. Populate O, where for each instance $i \in I_a$ we are interested in the most appropriate tag $m(i)$ in I_g. □

The Instance Tagging Problem. Given $O = (\{c_a, c_g\}, \{r\})$, where r expresses the relation between the two classes. Populate O, where for each relation instance pair $(i, j) \in I_a \times I_g$ we are interested in the *degree of applicability* $p(i, j)$ of tag j with respect to instance i. □

We first present two alternative approaches to extract relation instances on the web. In the next section, we present methods to address each of the three problems described above.

6.1.2 Two Alternatives to the Pattern-based Approach

We present two methods to extract information from the web, alternative to the one described in Chapters 2 and 3. In Chapter 5.2 we have already seen that the pattern-based approach does not suit all information demands. As the relation between an instance in I_a and a tag is not always expressed within a sentence, we compare our

pattern-based approach with two alternatives.

We base these methods on co-occurrences on the web. If the instances *Johnny Cash* and *U2* are often mentioned in the same context, we can conclude that these instances are related in some sense. The co-occurrences of instances on the web form the basis of the approach to deal with the problems as defined in this chapter. After discussing the alternatives, Section 6.2 handles the processing of these co-occurrences to solve the problems addressed in this chapter.

Pattern-based Method (PM). The Pattern-based Method (PM) is based on the methods described in the first part of this thesis. To find occurrences of relation instances (i, j) in $I_a \times I_g$ and (i, i') in $I_a \times I_a$, we apply this method and bookkeep the total number of co-occurrences encountered. From the snippets returned by the search engine, we thus identify the elements of either I_a or I_g to measure the number of co-occurrences of the pairs. Hence, using PM for instances i and j $\text{co}(i, j)$ is defined as follows,

Definition [PM co-occurrences]. $\text{co}_{\text{PM}}(i, j)$ gives the sum of the number of occurrences of i when querying patterns with j, plus the number of occurrences of j when querying patterns with i. □

Using PM we only need $\mathcal{O}(m+n)$ queries to collect co-occurrences of pairs in $I_a \times I_g$ and $I_a \times I_a$ for I_a of size n and I_g of size m.

Page-Count-based Method (PCM). As a first alternative to PM, we extract the estimated number of *hits* $\text{co}(i, j)$ [Cilibrasi & Vitanyi, 2007; Knees et al., 2004; Mika, 2007; Gligorov et al., 2007]. This method to find co-occurrences between instances is based on analyzing the total numbers of occurrences of pairs of instances on the web. We identify the co-occurrences $\text{co}(i, j)$ as follows,

Definition [PCM co-occurrences]. $\text{co}_{\text{PCM}}(i, j)$ gives the number of hits for the search engine query `"`i`", "`j`"`. □

We assume that the order of the terms i and j in the query does not effect the number of hits, thus we assume $\text{co}(i, j) = \text{co}(j, i)$.

This Page-Count-based Method (PCM) is simple and intuitive. If we are for example interested in categorizing music artists into genres, we analyze the number of hits to queries for combinations of the names of the artist and each genre. Assuming Johnny Cash to be a country artist, we expect that more documents contain both the terms *Country* and *Johnny Cash* than *Reggae* and *Johnny Cash*. An important drawback of PCM is the high Google complexity. For large sets this can

be problematic [Cafarella, Downey, Soderland, & Etzioni, 2005]. Moreover, the number of hits can fluctuate over time [Véronis, 2006], which hampers the reuse of old hit counts.

Using PCM we thus need to perform $m \cdot n$ queries to collect the co-occurrences between tags and instances and $\frac{1}{2}(n^2 - n)$ queries to gather all pairs of co-occurrences between the instances in I_a. Hence, the Google Complexity of PCM is $\mathcal{O}(mn + n^2)$. When we assume that the size of I_g does not exceed n, the Google Complexity of PCM is $\mathcal{O}(n^2)$.

Document-based Method (DM). In the Document-based Method (DM) approach we collect the first k URLs of the documents returned by the search engine for a given query, constructed using a known instance. These k URLs are the most relevant for the query submitted based on the ranking used by the search engine [Brin & Page, 1998]. The corresponding documents are subsequently scanned for occurrences of instances of the related class [De Boer et al., 2007].

In the first phase of the algorithm, we query all instances in both I_a and I_g and collect the top k documents for each of the queries. For instances in I_a, we retrieve each document using the URLs found by the search engine. We count the occurrences of the categories in I_g (thus the names of the categories) in the retrieved documents for the initial mapping m'. From the documents retrieved with a category $g \in I_g$, we similarly extract the occurrences of instances in I_a.

The documents obtained using DM are the most relevant for each element $b \in I_a$. For the instances in I_a queried, we expect to find biographies, fan pages, pages of museums, entries in database sites and so on. The labels in I_g (e.g. the genres, styles or other descriptors) mentioned in these pages will most probably reflect the genre of the artist queried. Thus $\mathrm{co}(i, j)$ is here defined as follows.

Definition [DM co-occurrences]. $\mathrm{co}_{\mathrm{DM}}(i, j)$ gives number of occurrences of j in documents found with i, plus the number of occurrences of i in documents found with j. □

Like PM, this method also requires only $\mathcal{O}(n+m)$ queries. However, additional data communication is required since for each query up to k documents have to be downloaded instead of using only the data provided by the search engine.

6.2 Processing Extracted Subjective Information

In the previous section, we discussed three methods to identify relation instances on the web. Here we show how we use the numbers of co-occurrences of these related instances to address the three problems presented in this chapter.

6.2.1 Identifying Relatedness between Instances

Having gathered a list of co-occurrences of instances in I_a using either PM, PCM or DM, we are interested to what extent these instances are expressed to be related. We assume that two instances are related when they are relatively often mentioned in the same context. For each instance i we could consider the instance $i' \in I_a$ with the highest $\mathrm{co}(i,i)$ to be the most related to i. However, we observe that, in that case, frequently occurring instances have a relatively large probability to be related to any other instance. This observation leads to an approach inspired by the theory of pointwise mutual information [Manning & Schütze, 1999; Downey et al., 2005]. We use $T(i,i')$ to express the relatedness of instances i' to i as follows,

$$T(i,i') = \frac{\mathrm{co}(i,i')}{\sum_{i'',i''\neq i'} \mathrm{co}(i'',i')}. \tag{6.1}$$

The function T can be normalized to t, i.e. with values $0 \leq t(i,i') \leq 1$

$$t(i,i') = \frac{T(i,i')}{\sum_{i''\in I_a} T(i,i'')}. \tag{6.2}$$

We address the Instance Relatedness Problem using $t(i,i')$ by identifying an ordered list of all instances related to i.

6.2.2 Categorizing Instances

The Instance Categorization Problem handles the identification of a most applicable $j \in I_g$ for a given instance $i \in I_a$. We use the co-occurrences between instances in I_a and I_g to compute scores $s(i,j)$ that express the applicability of tag j to instance i. For each instance i, we identify an initial mapping $m'(i)$ by selecting the tag with the highest score.

Subsequently, we investigate whether we can use the hypothesis that related instances often share a category, as we have created methods to identify relatedness between instances. We hence reuse the values $t(i,i')$ to find a final mapping.

Using either PM, PCM or DM, we can also acquire co-occurrence counts for pairs $(i,j) \in I_a \times I_g$. The function $S(i,j)$ expressing the extent of applicability of j to i is defined similarly as function T, namely

$$S(i,j) = \frac{\mathrm{co}(i,j)}{\sum_{i'\in I_a} \mathrm{co}(i',j)}, \tag{6.3}$$

and we normalize this function as follows

$$s(i,j) = \frac{S(i,j)}{\sum_{j'\in I_g} S(i,j')}. \tag{6.4}$$

Now, $s(i,j)$ can be read as the probability that tag j is applicable to i. If we are interested in the tag $m'(i)$ most applicable to i, we thus select the j such that $s(i,j)$ is maximized,

$$m'(i) = \text{argmax}_{j \in I_g} s(i,j). \tag{6.5}$$

The instance categorization problem focuses on the identification of one single tag or category for a given instance. We investigate whether we can improve the initial mapping m' by using the assumption that related instances in I_a often share a category. We are hence interested if the use of the computed relatedness between instances in I_a helps to improve the precision of the mapping m'.

We combine the scores t with the initial mapping m' as follows. For each $i \in I_a$, we inspect m' to determine the category that is assigned most often to i and its $k-1$ most related instances. We thus expect that the most appropriate category j for i is most often mapped by m' among i and its nearest neighbors.

For each instance $i \in I_a$, we construct an ordered list $\mathcal{B}_k(i)$ with i and its $k-1$ nearest neighbors

$$\mathcal{B}_k(i) = (i_1, i_2, ..., i_k)$$

with i as its first element, i.e. $i = i_1$, and

$$t(i,i_l) \geq t(i,i_{l+1}), \quad \text{for } 1 \leq l < k.$$

For a final mapping m of instances $i \in I_a$ to a category in I_g, we inspect the most occurring category mapped by m' to i and its $k-1$ nearest neighbors.

$$m(i,k) = \text{argmax}_{j \in I_g} \Big(\sum_{i' \in \mathcal{B}_k(i)} \tau(i',j) \Big)$$

with

$$\tau(i',j) = \begin{cases} 1 & \text{if } m'(i') = j \\ 0 & \text{otherwise.} \end{cases}$$

If two categories have an equal score, we select the first occurring one. That is, the category that is mapped by m' to i or to the instance most related to i.

Hence, we address the instance categorization problem by selecting the single tag (category, genre, etc.) $m(i,k)$.

6.2.3 Tagging Instances

With respect to the instance tagging problem, we assume that multiple tags may be applicable to an instance. Hence, we are interested in an ordered list of tags

for a given instance in I_a. Similar to the approach for the instance categorization problem, we will start with the scores $s(i, j)$ to compute an initial ordered list of tags for instance i. Likewise, we investigate whether the use of instance relatedness can lead to improvements over the initial tagging.

When addressing the instance categorization problem, we assumed the relation between instances and tags to be functional. That is, each instance in I_a was assumed to be related to at most one tag (e.g. a *genre* or *art style*). When dealing with the instance tagging problem however, we assume that multiple tags are applicable to a given instance. Thus the question is which of the tags are most applicable and to what extent.

The use of the score $s(i, j)$ is a first approximation to identify the tags most related to the given instance i. Similar to the computation of the final mapping m, we use the similarity between the instances in I_a to obtain a final score.

The degree of relatedness of an instance i' to i is given by $t(i, i')$. For tag j, the degree of applicability of j to i is given by $s(i, j)$.

We use the computed scores of relatedness $t(i, i')$ to improve the initial tagging $s(i, j)$. If two instances are closely related, we expect similar tags for the two. Hence, if i' is closely related to i, we want $s(i', j)$ to contribute significantly to the final score $p(i, j)$. Using the normalized scoring functions, we can compute the applicability $p'(i, j)$ of tag j to instance i as follows

$$p'(i, j) = \sum_{i', i' \neq i} t(i, i') \cdot s(i', j). \tag{6.6}$$

If erroneously a high score is found for $s(i, j)$, this error is decreased when close related instances i' have low scores for $s(i', j)$.

However, $p'(i, j)$ does not suffice as no self-relatedness score $t(i, i)$ is defined. We do consider $s(i, j)$ relevant when computing the scores for the tags with respect to instance i. Hence, we introduce a weight w for $s(i, j)$ as a substitute for $t(i, i)$ in the score $p(i, j)$,

$$p(i, j) = w \cdot s(i, j) + (1 - w) \cdot \sum_{i', i' \neq i} t(i, i') \cdot s(i', j). \tag{6.7}$$

Note that $p(i, j) = s(i, j)$ for $w = 1$. For instance i and tag j, $p(i, j)$ can be read as the confidence estimator that j is applicable to i. It simply shows that the sum of $p(i, j)$ for all tags j being applicable to i is 1.

Since both s and t are normalized, using simple calculus we show that p is normalized as well, i.e. the sum of $p(i, j)$ over all j equals 1.

$$\begin{aligned}
\sum_{j'} p(i,j') &= \sum_{j'} \Big(w \cdot s(i,j') + (1-w) \cdot \sum_{i',i' \neq i} t(i,i') \cdot s(i',j') \Big) \\
&= \sum_{j'} \big(w \cdot s(i,j') \big) \; + \; \sum_{j'} \big((1-w) \cdot \sum_{i',i' \neq i} t(i,i') \cdot s(i',j') \big) \\
&= w \cdot \sum_{j'} s(i,j') + (1-w) \cdot \sum_{j'} \sum_{i',i' \neq i} t(i,i') \cdot s(i',j') \\
&= w + (1-w) \cdot \sum_{j'} \sum_{i',i' \neq i} t(i,i') \cdot s(i',j') \\
&= w + (1-w) \cdot \sum_{i',i' \neq i} \sum_{j'} t(i,i') \cdot s(i',j') \\
&= w + (1-w) \cdot \sum_{i',i' \neq i} t(i,i') \cdot \sum_{j'} s(i',j') \\
&= w + (1-w) \cdot \sum_{i',i' \neq i} t(i,i') \\
&= w + (1-w) \\
&= 1
\end{aligned}$$

It now remains to find an appropriate value for w. One approach is to identify a training set of artists and related tags. Using the co-occurrences acquired we can determine the value of w, $0 \leq w \leq 1$, for which the scores of the tags fit the training set best. In the following section, we investigate whether the performance of the artist tagging method indeed improves for values of w smaller than 1.

Complexity Analysis. We analyze the computational complexity of the computation of values for $p(i,j)$. First we compute the complexity of creating all values for $t(i,i')$ and $s(i,j)$. We assume that the values for the co-occurrences scores $\mathrm{co}(i,j)$ and $\mathrm{co}(i,i')$ are stored in ordered lookup tables.

For instances i and i' with $\mathrm{co}(i,i') \geq 1$, $t(i,i')$ can be rewritten as follows,

$$\begin{aligned}
t(i,i') &= \frac{T(i,i')}{\sum_{i_0} T(i,i_0)} \\
&= \frac{\frac{\mathrm{co}(i,i')}{\sum_{i'',i'' \neq i'} \mathrm{co}(i'',i')}}{\sum_{i_0} \frac{\mathrm{co}(i,i_0)}{\sum_{i_1,i_1 \neq i_0} \mathrm{co}(i_1,i_0)}} \\
&= \frac{\mathrm{co}(i,i')}{c(i') \cdot \sum_{i_0} \frac{\mathrm{co}(i,i_0)}{c(i_0)}}
\end{aligned}$$

where $c(i)$ is given by

$$c(i) = \sum_{i',i' \neq i} \mathrm{co}(i,i'). \tag{6.8}$$

The value for $c(i)$ can be simply computed using n steps, where n is the size of I_a. Hence, computing $c(i)$ for all i requires $\mathcal{O}(n^2)$. The values of $c(i)$ and $\mathrm{co}(i,j)$ can hence be looked up in $\mathcal{O}(\log n)$, while the computation of $t(i,j)$ requires n lookups for $c(i_0)$. Hence, $t(i,i')$ can be computed in $\mathcal{O}(n \log n)$. Hence, computing all values for $t(i,i')$ requires a time complexity $\mathcal{O}(n^3 \log n)$.

The computation of $s(i,j)$ can be done in a similar fashion, requiring $\mathcal{O}(m \log(nm))$ steps, where m is the size of I_g. All values of $s(i,j)$ are thus computed in $\mathcal{O}(nm^2 \log(nm))$.

We store the values of s and t again in a lookup table. Assuming that the number of tags in I_g does not exceed the size of the set of instances in I_a, we conclude that the preprocessing step requires a computational complexity of $\mathcal{O}(n^3 \cdot \log n)$.

Having computed all values for $s(i,j)$ and $t(i,i')$, we can now compute $p(i,j)$ for a given w. The values for $t(i,i') \cdot s(i',j)$ are computed in $n-1$ steps of two lookups. Hence, the value for $p(i,j)$ can be computed in $\mathcal{O}(n \log n)$. To create ordered lists of tags for all instances in I_a thus requires a time complexity of $\mathcal{O}(m \cdot n^2 \log(n))$.

In total, the time complexity for the computation of all values of $p(i,j)$ is thus $\mathcal{O}(n^3 \log n + nm^2 \log(mn) + mn^2 \log n)$. If we assume $m < n$, the complexity is thus $\mathcal{O}(n^3 \log n)$. Especially for PCM this value is realistic, as it is to be expected that most co-occurrence counts are at least 1.

6.3 Evaluating Extracted Subjective Information

In this section, we investigate whether information from social websites can be used to evaluate the populated ontologies computed with the methods discussed in the previous section. We focus on one of the larger social websites, *Last.fm*, and its topic: music. We investigate the consistency of the tags as provided by the *Last.fm* community and compare this data with the concept *genre* that is often used by professionals to characterize music.

Researchers in music information retrieval widely consider musical genre to be an ill-defined concept [Aucouturier & Pachet, 2003; Scaringella, Zoia, & Mlynek, 2006; McKay & Fujinaga, 2006]. Several studies also showed that there is no consensus on genre taxonomies [Aleksovski, Kate, & Harmelen, 2006; Pachet & Cazaly, 2000]. However, automatic genre classification is a popular topic of research in music information retrieval (e.g. [Basili, Serafini, & Stellato, 2004; Tzanetakis & Cook, 2002; Li, Ogihara, & Li, 2003; Pampalk, Flexer, & Widmer, 2005; Schedl et al., 2006]).

McKay and Fujinaga [2006] conclude that musical genre classification is worth pursuing. One of their suggestions is to abandon the idea that only one genre is applicable to a recording. Hence, multiple genres can be applicable to one recording and a ranked list of genres should be computed per recording.

Today, the content of web sites such as *del.icio.us*, *flickr.com* and *youtube.com* is generated by their users. Such sites use community-based *tags* to describe the available items (photos, films, music, (scientific) literature, etc.). Although tags have proven to be suitable descriptors for items, no clear semantics are defined. Users can label an item with any term. The more an item is labeled with a tag, the more the tag is assumed to be relevant to the item.

Last.fm is a popular internet radio station where users are invited to tag the music and artists they listen to. Moreover, for each artist, a list of similar artists is given based on the listening behavior of the users. Ellis et al. [2002] propose a community-based approach to create a ground truth in musical artist similarity. The research question was whether artist similarities as perceived by a large community can be predicted using data from All Music Guide and from shared folders for peer-to-peer networks. Now, with the *Last.fm* data available for downloading, such community-based data is freely available for non-commercial use.

In *Last.fm*, tags are terms provided by users to describe music. They "are simply opinion and can be whatever you want them to be"[1]. For example, Madonna's music is perceived as *pop*, *glamrock* and *dance* as well as *80s* and *camp*. When we are interested in describing music in order to serve a community (e.g. in a recommender system), community-created descriptors can be valuable features.

In this section we investigate whether the *Last.fm* data can be used to generate a ground truth to describe musical artists. Although we abandon the idea of characterizing music with labels with defined semantics (e.g. genres), we follow the suggestion of MacKay and Fujinaga [2006] to characterize music with a ranked list of labels. We focus on the way listeners perceive artists and their music, and propose to create a ground truth using community data rather than to define one by experts. In line with the ideas of Ellis et al. [2002], we use artist similarities as identified by a community to create a ground truth in artist similarity. As tastes and opinions change over time, a ground truth for music characterization should be dynamic. We therefore present an algorithm to create a ground truth from the dynamically changing *Last.fm* data instead of defining it once and for all.

6.3.1 Analyzing the Last.fm Tags

Last.fm users are invited to tag artists, albums, and individual tracks. The 100 top-ranked tags (with respect to the frequency a tag is assigned) for these three

[1] `http://www.Last.fm/help/faq/?category=Tags`

rap	Gangsta Rap
Hip-Hop	Aftermath
hip hop	favorites
Eminem	metal
hiphop	Favorite
pop	rnb
rock	dance
alternative	american
detroit	classic rock
seen live	r and b

Table 6.1. Top 20 tags for Eminem.

categories are easily accessible via the Audioscrobbler web services API[2]. By analyzing the listening behavior of its users, *Last.fm* also provides artist similarities via Audioscrobbler[3]. Per artist, a list of the 100 most similar artists is presented.

We first analyze tags for artists. As the lists of the top-ranked tags tend to contain noise, we propose a simple mechanism to filter out such noise (Section 6.3.2). In order to check the consistency of the tags, we inspect whether users label similar artists with the same tags. We end Section 6.3 with a proposed mechanism to create a dynamic ground truth in artist tagging and similarity using *Last.fm* data.

Tagging of Artists

In Table 6.1, the 20 top-ranked tags for the artist Eminem are given, as found with the Audioscrobbler web service. The terms *rap*, *hiphop* and *detroit* can be seen as descriptive for the artist and his music. Eminem is tagged with multiple terms that reflect a genre but the tag *rap* is more significant than *metal*.

Without questioning the quality or applicability of the terms in the list in Table 6.1, we observe some noise in the tagging of this artist. Whether we consider Eminem to be a hip-hop artist or not, after encountering the second highest ranked tag *Hip-Hop*, the tags *hip hop*, *hiphop* do not provide any new information. Moreover, the tag *Eminem* does not provide any new information with respect to the catalog data. The tags *favorite* and *good* do not seem very discriminative.

To investigate whether the tags are indeed descriptive for a particular artist, we collected the tags applied to a set of artists. In [Schedl et al., 2006][4], a list of 1,995 artists was derived from All Music Guide. We calculated the number of artists that

[2] `http://ws.audioscrobbler.com`

[3] e.g. `http://ws.audioscrobbler.com/1.0/artist/Madonna/similar.xml`

[4] `http://www.cp.jku.at/people/schedl/music/C1995a_artists_genres.txt`

jazz (809)	country (308)
seen live (658)	hard rock (294)
rock (633)	singer songwriter (291)
60s (623)	oldies (289)
blues (497)	female vocalists (285)
soul (423)	punk (282)
classic rock (415)	folk (281)
alternative (397)	heavy metal (277)
funk (388)	hip-hop (267)
pop (381)	instrumental (233)
favorites (349)	rnb (231)
american (345)	progressive rock (229)
metal (334)	electronica (215)
electronic (310)	dance (209)
indie (309)	alternative rock (208)

Table 6.2. The 30 most popular tags and their frequencies for the set of 1995 artists.

grimey (1)	stuff that needs further exploration (1)
disco noir (1)	american virgin festival (1)
gdo02 (1)	lektroluv compilation (1)
808 state (1)	electro techo (1)
iiiii (1)	richer bad rappers have not existed (1)
mussikk (1)	crappy girl singers (1)
good gym music (1)	techno manchester electronic acid house (1)
knarz (1)	music i tried but didnt like (1)

Table 6.3. Some of the least used tags for the 1995 artists.

are labeled with each of the tags. The most frequently occurring tags over all artists are given in Table 6.2. Table 6.3 contains some of the tags that are applied only to one artist. For the 1,995 artists, we encountered 14,146 unique tags.

If a tag is applied to many diverse artists, it cannot be considered to be discriminative. We observe that there are no tags that are applied to a majority of the artists. The high number of artists labeled with jazz can be explained by the fact that the 1,995-artist-set contains 810 jazz artists. All frequent tags seem relevant characterizations for musical artists or for the relation of the users to the artists (e.g. *seen live*).

The most debatable tag among the best scoring ones may be *favorites*. Table 6.4

Radiohead	Coldplay
The Decemberists	Pink Floyd
Death Cab for Cutie	The Postal Service
The Beatles	Bright Eyes
The Shins	Elliot Smith

Table 6.4. The 10 top artists for the tag 'favorites'.

contains a list of the top artists for this tag, as extracted from audioscrobbler. We notice that no mainstream dance or pop artists are among the list of 100 top artists for *favorites*. The 100 top artists for *seen live* are artists that toured in the 00s.

Tags that are applied to only one, or only a few artists are not informative either. Since we do not consider the semantics of the tags, uniquely occurring tags cannot be used to compute artist similarities.

We observe that the tags that are only applied once to artists in this set are more prosaic, are in a language other than English, or simply contain typos (cf. "electro techo" in Table 6.3). It is notable that in total 7,981 tags (56%) are applied to only one artist. Only 207 tags are applied to at least 50 out of the 1,995 artists.

To check whether the 7,981 tags are descriptive for a larger set of artists, we computed the *top count*. For each of the at most 100 top artists[5] for this tag, we extract the number of times n_i the tag is applied to artist i. The top count is the sum over all (at most 100) n_i. Table 6.5 contains examples of tags applied once in the 1995 artist collection and their top counts. If this sum is one, only one user has tagged one artist with this tag. Hence, the larger the top count, the more people will have used the tag to describe their music. Out of these 7,981 tags, 7,238 have a top count of at most 100. For comparison, the tag 'rock' has a top count of 150,519. Hence, we can conclude that the tags that are found only once in a collection of artists are in general uncommon descriptors for an artist.

Based on these small experiments, we conclude that most frequently used tags are relevant characterizations for musical artists. Moreover, although users can label an artist with any term, the list of frequently used tags is relatively small. We conclude that the number of tags that describe multiple artists is in the order of thousands. If we select the tags that apply to 5% of the artists, the number is in the order of hundreds.

[5] e.g. `http://ws.audioscrobbler.com/1.0/tag/post-hardcore/topartists.xml` gives the top artists and counts for *post-hardcore*

post-hardcore (8134)	fagzzz (0)
twee (4036)	when somebody loves you (0)
futurepop (3162)	ravens music (0)
mathcore (2865)	bands i met (0)
piano rock (2558)	most definitely a bamf (1)

Table 6.5. Examples of tags occurring only once with the high and low top counts.

6.3.2 Filtering the Tags

As indicated above, not all tags provide sufficient information for our task since tags occur with small spelling variations and catalog data (such as the names of artists or songs) are used as tag as well. Moreover, tags that are only applied to few artists cannot be used to discriminate between artists, as no semantics are defined for tags. Suppose that we have a collection I_a of artists. We present a simple method to filter out such meaningless tags.

Normalizing Tags. As we want tags to be descriptive, we filter out tags attached to $i \in I_a$ as follows.

- If a tag equals the name of the artist, we remove it.
- We compute a normalized form for all tags by
  - turning them into lowercase,
  - computing the stem of all words in the tags using Porter's stemming algorithm [Porter, 1980], and
  - removing all non-letter-or-digit characters in the tags.
- If two tags have the same normalized form, we remove the second one in the list.
- We remove every infrequently applied tag. In our experiments, we remove the tags that are applied to less than 5% of the artists in I_a.

Track Filtering. As we want the tags to reflect the music of the artist, we propose a next filtering step based on the tags applied to the best scoring tracks of the artist. *Audioscrobbler* provides the most popular tracks per artist, based on the listening behavior of the *Last.fm* users. As tracks can also be tagged individually, we can compare the tags applied to the artist with the tags applied to the top tracks. In the Track Filtering step, we filter out tags applied to the artist, that are not applied to his top tracks.

hip hop	alternative
Eminem	seen live
hiphop	metal
Aftermath	classic rock

Table 6.6. Tags removed for Eminem after normalization (l.) and track-filtering (r.).

By removing the tags that do not reflect the (most popular) music of an artist, we perform a second filtering step.

- We collect the n top-ranked tracks according to *Last.fm* for every artist in the list.
- For each of these, we retrieve the most popular tags.
- We compute a normalized form for the tags for each track.
- For the list of the normalized tags of $i \in I_a$, we retain only those whose normalized form is applied to at least m out of the n top-ranked tracks for the respective artist.

In our experiments, we choose to retain the tags that are applied to at least 3 out of the 10 top-ranked tracks for the respective artist.

The tags from Table 6.1 for *Eminem* that are removed after normalization and track filtering are given in Table 6.6.

6.3.3 Checking the Consistency of the Tags

The artist similarities as provided by *Last.fm* are based on the listening behavior of the users. Since we want to use the *Last.fm* data as ground truth in music characterization, we investigate whether the tagging is consistent, i.e. similar artists should share a large number of tags.

To ensure this criterion, we selected the set of 224 artists used in [Knees et al., 2004][6], where the artists were originally chosen to be representatives of 14 different genres. For each of the 224 artists, we collected the 100 most similar artists according to *Last.fm*. For the resulting set of the 224 artists and their 100 nearest neighbors, we downloaded the lists of the 100 most popular tags. For each artist in the list of 224, we first compared the list of tags of the most similar artist. We did the same for the following (less) similar artists in the list.

[6] `http://www.cp.jku.at/people/knees/publications/artistlist224.html`

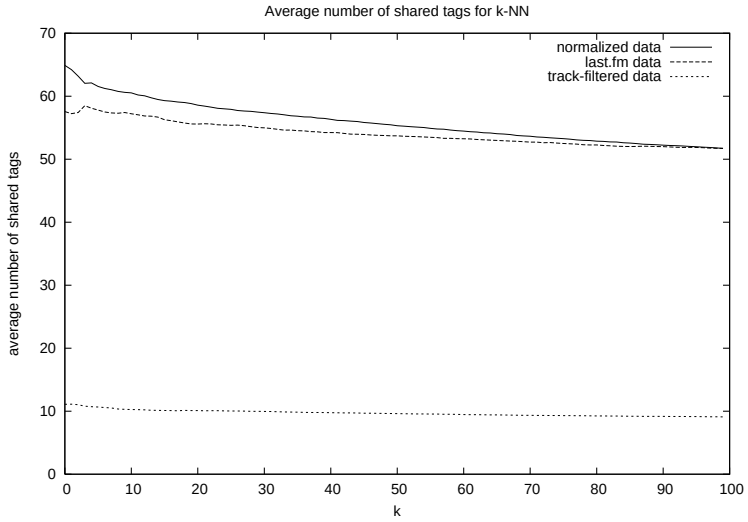

Figure 6.1. Average number of shared tags for the 224 artists.

We computed the average number of overlapping tags for the 224 artists and their k nearest neighbors and display the results in Figure 6.1. As – especially after track filtering – often less than 100 tags are assigned to each artist, we also computed the *similarity score* for each of the 224 artists and their k nearest neighbors by taking the average number of tags relative to the total number of tags for the nearest neighbors. For example, if an artist shares 34 out of 40 tags with an artist in the list of 224, the relative tag similarity score for this artist is $34/40$. The average similarity scores are given in Figure 6.2. The scores are computed using unfiltered, normalized and track-filtered *Last.fm* data.

The average number and score of overlapping tags decreases only slightly for the unfiltered and normalized data with increasing k. For the track-filtered data, we even note a small increase in the relative amount of tags shared (starting from $k = 25$). This can be explained by the small number of tags that remain after track-filtering, as can be found in Figure 6.1.

Using the unfiltered *Last.fm* tags of all retrieved artists, we estimate the expected number of tags shared by two randomly chosen artists as 29.8 and the relative number of shared tags as 0.58. When we filter the tags by normalization and compare the normalized forms of the tags, we obtain an average of 29.8 shared tags, with a relative number of 0.62. For the track filtering, these numbers are 3.87 and 0.64 respectively. Hence, the number of tags shared by similar artists is indeed much larger than that shared by randomly chosen artists.

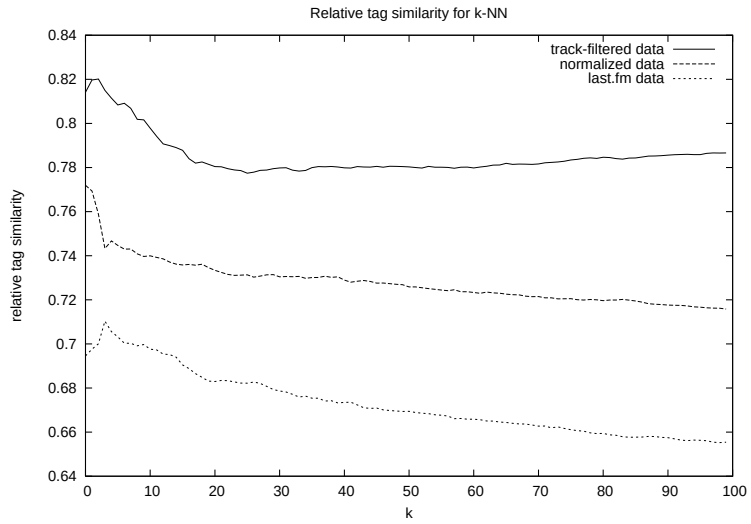

Figure 6.2. Relative tag similarity score for the 224 artists and their k Nearest Neighbors

6.3.4 Evaluating with Data from a Folksonomy

In earlier work (e.g. [Schedl et al., 2006; Pohle, Knees, Schedl, & Widmer, 2007; Geleijnse & Korst, 2006b]) computed artist similarities were evaluated using the assumption that two artists are similar when they share a genre. To our best knowledge, only the tagging of artists with a single tag, usually a genre name, has been addressed in literature. Also in other domains than music, the automatic creating of a list of tags from unstructured texts from multiple pages on the web has not been addressed.

As the *Last.fm* data shows to be reliable, we propose to use it as a ground truth for evaluating algorithms that identify tags for artists tagging and compute artist similarity. The use of such a rich, user-based ground truth gives better insights in the performance of the algorithm and provides possibilities to study the automatic labeling of artists with multiple tags. Moreover, by evaluating a method using artists and *Last.fm* ground truth we gain insights in the output of the method. High quality output for the musical artist data may lead to confidence on domains that can not be evaluated as easily.

A Dynamic Ground Truth Extraction Algorithm

As the perception of users changes over time, we propose a dynamic ground truth to evaluate a populated ontology with tags and instances. In the evaluation section of this chapter, we will use this evaluation method to evaluate populated ontologies on the artists using *Last.fm* data. Moreover, an ontology on books is similarly

evaluated using the social website *LibraryThing.com*.

For the evaluation of similarity of instances in I_a (e.g. artists), we use the similar artists for the artists in the set I_a as provided by the social website. For the lists of similar instances, we discard the artists that are not in I_a.

To create a ground truth for the ranked tags applicable to the artists in I_a, we download the top tags for each artist and compute the normalized tags as described in Section 6.3.2. The set of known tags I_g is constructed by collecting all normalized tags applied to the I_a artists.

Proposed Evaluation Measures

For a tag or an artist t_i given by the ground truth, $g_a(t_i)$ denotes the rank of t_i with respect to artist a. Hence, $g_a(t_i) - 1$ tags or artists are considered to be more applicable (or similar) to a than t_i. In contrast, with $r_a(t_i)$ we denote the rank of t_i for a as computed by the method to be evaluated.

We propose two evaluation measures. The first focuses on the traditional information retrieval measures precision and recall, the second evaluates the ranking.

Precision and Recall. We select the set S_n of the top n tags for artist a in the ground truth and evaluate precision and recall of the computed ordered list $\mathcal{L}_m$ of the m most applicable tags according to the tagging approach to be evaluated.

Ranking. We do not only consider the retrieval of the n top-ranked tags in the ground truth to be important, but we also want to evaluate the ranking itself, hence the correlation between the ranking in the ground truth $g_a(t_i)$ and the computed ranking $r_a(t_i)$. We evaluate the ranking for each artist using a standard measure, Spearman's Rank Correlation Coefficient [Kendall, 1975], as follows.

$$\rho(a) = 1 - \frac{6\sum_i (g_a(t_i) - r_a(t_i))^2}{n(n^2-1)}$$

We have now proposed a test set of artists and an algorithm to dynamically create a ground truth. Such ground truths are used in Section 6.4.3 to evaluate the tagging of musical artists and books.

6.4 Experimental Results

In this section, we focus on experiments conducted on each of the three problems described in the beginning of this chapter.

6.4.1 Identifying Relatedness between Instances

We first focus on the extraction of the relatedness of instances from the web. We will apply the methods as discussed in Section 6.2.1 and return to the historical

"like [person] and [person]"	*"namely [person] and [person]"*
"such as [person] and [person]"	*"[person] and [person]"*
"including [person] and [person]"	*"[person] [person] and other"*
"for example [person] and [person]"	

Table 6.7. Patterns used to find co-occurrences within the search engine snippets.

persons extracted (Chapter 4). Subsequently we discuss the identification of relatedness within a set of musical artists.

Famous People

Having gathered a list of historical persons with biographical information (Section 4.5), we are interested to know how the persons in the list are perceived to be related. Obviously such information can be extracted from the biographies, e.g. persons can be considered related when they share a profession, have the same nationality or lived in the same period.

However, we are interested in the way people nowadays relate the historical people extracted. For example, we are interested to identify the person who is considered to be most related to Winston Churchill. We therefore mine the web for a social network of people extracted using the method in the previous section.

We assume that two persons are related when they are often mentioned in the same context. Using the hypothesis that enumerated items are often related, we use the pattern-based approach by selecting enumeration patterns (Table 6.7).

For each of the best 3,000 ranked persons found in Section 4.5, we computed a ranked list based on $t(p,q)$ of most related persons in the large set of the 10,000 persons with biographies.

Aiming for a reflection of the collective knowledge of web contributors on historical figures, the extracted social network of historical persons is not a verifiable collection of facts. We illustrate the social network extracted by two examples. Figure 6.5 depicts the relatedness among the best ranked persons. An arrow from person p to q is drawn if q is among the 20 nearest neighbors of p. Using the same criterion, Figure 6.6 depicts the relatedness among the best ranked authors.

We are able to verify the precision of the relatedness between historical persons if we make the following assumptions. We consider the following "minimal criteria for" two persons to be related:

- either they lived in the same period, i.e. there is an overlap in the periods the two lived, or
- they shared a profession, or
- they shared a nationality, or

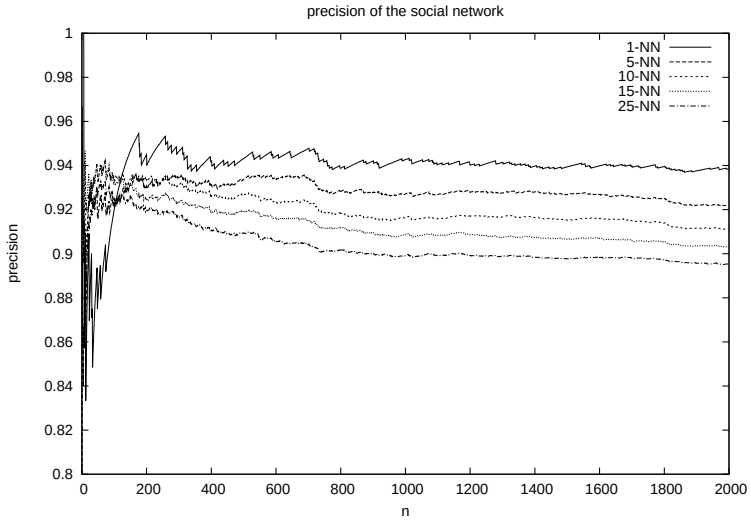

Figure 6.3. Precision for the social network for the n highest ranked persons and their k nearest neighbors.

- they are both female.

Of course we cannot evaluate recall of the algorithm on these criteria, as for example not all persons sharing a nationality need to be consider to be related. We therefore evaluate precision of the social network on these minimal criteria. For the 3,000 best ranked persons, we select the k most related persons. Per pair we evaluate whether either one of the four criteria is being met. This precision rate is presented in Figure 6.3. In comparison, the probability of any of the 3,000 persons to be related to a person in the large list of 10,000 is 45%. The precision rates for the social network per criterion can be found in Figure 6.4. The probabilities for two randomly selected persons to share a period, profession and nationality are 38%, 7.5% and 6.5% respectively. The chance that two historical persons are both female is only 0.5% for the studied list of 10,000. We hence conclude that these results give good confidence in the quality of the extracted social network.

Musical Artists

In the second case-study on the extraction of a network of related instances, we focus on musical artists. Hereto, we use two standard sets of artists: a set of 224 artists, equally divided over 14 genres [Knees et al., 2004] and a large set of 1995 artists divided over 9 genres [Schedl et al., 2006]. For both sets of artists, each artist is only associated with one genre. We consider two artists to be similar, if

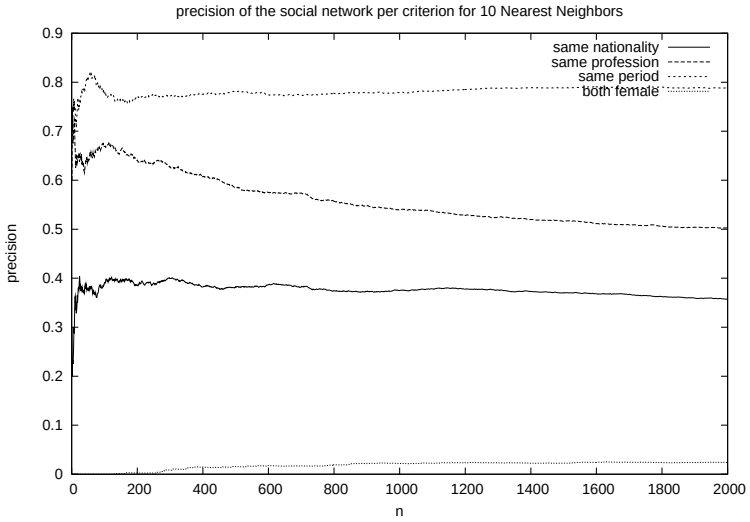

Figure 6.4. Precision for 10-NN per criterion.

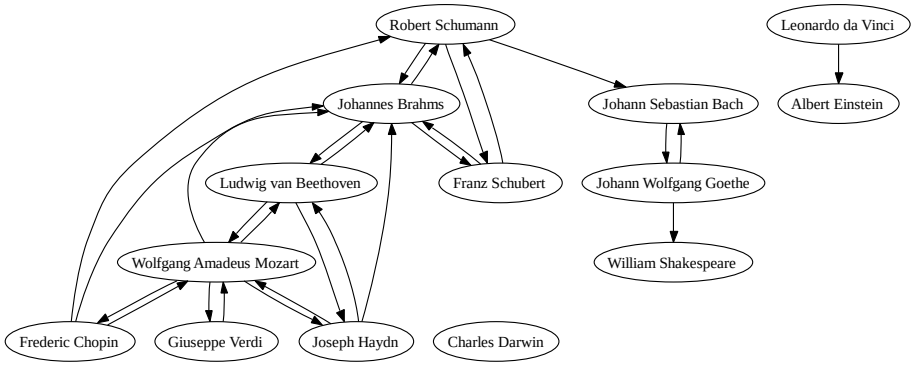

Figure 6.5. The extracted social network for the 15 highest ranked persons.

they share a genre in the test set.

We use the common test set I_{224} of 224 artists, equally divided over 14 genres as defined by Knees et al. [2004][7] to evaluate the computed artist similarities $t(i,i')$. We consider two artists to be similar, if they share a genre in the test set. In these experiments, we only evaluate precision. If for an artist i no mapping or related instance could be found, we consider the result to be incorrect.

In this case-study, we compare the results of the three alternative methods to obtain the co-occurrence counts. For PCM we added the extra term *music* for find-

[7] www.cp.jku.at/people/knees/publications/artistlist224.html

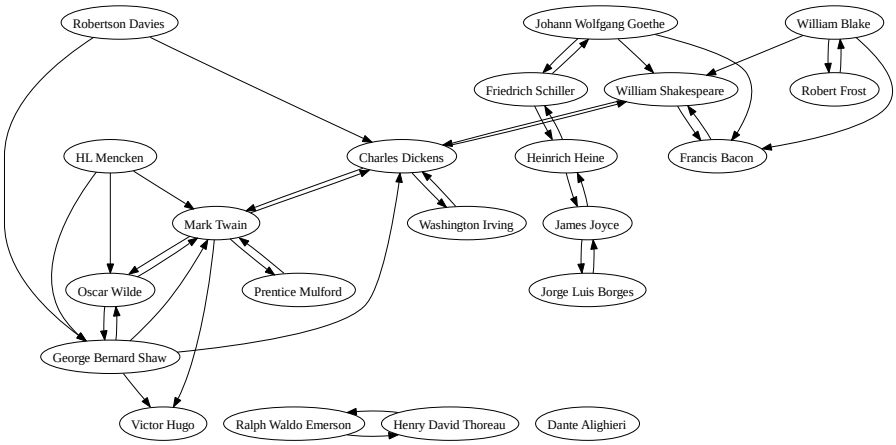

Figure 6.6. The extracted social network for the highest ranked authors.

like [Artist] and [Artist]
such as [Artist] and [Artist]
including [Artist] and [Artist]
namely [Artist] *and* [Artist]
[Artist] and [Artist]
[Artist] [Artist] and other

Table 6.8. patterns for artist - artist relation.

ing co-occurrences of the artists. For example the terms *Bush* and *Inner Circle* co-occurred a lot on the web, due to American politics. By adding the term *music* we restrict ourselves to documents handling music.

Since we are not interested in the nature of the relatedness between artists, for PM we selected general enumeration patterns (Table 6.8) to obtain co-occurrences.

Figure 6.7 shows the average precision of the similarity of the artists and their k-NN for the sets of 224 artists. Note that for each artist only 15 others are defined to be related. We can conclude that the pattern based method PM gives good results and outperforms both DM and PCM. For smaller values of k the method most inefficient in the number of queries is outperformed by both DM and PM. The performance of DM drops quickly due to the fact that only few related artists are mentioned among the highest ranked pages for the queried instances.

We have also compared the results of the three methods with the data from *Last.fm*. For each of the 224 artists, we have extracted a ranked list of similarities

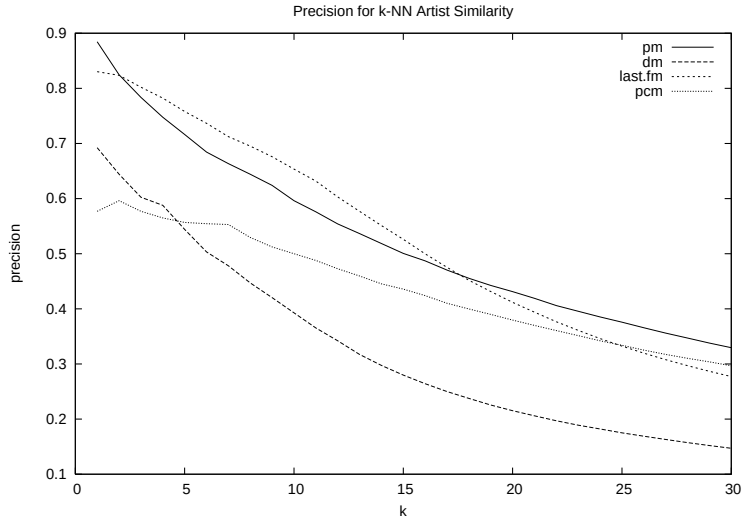

Figure 6.7. Precision for the categorization of the 224 musical artists compared with the data extracted from Last.fm.

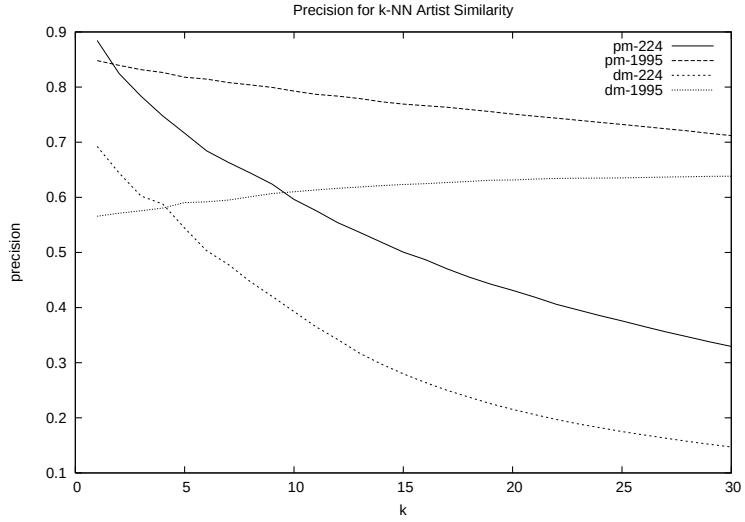

Figure 6.8. Precision for the sets of 224 and 1995 artists.

with the other 223 artists[8]. For small sizes of k the *Last.fm* data is more precise than PCM, but for larger k all web-based methods outperform the experiment with the *Last.fm* experiment. This can be explained by the fact that for each artist only 100 similar artists are provided. Hence, the average number of identified related

[8] e.g. `http://ws.audioscrobbler.com/1.0/artist/Madonna/similar.xml`

Instructors can add Voice **Tool** and **Live** Classroom links directly to the Vista Calendar.
... States W2, Canada **B12**, and **Japan** [H18, K9] indicate ...
If you work in a large population center, for example, **Chicago**, **Boston**, New York City, and ...
They go together like **lamb** and tuna **fish**,

Table 6.9. Not all composed queries lead to relevant texts.

artists in the set of 224 is lower than all web-based methods.

Figure 6.8 shows the average precision of the similarity of the artists and their k-NN for the sets of 224 and 1995 artists. We can conclude that the pattern based method gives good results and outperforms DM in both sets. For the set of 1995 we did not compute the co-occurrences using PCM, as this would take over a year using the *Yahoo!* API.

Dealing with Ambiguity. As the use of PM leads to good results in the identification of artist similarities, we applied the method to a collection of 1732 artists, used in music recommender experiments [Tiemann & Pauws, 2007]. As this set contains a number of ambiguous artist names, we return to the work discussed in Section 3.2.1, and propose a method to identify related instances for a set I_a with ambiguous terms.

We computed the artist similarities using t. Since no ground truth is available for this collection of artist, we cannot evaluate the precision. In Table 6.11 we give the top most related artists to six of the artists in the collection using the method as discussed in Section 6.2.1.

We observe that the artists *Tool*, *Live* and *Fish* frequently occur amidst the most related artists. Especially for lesser famous artists, where the data is sparse, these artist can be found often among the nearest neighbors. *Tool* is for 1227 out of the 1731 other artists one of the 5 most similar artists, *Live* is 1334 times in the top 5 and *Fish* is 724.

Unlike most of the artist names in the commonly used evaluation sets, these frequently occurring artist names are very ambiguous. A number of examples of irrelevant snippets due to artist name ambiguity can be found in Table 6.9.

In Section 3.2 we addressed this problem by using *Google*'s define functionality. We use the number of definitions as an estimator for the probability that the term indeed reflects the intended instance.

ARTIST	BASELINE	p_{lin}	p_{sqrt}
Live	1227	1	54
Tool	1334	0	642
Fish	724	0	7
Juli	691	1251	1207

Table 6.10. Number of times an ambiguous artist name occurs among the top 5 nearest neighbors of the 1731 other artists.

Ideally, for each occurrence of an artist name in a text we want to observe whether the occurrence indeed reflects the intended artist. However, the automatic parsing of sentences is troublesome as the snippets contain broken sentences and may be multilingual. Moreover if an artist name is identified as a subject or object within a sentence, then we still do not know whether the term indeed reflects the artist.

We therefore aim for a method where we estimate the probability that a term a indeed reflects the intended artist named a. Using functions p_{lin} (equation (3.3) on page 43) or p_{sqrt} (equation (3.4)), we estimate the relatedness between two instances as follows,

$$T'(a,b) = \frac{\text{co}'(i,i')}{\sum_{i'',i''\neq i'} \text{co}'(i'',i')}, \tag{6.9}$$

with

$$\text{co}'(i,i') = \text{co}(i,i') \cdot p(i) \cdot p(i'). \tag{6.10}$$

Note that for $p(i) = p(i') = 1$, we have the baseline function $T(i,i')$.

In this section we investigate the effect of the use of the ambiguity correction on the performance on the test sets.

For both the sets of 224 and 1995 artists, we collected the numbers of definition for all the artist names. We recomputed the artist similarities using the linear approach p_{lin} and the square root approach p_{sqrt} and compared the two with the baseline.

An alternative approach is to explicitly add terms such as 'music' to the query expression. However, this approach leads to less snippets, while the snippets re-

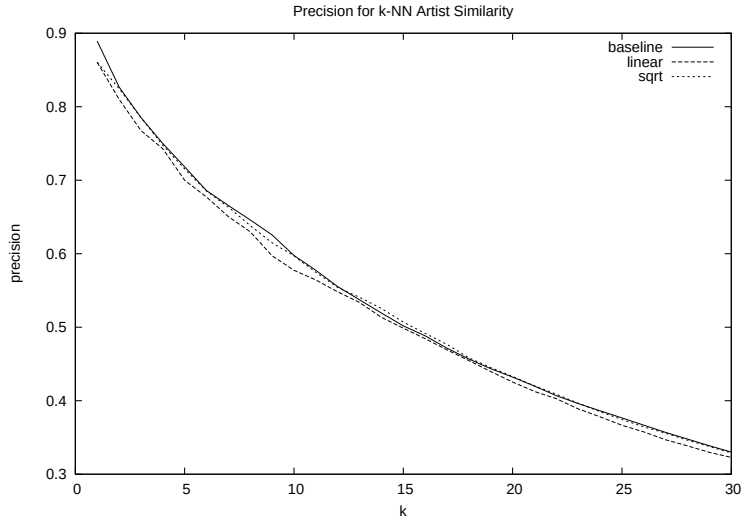

Figure 6.9. Precision for the sets of 224 artists using the three ambiguity estimators.

turned contained less related instances.

We present the results for the sets I_{224} and I_{1995} in Figures 6.9 and 6.10. For the set of 224 the performance of the methods using disambiguation is slightly less than that of the baseline approach. This result is expected, as no ambiguous terms occur in the set of 224 artists. For the set of 1995 artists however, the results improve using either the *uniform* or the *sqrt* approach. We note that contrary to the set of 224 artists, the 1995 set does contain some ambiguous names such *Autograph*, *Gamma Ray* and *Hypocrisy*.

For the set of 1732 artists in our own collection, we compare the number of times that ambiguous artist names occur among the 5 nearest neighbors for the other artists (Table 6.10). We note that for the term *Juli* only one definition is found. Although the distribution of ambiguous names is quite different for p_{lin} and p_{sqrt}, we cannot draw conclusions on which approach is better suited as currently no ground truth for artist similarity ranking is available. Hence, a ground truth data set for such a diverse collection with ambiguous artist names is needed. With such a set, we can obtain better insights in the quality of web information extraction methods for these purposes.

6.4.2 Categorizing Instances

In this subsection, we focus on experiments that address the categorization of the instances in I_a by selecting the most applicable label in I_g.

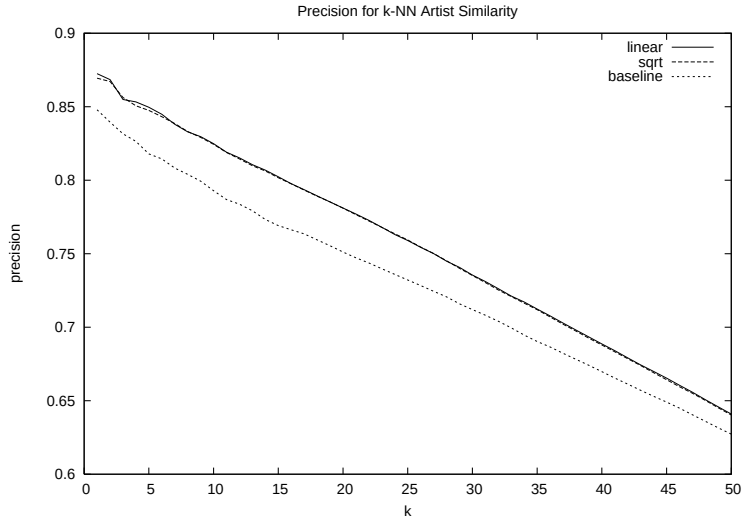

Figure 6.10. Precision for the sets of 1995 artists using the three ambiguity estimators.

Musical Artists

In this experiment, I_{224} is again the set of all artist names in the list composed by Knees et al. [2004]. This list consists of 14 genres, each with 16 artists.

To find the most appropriate genre for the artists in I_{224}, the genres mentioned in the list are not all suitable for finding co-occurrences. For example, the term *classical* is ambiguous and *Alternative Rock/Indie* is an infrequent term. We therefore manually rewrote the names of the genres into unambiguous ones (such as *classical music*) and added some synonyms. After collecting the numbers of co-occurrences of artists and genres, we summed up the scores of the co-occurrences for synonyms. In this way, for each artist b the number of co-occurrences with the terms *Indie* and *Alternative Rock* are added to the co-occurrences of b with the genre *Alternative Rock/Indie*. Although the absolute number of co-occurrences with *Alternative Rock/Indie* may increase using this approach, it is notable that we use a relative measure to determine the most applicable category per artist.

Motivated by the results in [Schedl et al., 2005], for PCM we used the `allintitle` option in the artist categorization experiment.

For PM we selected for the genre-artist relations the patterns in Table 6.12 from a list of patterns expressing this relation.

For all three methods, we reuse the artist similarities computed in the previous experiments.

In Table 6.13 the performance of the initial mappings can be found for the

	baseline	using p_{lin}	using p_{sqrt}
Babylon Zoo:			
1.	Tool	Juli	Juli
2.	Live	Chumbawamba	Chumbawamba
3.	Fish	Jamiroquai	Jamiroquai
4.	Juli	Shakira	Tool
5.	Chumbawamba	Sonic Youth	Shakira
6.	Play	Right Said Fred	Janet Jackson
B12:			
1.	Tool	Juli	Juli
2.	Live	Carl Craig	Carl Craig
3.	Fish	Jamiroquai	Jamiroquai
4.	Juli	Autechre	Tool
5.	Play	Shakira	Autechre
6.	Japan	Speedy J.	Shakira
B. Springsteen:			
1.	Neil Young	T. Petty & Heartbreakers	Neil Young
2.	U2	Tom Petty	T. Petty & Heartbreakers
3.	Bob Dylan	The Afghan Wigs	Tom Petty
4.	Tom Petty	Neil Young	The Afghan Wigs
5.	Tool	Patti Smith	Bob Dylan
6.	The Afghan Wigs	Robert Plant	Patty Smith
Tool:			
1.	Freefrom	Mudvayne	Mudvayne
2.	Mudvayne	Type O Negative	Type O Negative
3.	Racoon	Hothouse Flowers	Hothouse Flowers
4.	Strauss	Massive Attack	Massive Attack
5.	Type O Negative	Nine Inch Nails	Nine Inch Nails
6.	Hothouse Flowers	Dream Theater	Dream Theater
U2:			
1.	Hothouse Flowers	Hothouse Flowers	Hothouse Flowers
2.	Radiohead	Radiohead	Radiohead
3.	Sinéad O'Connor	Sinéad O'Connor	Sinéad O'Connor
4.	Madonna	Coldplay	Coldplay
5.	Coldplay	Bruce Springsteen	Talking Heads
6.	Elvis Presley	Pearl Jam	Bruce Springsteen

Table 6.11. Examples of most related artists using the baseline method and the two alternatives p_{lin} (3.3) and p_{sqrt} (3.4).

[Genre] artists like [Artist]
[Genre] artists such as [Artist]
[Genre] artists for example [Artist]
[Artist] and other [Genre] artists

Table 6.12. Four of the patterns for the artist-genre relation. In the other patterns, *artists* is respectively replaced with *acts*, *musicians* and *bands*.

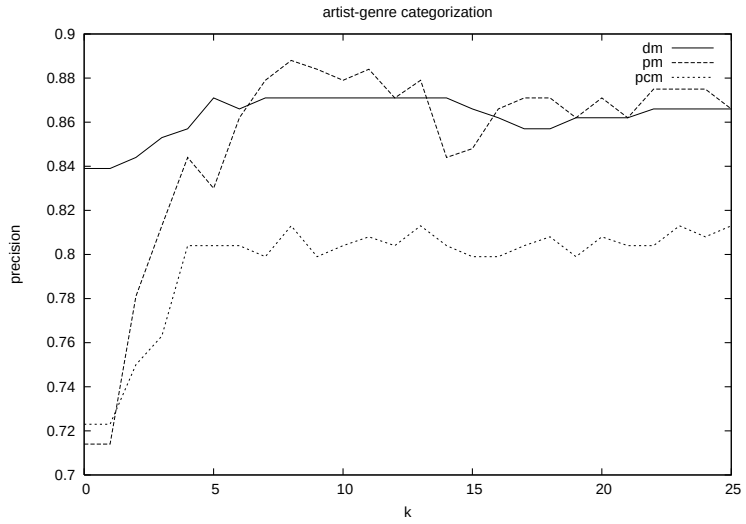

Figure 6.11. Precision for the categorization of the musical artists.

three methods ($k = 0$). We were able to map all artists to a genre. Co-occurrences between genres and artists thus could be found using PCM, PM as well as DM. The latter performs best. With respect to the preliminary mapping, the method with the smallest amount of Google queries performs best.

Using DM only few related artists can be found on the documents visited. Increasing k hence does not effect the performance for the final mapping, as the lists of related artists are small (Figure 6.11). Contrary to especially PCM, large numbers of k do not deteriorate the precision.

The performance of the pattern-based method strongly improves by considering related artists, the best performance is obtained for $k = 8$. All methods perform best for values of k between 5 and 13. The *Rock n' Roll* artists proved to be the most problematic to categorize. The artists in the genres *classical*, *blues* and *jazz* were all correctly categorized with the best scoring settings.

With the supervised music artist clustering method discussed in [Knees et al., 2004] a precision of 87% was obtained using complex machine learning techniques

method	$k = 0$	best	(corresponding k)
PCM	0.71	0.81	(13)
PM	0.72	0.89	(8)
DM	0.84	0.87	(5)

Table 6.13. Precision without related artists and best precision per method.

and a relatively large training set. In [Schedl et al., 2005] a precision of up to 85% precision was obtained using $\mathcal{O}(|I_a|^2)$ queries. We can conclude that our simple and unsupervised method produces similar results. Moreover, we compute a categorization of artists into genres instead of clusters of artists.

We also conducted this experiment using *Last.fm* data. For each artist, we initially selected the genre that gets the highest score. In this case, we thus select the genre that is mentioned as the highest ranked tag. If no genre is mentioned for an artist, initially no genre is assigned.

For the experiment with the *Last.fm* data, we retrieved for each artist of the 224 artist set the list of the (at most) 100 most similar artists. Having obtained an initial mapping between each of the 224 artists and a genre, we use the nearest neighbors to compute a final mapping. Alike the three web-based methods, we compute a majority voting among the initial genre for each artist and its k nearest neighbors using PM, DM and the *Last.fm* data.

We compare the results of the *Last.fm*-based artist categorization with the best two results from [Geleijnse & Korst, 2006c] in Figure 6.12. For the method DM co-occurrences between artists and genres within full web documents are used to compute the initial mapping. To compute artist similarity using DM, we use co-occurrences of artist names within documents. The method PM uses co-occurrences within phrases that express the relations of interest.

The results for artist categorization using the *Last.fm* data are similar to the ones gained using web-data collected with a search engine. The results for *Last.fm* are best when incorporating the tags of the 3 nearest neighbors of every artist. Since an average number of 14 similar artists (out of the set of 224) is identified, the performance deteriorates for larger values of k.

It is notable that for all three methods most misclassifications were made in the *Folk*, *Heavy* and *Rock 'n Roll* genres, where often the genre *Indie/Alternative* was assigned to the artist.

When we classify the artists using the *Last.fm* data after track filtering (see page 123), the initial mapping ($k = 0$) improves slightly as *Chubby Checker* is now

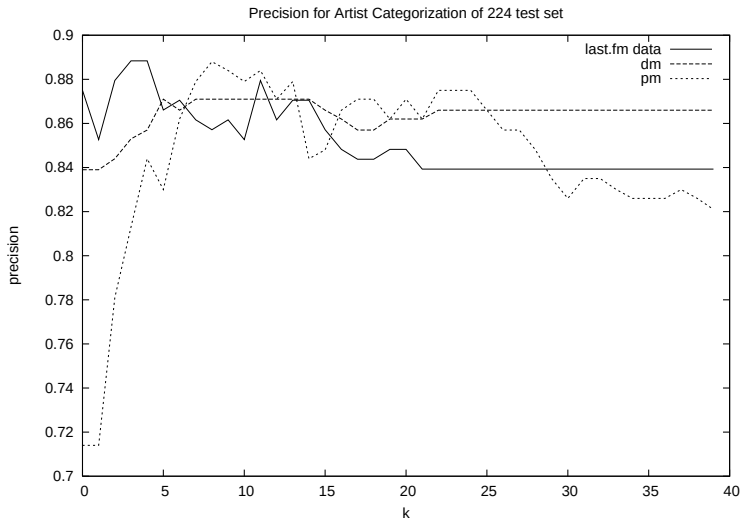

Figure 6.12. Precision of the 224 artist categorization for k-NN using Last.fm and the two best web-based methods.

correctly classified. For values of k larger than 1, the performance using the track filtered data is equal to the one using either the raw or the normalized *Last.fm* data.

As the results of the genre categorization using the *Last.fm* data are equally good as those gained with the best methods using arbitrary web-data, we conclude that DM and PM are reliable methods for this classification task. We also observe that there is no complete overlap with the data extracted from *Last.fm* and the ground truth composed by experts in the field. This on the one hand gives confidence in our methods, but on the other hand raises questions on the fact that not all artist-genre combinations are recognized by the general public. We therefore investigate the use of *Last.fm* data as a ground truth in the last part of this section.

Categorizing Painters into Movements

For this experiment, we constructed a list of painters I_a and a list of movements I_g in art using Wikipedia and map the two. From Wikipedia we extracted a set I_a of 1,280 well-known painters from the article *List of painters* and a set I_g of 77 movements in art from *List of art movements*[9]. We tested the performance of the algorithm on the subset of 160 painters who could be extracted from the Wikipedia pages describing movements (e.g. from the page on *Abstract Expressionism*). The other 1,120 painters are either not mentioned on the pages describing styles or are mentioned on more than one page. However, when computing similarities between the painters, we take all 1,280 painters into account. For the elements of I_g in this

[9] www.wikipedia.org Both pages visited in April 2006.

[Painter] synthetic [Movement]	*[Painter] [Movement]*
[Movement] artist [Painter]	*[Movement] [Painter]*
[Painter] and other [Movement]	*[Painter] express [Movement]*
[Painter] and [Movement]	*[Painter] of the [Movement]*
[Painter] tog initiativ til [Movement]	*[Painter] uit de [Movement]*
[Painter] experimenting with [Movement]	*[Painter] and the [Movement]*
[Painter] surrealism [Movement]	*[Painter] arte [Movement]*

Table 6.14. Best scoring learned patterns for painter - movement relation.

	PAINTER-MOVEMENT		
method	$k = 0$	best	(corresp. k)
PCM	0.35	0.35	(0)
PM	0.54	0.64	(18)
DM	0.65	0.81	(20)
PM-STEMMING	0.53	0.62	(28)

Table 6.15. Precision without related instances and best precision per method.

test no synonyms were added. For fairness, we excluded pages from the domain *wikipedia.org* in the search queries.

For PM, we selected learned patterns for the mapping between the elements in I_a and I_g. For learning, we used instance-pairs outside the test set. The best scoring patterns can be found in Table 6.14. For the relation between the instances in I_a, these patterns found were mostly enumeration patterns, e.g. *"including b and"*. The complete details of both experiments and the patterns used in PM can be found on the web page[10]. Due to the rareness of some of the painters and names of movements, we did not use any additional terms in the queries for DM or PCM.

In Table 6.15 the performance of the initial mapping m' can be found for the three methods ($k = 0$). The experiments show that in general the use of related instances improves the categorization (see Table 6.15 and Figure 6.13). It shows again that the methods with the lowest Google Complexity thus PM and DM perform better than PCM.

Although in the painter-movement experiment the number of categories identified (77) is much larger than in the previous experiment (16), the performance

[10] `http://gijsg.dse.nl/webconmine/`

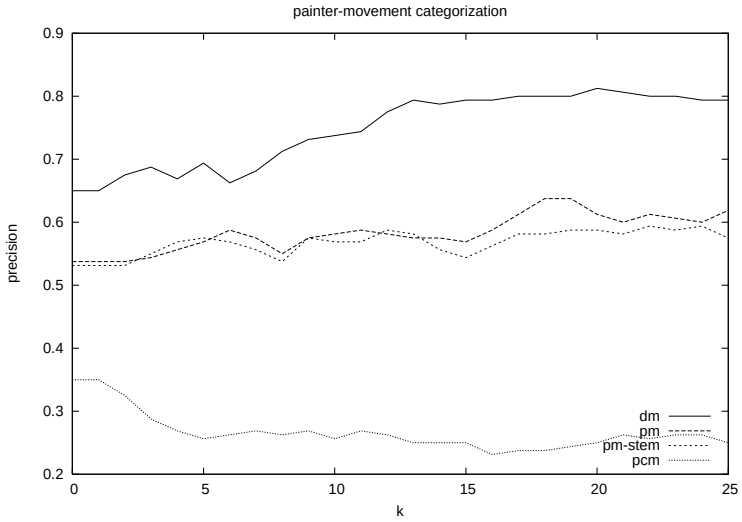

Figure 6.13. Precision for categorization of the painters.

of PM and especially DM is still good. The results of PCM indicate that when the precision of the intermediate mapping is low (35%), the use of related instances does not improve the results. In this experiment we even observe a deterioration of the performance. Here DM clearly outperforms PM. This can be explained by the fact that using PM considerably less painter-movement pairs could be extracted. We expected the recall of PM to increase when applying stemming on the names of movements and the texts extracted [Porter, 1980]. Although the number of pairs extracted slightly increases, the precision does not improve (Table 6.15).

6.4.3 Tagging Instances

In this subsection, we focus on two case-studies on the tagging of instances related to the methods described in Section 6.2.3. We compare the extracted lists of tags with ground truth extracted from a social website. No previous work is known to us in this field. We therefore present two exploratory studies in the automatic tagging of instances. In the first experiment, we tag the set of 224 artists and evaluate the tagging using *Last.fm*. The second experiment focusses on books, where the results are compared with data from *LibraryThing.com*.

Tagging Musical Artists

In this experiment, we focus on the *tagging* of the 224 artists as done by the *Last.fm* community using the method described in Section 6.3. Using a large set of artists, we select the 248 most frequently applied tags after the normalization procedure[11].

[11]The list of tags used can be found at `http://gijsg.dse.nl/tags224.html`

We investigate whether our method is well suited to label the 224 artists and compare the results with the tags as applied by the *Last.fm* users.

The previous experiments showed that PM was the most successful alternative to identify artist similarities, while DM outperformed PM with respect to the labeling of artists with genre names. We hence use DM to find the co-occurrences between artist names and tags and reuse the results from PM to identify the artist similarities. For fairness, the pages from *Last.fm* and *Audiocrobbler.com* are excluded from the search results.

Per artist in the test set, an average of 79 tags was identified using DM. All tags in the test set were linked to at least one artist, however not for all tag/artist combinations a score could be identified, as not all artists are related to one another.

We compare the computed ranking of the tags for the artists with a normalized ranking as identified by the *Last.fm* users as described in Section 6.3. For instance, the terms *'Rocker'* and *'rock'* have the same normalized form.

We evaluate the computed rankings for the different values of w as follows. We first evaluate the precision and recall for the highest ranked tags and secondly compute Spearman's rank correlation between the computed ranking and the one from *Last.fm*.

Precision and Recall. We selected the set S_n of the top n tags for artist i in the ground truth (i.e. the normalized *Last.fm* data) and evaluated precision p and recall r of the computed ordered list $\mathcal{L}_m$ of the m most applicable tags for i.

$$p = \frac{|S_n \bigcap \mathcal{L}_m|}{|\mathcal{L}_m|} \quad \text{and} \quad r = \frac{|S_n \bigcap \mathcal{L}_m|}{|S_n|}$$

The average recall and precision for the computed 25 highest ranked tags (i.e. $m = 25$) compared with the 25 highest ranked tags by the *Last.fm* (i.e. $n = 25$) is given in Figure 6.14. For all values of w the precision is marginally larger than recall, as we found less than 25 tags for few of the artists. We note that for the given set of tags random precision and recall are both 0.10.

For the given settings, we hence obtain precision and recall rates between 0.25 and 0.3. For $w = 0.25$ we obtained the best results. Hence, we again observe that the use of artist similarities improves the labeling of the artists.

For $w = 0.25$ we compute the average precision and recall of the top n *Last.fm* tags by repeatedly increasing m from 1 to 100. The results for various values of n can be found in Figure 6.15.

Ranking. We also evaluate the ranking itself, hence the correlation between the ranking of tag t_i in the ground truth $g_a(t_i)$ and the computed ranking $r_a(t_i)$. For a given artist, we focus on the ranking of the tags that are both in the ground truth data and in the computed list.

The average Correlation Coefficient ρ per w for the 224 artists is given in Fig-

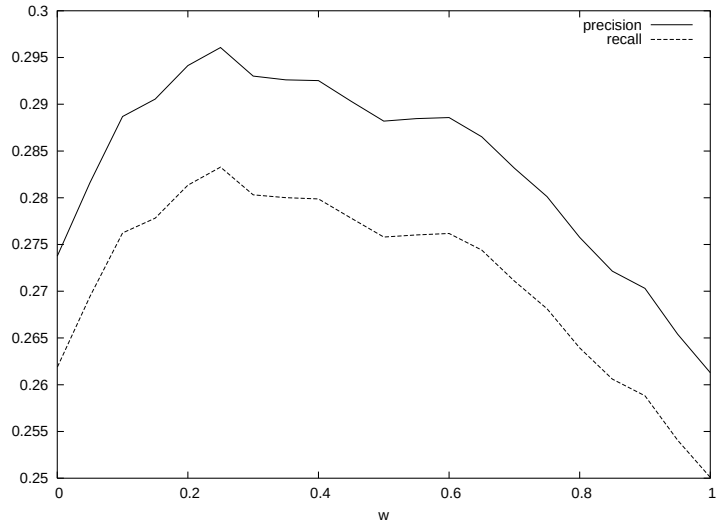

Figure 6.14. Precision and Recall for the 25 best scoring computed tags with respect to the 25 best scoring normalized Last.fm tags for the 224 artists.

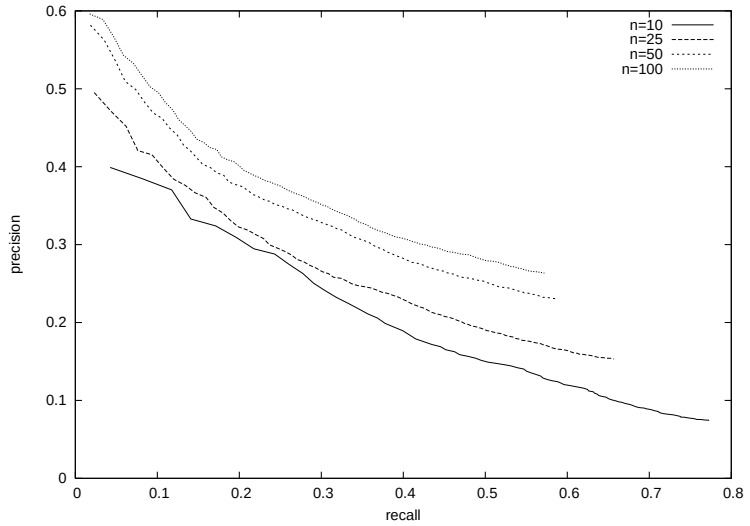

Figure 6.15. Precision and Recall for the n best scoring computed tags with respect to the 25 best scoring normalized Last.fm tags for the 224 artists with $w = 0.25$.

ure 6.16. The correlation is indeed positive – but weak – for all values of w. We note that the value for ρ is slightly lower for values of w approaching both 0 and 1.

The results of the labeling of artists with the tags as applied by *Last.fm* users

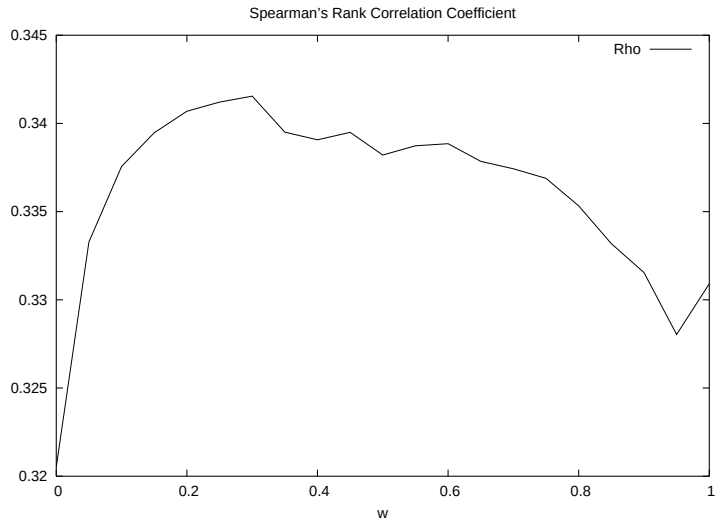

Figure 6.16. Spearman's correlation coefficient between the 224 artist tagging and the Last.fm ground truth.

are modest. Given the difficulty of the task and the nature of the ground truth, we are nevertheless encouraged by the results.

We observe that some frequently applied tags occur infrequently in web texts (e.g. *'i want to hear everything streamable by them'*). Such tags were rarely identified in the texts on the web. On the other hand, among the best scoring tags we find terms that seem less descriptive but often occur on the web, for example *good*, *hot* and *fun*.

Tagging Books

In this second experiment, we focus on books and their tags. Using the social website *LibraryThing.com*, we create a ground truth for the 500 most popular books on this website (Table 6.16 gives the top 25 at the moment of conducting the experiment). After normalization, we reduced the size of the ground truth set of tags to 286. The book titles have been slightly simplified by removing the text after the colon (e.g. in *Animal farm : a fairy story*).

As two author-title combinations are less likely to co-occur within a sentence, we gather the co-occurrences scores for the books in I_a using DM. We query the book title and the name of the author and gather the (at most) 100 resulting documents. Again, the pages of the evaluation website are excluded. To identify co-occurrences, we scan the documents only for the titles of the other books. The identification of co-occurrences between tags and books is done is a similar fashion.

1.	Harry Potter and the Sorcerer's Stone by J.K. Rowling (21,415)
2.	Harry Potter and the Half-Blood Prince by J.K. Rowling (20,650)
3.	Harry Potter and the Order of the Phoenix by J.K. Rowling (19,510)
4.	Harry Potter and the Goblet of Fire by J.K. Rowling (18,658)
5.	Harry Potter and the Chamber of Secrets by J.K. Rowling (18,638)
6.	Harry Potter and the Prisoner of Azkaban by J.K. Rowling (18,567)
7.	The Da Vinci code by Dan Brown (16,013)
8.	The Hobbit by J.R.R. Tolkien (14,538)
9.	1984 by George Orwell (13,655)
10.	The Catcher in the Rye by J.D. Salinger (13,363)
11.	Pride and prejudice by Jane Austen (12,813)
12.	To Kill a Mockingbird by Harper Lee (11,890)
13.	The Great Gatsby by F. Scott Fitzgerald (11,331)
14.	The Lord of the Rings by J.R.R. Tolkien (10,572)
15.	Jane Eyre by Charlotte Bronte (9,847)
16.	The Curious Incident of the Dog in the Night-Time by Mark Haddon (9,526)
17.	Brave New World by Aldous Huxley (9,142)
18.	Life of Pi : a novel by Yann Martel (9,071)
19.	Animal Farm : a fairy story by George Orwell (8,967)
20.	Angels & Demons by Dan Brown (8,799)

Table 6.16. The taste of the crowds: the most popular books among the LibraryThing community in August 2007. The figures between parentheses reflect the number of people claiming to own the book.

Spearman's rank correlation coefficient is given in Figure 6.17. It shows that contrary to the previous experiments the use of related books has a negative effect on the correlation with the ground truth. Again, the correlation is positive, but weak, with best coefficients around 0.3.

Using $w = 1$, we also computed the average precision and recall for the top 25 tags in the ground truth set, see Figure 6.18. These results are also comparable with the *Last.fm* experiment.

The results for the tagging experiments are open to improvement. Mika [2007] proposes to compute a semantic distance between tags. In future work, such an approach can be used both to identify a 'cleaner' ground truth and to identify synonyms of tags. The use of learned synonyms may improve the performance as many tags occur infrequently in unstructured sources on the web. Hence, the assumption that instances can be linked to tags using occurrences of pairs of the two

fiction	series
classic	science fiction
novel	english
paperback	british
literature	american literature
20th century	sf
Favorites	Contemporary Fiction
American	Humor
fantasy	contemporary
hardcover	1001 books

Table 6.17. The 20 most frequently applied tags on *LibraryThing.com*.

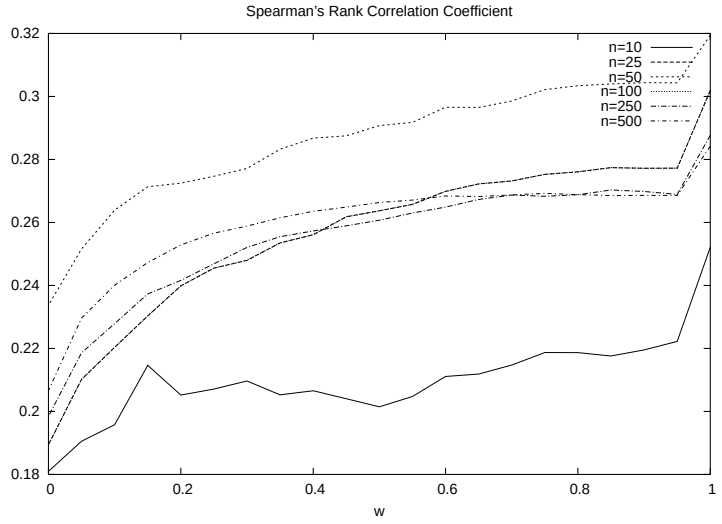

Figure 6.17. Spearman's correlation coefficient between the computed tags and the LibraryThing ground truth.

in texts does not hold for these cases.

Future work should therefore focus on the identification of formulations of tags in unstructured texts. Using an annotated training set of artists and tags we can learn such formulations. Moreover, currently we assume the tags in the set I_g to be given. We are interested to exploit methods to learn new terms for the set I_g. This can for instance be done with the *tf·idf*-approach [Knees et al., 2004; Manning & Schütze, 1999].

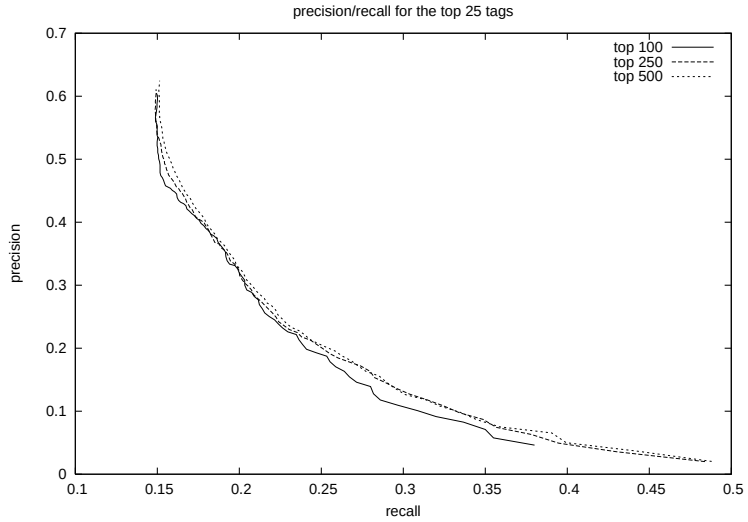

Figure 6.18. Average precision and recall for the top 25 tags for the books in LibraryThing.

6.5 Conclusions

In this chapter we have focussed on the extraction of information from the web that is not present as such. By combining information from various sources, we have created characterizations of instances based on the collective knowledge on the web. We have argued that such collaborative characterizations are often more informative than the ones provided by experts.

We presented a simple method to identify relatedness among instances in one class (e.g. artists) using co-occurrences found with web information extraction methods. With similar techniques we find the most applicable category (e.g. a genre) for each instance in a given set. The last problem addressed focuses on the identification of a ranked list of tags applicable to an instance, based on the information available on the web.

The experimental results for the identification of the relatedness among instances and the categorization of instances are both convincing. We have shown that the use of the identified relatedness improves the categorization.

In the last part of this chapter, we focused on a novel task in web information extraction. We identified an ordered list of tags for a given set of instances. The computed lists were compared with a ground truth for a social website (e.g. *Last.fm*). Although the results of both tagging experiments are modest, we are encouraged given the difficulty of the task. As we have sketched directions for improvement, we hope that this work inspires to continue the research on the tagging

of instances using web information extraction.

To evaluate the methods presented, we compared the output of the method with benchmarks composed by experts as well as sets collected from social web sites. Although the use of these benchmark sets gives valuable insights in the quality of the output, it will be interesting to analyze the use of extracted community data in applications. For example, the questions remain how the extracted information can contribute to a recommender system, and how the performance compares with the use of information gathered from other sources.

7

Conclusions

Intelligent applications can benefit from the collective knowledge of the internet community as to be found on the web. However, the vast majority of the information on the web is represented in a human-friendly format using natural language texts. Such information in natural language texts is not machine interpretable.

In this thesis, we presented approaches to find, extract and structure information from natural language texts on the web. Such structured information can be machine interpreted and hence be used in intelligent applications.

Information extraction is the task of identifying instances of classes and their relations in a text corpus. We adopted the concept of ontology to model the information demand. The information extraction problem is translated into an ontology population problem.

We proposed a simple ontology population method using patterns. Patterns are commonly occurring phrases that are typically used to express a given relation. Patterns are accompanied by placeholders for instances, for example *[City] is the capital of [Country]*. We combine such patterns and known instances into search engine queries (e.g. *Amsterdam is the capital of*). Subsequently, we extract instances and relations from the retrieved documents. The use of the constructed queries serves two goals. On the one hand, it shows to be an effective mechanism to access highly relevant texts, while on the other hand we can identify relations between instances.

State-of-the-art search engines have restrictions on the accepted numbers of automated queries per day. We hence analyze our methods by their *Google Complexity*.

After having discussed a general approach to populate an ontology in Chapter 2, we focused on two subproblems: the identification of effective patterns and the recognition of the instances of the defined classes in texts. The presented approach contains bootstrapping mechanisms, as learned instances and patterns are used to formulate new search engine queries.

We make use of the redundancy of information on the web. Many statements (i.e. subject – relation – object triples) can be found on various pages using diverse formulations. We use this characteristic of the web as a corpus to filter out erroneously extracted data. The more a statement is identified on the web, the higher the confidence in its correctness.

We have argued that precision of a pattern is not the only criterion for a pattern to be *effective*. The patterns identified in the case-studies are recognizable formulations of the corresponding relation.

To recognize instances in web texts, and more specifically in snippets, we presented two alternative approaches. On the one hand, we can identify instances using a knowledge-oriented approach, where regular expressions are created to match instances of a given class. On the other hand, we presented a data-oriented approach. Given a set of known instances, a collection of texts is annotated. A classifier uses the annotated texts as training set in order to recognize new instances in the other texts.

The methods discussed are illustrated with several case-studies. In the thesis, we focused on three tasks in web information extraction. In order to benchmark our method, we extract *facts* from the web. The second part focuses on two applications of web information extraction, while the last part of the thesis focuses on the discovery of information. By combining content of multiple documents, we create community-based descriptions for instances such as books, painters and popular artists.

In the case-studies in Chapter 4 we show that we can precisely identify relation instances using the pattern-based approach. Both with manually constructed patterns as well as with learned patterns good results were achieved in the studied cases. The use of the pattern-instance combinations in queries is an effective approach to access relevant search results. We have shown that the redundancy of information on the web enables us to precisely identify instances using the rule-based approach. For the data-oriented approach, the use of a large and representative set of known instances is crucial.

In Chapter 5, we discussed two applications of web information extraction. In the first part of the chapter, we presented a method to map arbitrary terms to a

semantically related term in a given thesaurus. As the thesaurus is used to index a collection, the access to this collection is improved.

For those interested in the retrieval of the lyrics of a given song, we developed an application that extracts versions of the lyrics from the web and combines them into a most plausible version. The results of the experiments are convincing.

Chapter 6 focuses on community-based data. We are interested in the identification of characterizations of instances like musical artists and painters, based on the *wisdom of the crowds* as expressed on the web. We have discussed three tasks in the identification of community-based data: the identification of the perceived relatedness between instances, the categorization of instances and the tagging of instances. The experimental results for the identification of the relatedness among instances and the categorization of instances are both convincing. We have shown that the use of the identified relatedness improves the categorization. With respect to the tagging of instances using texts on the web, no comparable previous work is known. Although the results of both tagging experiments are modest, we are encouraged given the difficulty of the task. We have developed an algorithm to generate a dynamic ground truth to evaluate the tagging of instances, which facilitates the challenging research beyond the categorization of instances.

In this thesis we have shown that we can extract information from the web in a simple and efficient manner. By combining and structuring information from the Web, we create a valuable surplus to the knowledge already available. The identification of collective knowledge and opinions is perhaps more interesting than collecting plain facts, which often can be mined from semi-structured sources.

As the web is an ever growing corpus of texts, and intelligent applications can benefit from the extracted information, the future for web information extraction is bright and promising.

Bibliography

Abney, S., Collins, M., & Singhal, A. [2000]. Answer extraction. In *In proceedings of the sixth applied natural language processing conference* (pp. 296–301).

Agichtein, E., & Gravano, L. [2000]. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM international conference on digital libraries.*

Aleksovski, Z., Kate, W. ten, & Harmelen, F. van. [2006]. Approximate semantic matching of music classes on the internet. In *Intelligent algorithms in ambient and biomedical computing* (pp. 133 – 148). Springer.

Aucouturier, J.-J., & Pachet, F. [2003]. Representing musical genre: A state of the art. *Journal of New Music Research*, *32*(1), 83 – 93.

Auer, S., Bizer, C., Lehmann, J., Kobilarov, G., Cyganiak, R., & Ives, Z. [2007]. Dbpedia: A nucleus for a web of open data. In *Proceedings of the sixth international semantic web conference and the second asian semantic web conference (iswc + aswc 2007)* (pp. 715–728). Busan, Korea.

Balog, K., Mishne, G., & De Rijke, M. [2006]. Why are they excited? identifying and explaining spikes in blog mood levels. In *Conference companion of the 11th meeting of the european chapter of the association for computational linguistics (eacl 2006)* (pp. 207 – 210). Trento, Italy.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., & Etzioni, O. [2007]. Open information extraction from the web. In *Proceedings of the 20th international joint conference on artificial intelligence (ijcai 2007)* (p. 2670-2676). Hyderabad, India.

Basili, R., Serafini, A., & Stellato, A. [2004]. Classification of musical genre: a machine learning approach. In *Proceedings of 5th international conference on music information retrieval (ismir'04).*

Brill, E. [1992]. A simple rule-based part-of-speech tagger. In *Proceedings of the third conference on applied natural language processing (ANLP'92)* (pp. 152 – 155). Trento, Italy.

Brin, S., & Page, L. [1998]. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, *30*(1 – 7), 107–117.

Brothwick, A. [1999]. *A maximum entropy approach to named entity recognition*. Unpublished doctoral dissertation, New York University.

Bunescu, R., & Mooney, R. J. [2007]. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th annual meeting of the association for computational linguistics (acl 2007).* Prague, Czech Republic.

Cafarella, M. J., Downey, D., Soderland, S., & Etzioni, O. [2005]. Knowitnow: Fast, scalable information extraction from the web. In *Proceedings of human language technology conference and conference on empirical methods in nlp* (pp. 563 – 570). Vancouver, Canada.

Caraballo, S. A. [1999]. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th annual meeting of the association for computational linguistics on computational linguistics (acl'99)* (pp. 120–126). College Park, Maryland, USA.

Chang, C.-H., Kayed, M., Girgis, M. R., & Shaalan, K. F. [2006]. A survey of web information systems. *IEEE Transactions on Knowledge and Data Engineering*, *18*(10), 1411 – 1428.

Chen, K., Gao, S., Zhu, Y., & Sun, Q. [2006]. Popular song and lyrics synchronization and its application to music information retrieval. In *Proceedings of the thirteenth annual multimedia computing and networking conference 2006.* San Jose, CA.

Chinchor, N. A. (Ed.). [1995]. *Proceedings of the sixth message understanding conference (muc-6)*. Columbia, Maryland: Morgan Kaufmann.

Chinchor, N. A. (Ed.). [1998]. *Proceedings of the seventh message understanding conference (muc-7)*. Fairfax, Virginia: Morgan Kaufmann.

Cilibrasi, R., & Vitanyi, P. [2007]. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Management*, *19*(3), 370 – 383.

Cimiano, P. [2006]. *Ontology learning and population from text: algorithms, evaluation and applications*. Heidelberg, Germany: Springer.

Cimiano, P., & Staab, S. [2004]. Learning by Googling. *SIGKDD Explorations Newsletter*, *6*(2), 24 – 33.

Ciravegna, F., Chapman, S., Dingli, A., & Wilks, Y. [2004]. Learning to harvest information for the semantic web. In *Proceedings of the 1st european semantic web symposium (esws-2004).*

Collins, M. [2002]. Ranking algorithms for named entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th annual meeting of the association for computational linguistics (acl 2002)* (pp. 489–496). Philadelphia, PA.

Corpet, F. [1988]. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, *16*(22), 10881 - 10890.

Crescenzi, V., & Mecca, G. [2004]. Automatic information extraction from large websites. *Journal of the ACM*, *51*(5), 731 – 779.

Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. [2002]. Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of the 40th annual meeting of the association for computational linguistics (acl 2002).* Philadelphia, PA.

Daelemans, W., & Bosch, A. van den. [2005]. *Memory-based language processing*. Cambridge University Press.

Dave, D., & Lawrence, S. [2003]. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of the twelfth international world wide web conference (www2003)* (pp. 519 – 528). Budapest, Hungary: ACM Press.

De Boer, V., Someren, M. van, & Wielinga, B. [2006]. Extracting art style periods from the web. In *Proceedings of the eswc 2006 workshop on mastering the gap: From information extraction to semantic representation.* Budva, Montenegro.

De Boer, V., Someren, M. van, & Wielinga, B. J. [2007]. A redundancy-based method for the extraction of relation instances from the web. *International Journal of Human-Computer Studies*, *65*(9), 816 – 831.

De Bruijn, N. [1968]. *Automath, a language for mathematics* (Tech. Rep. No. TH-report 68-WSK-05). Technische Hogeschool Eindhoven, onderafdeling der wiskunde. (http://www.win.tue.nl/automath/archive/pdf/aut001.pdf)

De Meulder, F., & Daelemans, W. [2003]. Memory-based named entity recognition using unannotated data. In *Proceedings of the seventh conference on natural language learning at hlt-naacl 2003* (pp. 208–211). Edmonton, Canada: Association for Computational Linguistics.

Downey, D., Broadhead, M., & Etzioni, O. [2007]. Locating Complex Named Entities in Web Text. In *Proceedings of the twentieth international joint conference on artificial intelligence (ijcai'07).* Hyderabad, India.

Downey, D., Etzioni, O., & Soderland, S. [2005]. A probabilistic model of redundancy in information extraction. In *19th international joint conference on artificial intelligence (ijcai'05)* (pp. 1034 – 1041). Edinburgh, UK.

Duda, R. O., Hart, P. E., & Stork, D. G. [2000]. *Pattern classification*. Wiley-Interscience Publication.

Dumais, S., Banko, M., Brill, E., Lin, J., & Ng, A. [2002]. Web question answering: is more always better? In *Sigir '02: Proceedings of the 25th annual international acm sigir conference on research and development in information retrieval* (pp. 291–298). New York, NY, USA: ACM Press.

Ellis, D. P., Whitman, B., Berenzweig, A., & Lawrence, S. [2002]. The quest for ground truth in musical artist similarity. In *Proceedings of the third international conference on music information retrieval (ismir'02)* (pp. 170 – 177). Paris, France.

Etzioni, O., Cafarella, M. J., Downey, D., Popescu, A., Shaked, T., Soderland, S., et al. [2005]. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, *165*(1), 91 – 134.

Fellbaum, C. (Ed.). [1998]. *Wordnet: An electronic lexical database*. Cambridge, MA: MIT Press.

Finkel, J. R., Grenager, T., & Manning, C. D. [2005]. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl 2005).* Ann Arbor ,MI.

Fleischman, M., & Hovy, E. [2002]. Fine grained classification of named entities. In *Proceedings of the 19th international conference on computational linguistics* (pp. 1–7). Morristown, NJ, USA: Association for Computational Linguistics.

Frantzi, K., Ananiado, S., & Mima, H. [2000]. Automatic recognition of multiword terms: the c-value/nc-value method. *International Journal on Digital Libraries*, *3*, 115 – 130.

Geleijnse, G. [2004]. *Comparing two user friendly languages for mathematics: Wtt and mizar*. Unpublished master's thesis, Eindhoven University of Technology.

Geleijnse, G., & Korst, J. [2006a]. Efficient lyrics extraction from the web. In R. Dannenberg, K. Lemström, & A. Tindale (Eds.), *Proceedings of the seventh international conference on music information retrieval (ismir'06)* (pp. 371 – 372). Victoria, Canada: University of Victoria.

Geleijnse, G., & Korst, J. [2006b]. Tagging artists using co-occurrences on the web. In W. Verhaegh, E. Aarts, W. ten Kate, J. Korst, & S. Pauws (Eds.), *Proceedings third philips symposium on intelligent algorithms (soia 2006)* (pp. 171 – 182). Eindhoven, the Netherlands.

Geleijnse, G., & Korst, J. [2006c]. Web-based artist categorization. In R. Dannenberg, K. Lemström, & A. Tindale (Eds.), *Proceedings of the seventh international conference on music information retrieval (ismir'06)* (pp. 266 – 271). Victoria, Canada: University of Victoria.

Geleijnse, G., & Korst, J. [2007]. Improving the accessibility of a thesaurus-based catalog by web content mining. In *Proceedings of the first international workshop on cultural heritage on the semantic web.* Busan, Korea: CEUR-WS.org.

Geleijnse, G., Schedl, M., & Knees, P. [2007]. The quest for ground truth in musical artist tagging in the social web era. In S. Dixon, D. Bainbridge, & R. Typke (Eds.), *Proceedings of the eighth international conference on music information retrieval (ismir'07)* (pp. 525 – 530). Vienna, Austria: Austrian Computer Society.

Geleijnse, G., Sekulovski, D., Korst, J., Kater, B., Pauws, S., & Vignoli, F. [2008]. Enriching music with synchronized lyrics, images and colored lights. In *First international conference on ambient media and systems (ambi-sys 2008).* Quebec, QC, Canada.

Giles, J. [2005]. Internet encyclopaedias go head to head. *Nature*, *438*(15), 900 – 901.

Gligorov, R., Aleksovski, Z., Kate, W. ten, & Harmelen, F. van. [2007]. Using Google Distance to weight approximate ontology matches. In *Proceedings of the 16th international conference on world wide web (www2007).* Banff, Canada.

Gruber, T. [1995]. Toward principles for the design of ontologies used for knowledge sharing. *International Journal Human-Computer Studies*, *43*, 907 – 928.

Gusfield, D. [1997]. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge, UK: Cambridge University Press.

Hearst, M. [1992]. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on computational linguistics* (pp. 539 – 545). Nantes, France.

Hearst, M. [1998]. Automated discovery of wordnet relations. In C. Fellbaum (Ed.), *Wordnet: An electronic lexical database.* Cambridge, MA: MIT Press.

IJzereef, L. [2004]. *Automatische extractie van hyponiemrelaties uit grote textcorpora*. Unpublished master's thesis, Rijksuniversiteit Groningen, Groningen, the Netherlands.

Iskander, D., Wang, Y., Kan, M.-Y., & Li, H. [2006]. Syllabic level automatic synchronization of music signals and text lyrics. In *Multimedia '06: Proceedings of the 14th annual acm international conference on multimedia* (pp. 659 – 662). Santa Barbara, CA.

Jin, Y., Matsuo, Y., & Ishizuka, M. [2006]. Extracting a social network among entities by web mining. In *Proceedings of the iswc 2006 workshop on web content mining with human language technologies (webconmine).* Athens, GA.

Keen, A. [2007]. *The cult of the amateur: how today's internet is killing our culture*. Bantam Dell Pub Group.

Kendall, M. [1975]. *Rank correlation methods, fourth edition*. Charles Griffin & Co. Ltd.

Kim, S.-M., & Hovy, E. [2004]. Determining the sentiment of opinions. In *Coling '04: Proceedings of the 20th international conference on computational linguistics* (p. 1367). Morristown, NJ, USA: Association for Computational Linguistics.

Kiss, T., & Strunk, J. [2006]. Unsupervised multilingual sentence boundary detec-

tion. *Computational Linguistics*, *32*(4), 485–525.

Kleedorfer, F. [2008]. *Automatic topic detection in song lyrics*. Unpublished master's thesis, Technische Universität Wien.

Knees, P., Pampalk, E., & Widmer, G. [2004]. Artist classification with web-based data. In *Proceedings of 5th international conference on music information retrieval (ismir'04)* (pp. 517 – 524). Barcelona, Spain.

Knees, P., Schedl, M., & Widmer, G. [2005]. Multiple Lyrics Alignment: Automatic Retrieval of Song Lyrics. In *Proceedings of 6th international conference on music information retrieval (ismir'05)* (pp. 564 – 569). London, UK.

Korst, J., & Geleijnse, G. [2006]. Efficient lyrics retrieval and alignment. In W. Verhaegh, E. Aarts, W. ten Kate, J. Korst, & S. Pauws (Eds.), *Proceedings third philips symposium on intelligent algorithms (soia 2006)* (pp. 205 – 218). Eindhoven, the Netherlands.

Kraaij, W. [2004]. *Variations on language modeling for information retrieval*. Unpublished doctoral dissertation, Twente University.

Li, T., Ogihara, M., & Li, Q. [2003]. A comparative study on content-based music genre classification. In *Sigir '03: Proceedings of the 26th annual international acm sigir conference on research and development in informaion retrieval* (pp. 282 – 289). New York, NY, USA: ACM Press.

Lin, D. [1998]. Dependency based evaluation of minipar. In *Proceedings of the workshop on the evaluation of parsing systems at the first international conference on language resources and evaluation.* Granada, Spain.

Logan, B., Kositsky, A., & Moreno, P. [2004]. Semantic analysis of song lyrics. In *Ieee international conference on multimedia and expo (icme).* Taipei, Taiwan.

Mahedero, J. P. G., Martínez, A., Cano, P., Koppenberger, M., & Gouyon, F. [2005]. Natural language processing of lyrics. In *Multimedia '05: Proceedings of the 13th annual acm international conference on multimedia* (pp. 475 – 478). New York, NY, USA: ACM Press.

Malaisé, V., Isaac, A., Gazendam, L., & Brugman, H. [2007]. Anchoring dutch cultural heritage thesauri to wordnet: two case studies. In *Proceedings of the workshop on language technology for cultural heritage data (latech 2007)* (p. 57 - 64). Prague, Czech Republic.

Manning, C. D., & Schütze, H. [1999]. *Foundations of statistical natural language processing*. Cambridge, Massachusetts: The MIT Press.

Marneffe, M.-C. de, MacCartney, B., & Manning, C. D. [2006]. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the ieee / acl 2006 workshop on spoken language technology.*

McCallum, A. [2005]. Information extraction: distilling structured data from

unstructured text. *ACM Queue*, *3*(9), 48–57.

McDowell, L., & Cafarella, M. J. [2006]. Ontology-driven information extraction with ontosyphon. In *Proceedings of the 5th international semantic web conference (iswc 2006)* (Vol. 4273, pp. 428 – 444). Athens, GA: Springer.

McKay, C., & Fujinaga, I. [2006]. Musical genre classification: Is it worth pursuing and how can it be improved? In *Proceedings of the seventh international conference on music information retrieval (ismir'06)* (pp. 101 – 106). Victoria, Canada.

McKinney, M. F., & Breebaart, J. [2003]. Features for audio and music classification. In *Proceedings of the 4th international conference on music information retrieval.* Baltimore, MD: Johns Hopkins University.

Meilicke, C., Stuckenschmidt, H., & Tamilin, A. [2007]. Repairing ontology mappings. In *Proceedings of the 22nd conference on artificial intelligence (aaai-07).* Vancouver, Canada.

Mika, P. [2007]. Ontologies are us: A unified model of social networks and semantics. *Journal of Web Semantics*, *5*(1), 5 – 15.

Mitchell, T. [1997]. *Machine learning*. McGraw Hill.

Mori, J., Tsujishita, T., Matsuo, Y., & Ishizuka, M. [2006]. Extracting relations in social networks from the web using similarity between collective contexts. In *Proceedings of the 5th international semantic web conference (iswc 2006)* (Vol. 4273, pp. 487 – 500). Athens, GA: Springer.

Navigli, R., & Velardi, P. [2006]. Enriching a formal ontology with a thesaurus: an application in the cultural heritage domain. In *Proceedings of the 2nd workshop on ontology learning and population: Bridging the gap between text and knowledge* (pp. 1–9). Sydney, Australia: Association for Computational Linguistics.

Nederpelt, R., Geuvers, J. H., & De Vrijer, R. C. (Eds.). [2004]. *Selected papers on automath*. Amsterdam, the Netherlands: North-Holland.

Pachet, F., & Cazaly, D. [2000]. A taxonomy of musical genres. In *Content-based multimedia information access conference (riao).* Paris, France.

Pampalk, E., Flexer, A., & Widmer, G. [2005]. Improvements of audio-based music similarity and genre classificaton. In *Proceedings of the sixth international conference on music information retrieval (ISMIR'05)* (pp. 628 – 633). London, UK.

Pang, B., & Lee, L. [2005]. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43th annual meeting of the association for computational linguistics (acl 2005)* (pp. 115–124). Ann Arbor, MI.

Pang, B., Lee, L., & Vaithyanathan, S. [2002]. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 con-*

ference on empirical methods in natural language processing (emnlp) (pp. 79–86).

Pantel, P., & Pennacchiotti, M. [2006]. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of conference on computational linguistics / association for computational linguistics (coling/acl-06)* (pp. 113 – 120). Sydney, Australia.

Pohle, T., Knees, P., Schedl, M., & Widmer, G. [2007]. Building an interactive next-generation artist recommender based on automatically derived higher-level concepts. In *Proceedings of the fifth international workshop on content-based multimedia indexing (cbmi'07).* Bordeaux, France.

Porter, M. F. [1980]. An algorithm for suffix stripping. *Program*, *14*, 130 – 136.

Ravichandran, D., & Hovy, E. [2002]. Learning surface text patterns for a question answering system. In *Proceedings of the 40th annual meeting of the association for computational linguistics (acl 2002)* (pp. 41 – 47). Philadelphia, PA.

Salton, G., & Buckley, C. [1988]. Term-weighting approaches in automatic text retrieval. In *Information processing and management* (pp. 513–523).

Sazedj, P., & Pinto, H. S. [2006]. Factbox - a framework for instantiating ontological relations from text. In *Proceedings of the iswc 2006 workshop on web content mining with human language technologies (webconmine).* Athens, GA.

Scaringella, N., Zoia, G., & Mlynek, D. [2006]. Automatic genre classification of music content. *IEEE Signal Processing Magazine : Special Issue on Semantic Retrieval of Multimedia*, *23*(2), 133 – 141.

Schedl, M., Knees, P., & Widmer, G. [2005]. A Web-Based Approach to Assessing Artist Similarity using Co-Occurrences. In *Proceedings of the fourth international workshop on content-based multimedia indexing (CBMI'05).* Riga, Latvia.

Schedl, M., Pohle, T., Knees, P., & Widmer, G. [2006]. Assigning and visualizing music genres by web-based co-occurrence analysis. In *Proceedings of the seventh international conference on music information retrieval (ismir'06)* (pp. 260 – 265). Victoria, Canada.

Schedl, M., & Widmer, G. [2007]. Automatically Detecting Members and Instrumentation of Music Bands via Web Content Mining. In *Proceedings of the 5th workshop on adaptive multimedia retrieval (AMR'07).* Paris, France.

Schlobach, S., Ahn, D., Rijke, M. de, & Jijkoun, V. [2007]. Data-driven type checking in open domain question answering. *Journal of Applied Logic*, *5*(1), 121–143.

Schutz, A., & Buitelaar, P. [2005]. Relext: A tool for relation extraction from text in ontology extension. In *Proceedings of the fourth international semantic*

web conference (iswc 2005) (p. 593-606). Galway, Ireland.

Sekulovski, D., Geleijnse, G., Kater, B., Korst, J., Pauws, S., & Clout, R. [2008]. Enriching text with images and colored light. In *Proceedings of the is&t/spie 20th annual electronic imaging symposium.* San Jose, CA.

Shchekotykhin, K. M., Jannach, D., Friedrich, G., & Kozeruk, O. [2007]. *Allright: Automatic ontology instantiation from tabular web documents.* Busan, Korea.

Shvaiko, P., & Euzenat, J. [2005]. A survey of schema-based matching approaches. In *Journal on data semantics iv* (Vol. 3730, pp. 146 – 171). Heidelberg, Germany: Springer.

Shvaiko, P., Euzenat, J., Noy, N., Stuckenschmidt, H., Benjamins, R., & Uschold, M. (Eds.). [2006]. *Proceedings of the iswc'06 international workshop on ontology matching*. CEUR-WS Vol-225. (http://www.om2006.ontologymatching.org/)

Sigurd, B., Eeg-Olofsson, M., & Van Weijer, J. [2004]. Word length, sentence length and frequency - zipf revisited. *Studia Linguistica*, *58*(1), 37 - 52.

Smith, M. K., Welty, C., & McGuinness, D. L. [2004]. *Owl web ontology language guide.*

Snow, R., Jurafsky, D., & Ng, A. Y. [2005]. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems 17* (pp. 1297–1304). Cambridge, MA: MIT Press.

Snow, R., Jurafsky, D., & Ng, A. Y. [2006]. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the acl (coling/acl 2006)* (pp. 801–808). Sydney, Australia.

Sumida, A., Torisawa, K., & Shinzato, K. [2006]. Concept-instance relation extraction from simple noun sequences using a full-text search engine. In *Proceedings of the iswc 2006 workshop on web content mining with human language technologies (webconmine).* Athens, GA.

Ter Horst, H. [2005]. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Journal Web Semantics*, *3*(2-3), 79 – 115.

Tiemann, M., & Pauws, S. [2007]. Towards ensemble learning for hybrid music recommendation. In *Proceedings of the 2007 acm conference on recommender systems (recsys 2007)* (p. 177-178). Minneapolis, MN, USA.

Tjong Kim Sang, E., & Hofmann, K. [2007]. Automatic extraction of dutch hypernym-hyponym pairs. In *Proceedings of computational linguistics in the netherlands (clin-17).* Leuven, Belgium.

Tzanetakis, G., & Cook, P. [2002]. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, *10*(5).

Van Assem, M., Menken, M. R., Schreiber, G., Wielemaker, J., & Wielinga, B. [2004, November]. A Method for Converting Thesauri to RDF/OWL. In S. A. McIlraith, D. Plexousakis, & F. van Harmelen (Eds.), *Proceedings of the third international semantic web conference (iswc'04)* (pp. 17–31). Hiroshima, Japan: Springer-Verlag.

Van Hage, W. R., Kolb, H., & Schreiber, G. [2006]. A method for learning part-whole relations. In *Proceedings of the 5th international semantic web conference (iswc 2006)* (Vol. 4273, pp. 723 – 736). Athens, GA: Springer.

Van Rijsbergen, C. J. [1979]. *Information retrieval, 2nd edition*. London, UK: Butterworths.

Verberne, S., Boves, L., Oostdijk, N., & Coppen, P.-A. [2007]. Evaluating discourse-based answer extraction for why-question answering. In *Proceedings of the 30th annual international acm sigir conference on research and development in information retrieval (sigir 2007)* (pp. 735 – 737). Amsterdam, the Netherlands.

Véronis, J. [2006]. *Weblog.* (http://aixtal.blogspot.com)

Voorhees, E. [2004]. Overview of the trec 2004 question answering track. In *Proceedings of the 13th text retrieval conference (trec 2004).* Gaithersburg, Maryland.

Voorhees, E. [2005]. Common evaluation measures. In E. Voorhees (Ed.), *The fourteenth text retrieval conference (trec 2005) proceedings.* NIST.

Wang, L., & Gusfield, D. [1997]. Improved approximation algorithms for tree alignment. *J. Algorithms*, *25*(2), 255 – 273.

Wang, L., & Jiang, T. [2004]. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, *1*(4), 337 – 348.

Wang, L., Jiang, T., & Lawler, E. L. [1996]. Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica*, *16*(3), 302 – 315.

Wang, Y., Kan, M.-Y., Nwe, T. L., Shenoy, A., & Yin, J. [2004]. Lyrically: automatic synchronization of acoustic musical signals and textual lyrics. In *Multimedia '04: Proceedings of the 12th annual acm international conference on multimedia* (pp. 212 – 219). New York, NY.

Whitman, B., & Ellis, D. [2004]. Automatic record reviews. In *Proceedings of 5th international conference on music information retrieval (ismir'04)* (pp. 86 – 93). Barcelona, Spain.

Zadel, M., & Fujinaga, I. [2004]. Web services for music information retrieval. In *Proceedings of 5th international conference on music information retrieval (ismir'04).* Barcelona, Spain.

Zhou, G., & Su, J. [2002]. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th annual meeting of the association for computational linguistics (acl 2002)* (pp. 473 – 480). Philadelphia, PA.

Zong, W., Wu, D., Sun, A., Lim, E.-P., & Goh, D. H.-L. [2005]. On assigning place names to geography related web pages. In *Jcdl '05: Proceedings of the 5th acm/ieee-cs joint conference on digital libraries* (pp. 354–362). New York, NY, USA: ACM.

Publications

The following scientific publications relate to this thesis.

- Geleijnse, G., Korst, J. [2005]. Automatic ontology population by googling. In K. Verbeeck, K. Tuyls, A. Nowé, B. Kuijpers, B. Manderick (Eds.), *Proceedings of the seventeenth belgium-netherlands conference on artificial intelligence (bnaic 2005)* (pp. 120 – 126). Brussels, Belgium: Koninklijke Vlaamse Academie van België voor Wetenschappen en Kunsten.
- Geleijnse, G., Korst, J. [2006a]. Efficient lyrics extraction from the web. In R. Dannenberg, K. Lemström, A. Tindale (Eds.), *Proceedings of the seventh international conference on music information retrieval (ismir'06)* (pp. 371 – 372). Victoria, Canada: University of Victoria.
- Geleijnse, G., Korst, J. [2006b]. Learning effective surface text patterns for information extraction. In *Proceedings of the 18th benelux conference on artificial intelligence (bnaic 2006).* Namur, Belgium.
- Geleijnse, G., Korst, J. [2006c]. Learning effective surface text patterns for information extraction. In *Proceedings of the eacl 2006 workshop on adaptive text extraction and mining (atem 2006)* (pp. 1 – 8). Trento, Italy: Association for Computational Linguistics.
- Geleijnse, G., Korst, J. [2006d]. Tagging artists using co-occurrences on the web. In W. Verhaegh, E. Aarts, W. ten Kate, J. Korst, S. Pauws (Eds.), *Proceedings third philips symposium on intelligent algorithms (soia 2006)* (pp. 171 – 182). Eindhoven, the Netherlands.
- Geleijnse, G., Korst, J. [2006e]. Web-based artist categorization. In R. Dannenberg, K. Lemström, A. Tindale (Eds.), *Proceedings of the seventh international conference on music information retrieval (ismir'06)* (pp. 266 – 271). Victoria, Canada: University of Victoria.
- Geleijnse, G., Korst, J. [2007a]. Creating a Dead Poets Society: Extracting a social network of historical persons from the web. In K. Aberer et al. (Eds.), *Proceedings of the sixth international semantic web conference and the second asian semantic web conference (iswc + aswc 2007), Busan, Korea* (Vol. 4825 of Lecture Notes in Computer Science (LNCS), pp. 156 – 168). Heidelberg, Germany: Springer.

- Geleijnse, G., Korst, J. [2007b]. Improving the accessibility of a thesaurus-based catalog by web content mining. In *Proceedings of the first international workshop on cultural heritage on the semantic web.* Busan, Korea: CEUR-WS.org.
- Geleijnse, G., Korst, J. [2007c]. Tagging musical artists by web content mining. *Journal of Web Semantics*. (submitted)
- Geleijnse, G., Korst, J. [2007d, September]. Tool play live: Dealing with ambiguity in artist similarity mining from the web. In S. Dixon, D. Bainbridge, R. Typke (Eds.), *Proceedings of the eighth international conference on music information retrieval (ismir'07)* (pp. 119 – 120). Vienna, Austria: Austrian Computer Society.
- Geleijnse, G., Korst, J. [2008]. Search engine-based web information extraction. In J. Cardoso M. Lytras (Eds.), *Semantic web engineering in the knowledge society.*
- Geleijnse, G., Korst, J., De Boer, V. [2006]. Instance classification using co-occurrences on the web. In *Proceedings of the iswc 2006 workshop on web content mining with human language technologies (webconmine).* Athens, GA. (http://orestes.ii.uam.es/workshop/3.pdf)
- Geleijnse, G., Korst, J., Pronk, V. [2006]. Google-based information extraction. In F. de Jong W. Kraaij (Eds.), *Proceedings of the 6th dutch-belgian information retrieval workshop (dir 2006)* (pp. 39 – 46). Delft, the Netherlands: Neslia Paniculata, Enschede.
- Geleijnse, G., Schedl, M., Knees, P. [2007]. The quest for ground truth in musical artist tagging in the social web era. In S. Dixon, D. Bainbridge, R. Typke (Eds.), *Proceedings of the eighth international conference on music information retrieval (ismir'07)* (pp. 525 – 530). Vienna, Austria: Austrian Computer Society.
- Geleijnse, G., Sekulovski, D., Korst, J., Kater, B., Pauws, S., Vignoli, F. [2008]. Enriching music with synchronized lyrics, images and colored lights. In *First international conference on ambient media and systems (ambi-sys 2008).* Quebec, QC, Canada.
- Korst, J., Geleijnse, G. [2006]. Efficient lyrics retrieval and alignment. In W. Verhaegh, E. Aarts, W. ten Kate, J. Korst, S. Pauws (Eds.), *Proceedings third philips symposium on intelligent algorithms (soia 2006)* (pp. 205 – 218). Eindhoven, the Netherlands.
- Korst, J., Geleijnse, G., De Jong, N., Verschoor, M. [2006]. Ontology-based extraction of information from the World Wide Web. In W. Verhaegh,

E. Aarts, J. Korst (Eds.), *Intelligent algorithms in ambient and biomedical computing* (pp. 149 – 167). Heidelberg, Germany: Springer.

- Sekulovski, D., Geleijnse, G., Kater, B., Korst, J., Pauws, S., Clout, R. [2008]. Enriching text with images and colored light. In *Proceedings of the is&t/spie 20th annual electronic imaging symposium.* San Jose, CA.

The following patent applications relate to this thesis.

- Geleijnse, G., Korst, J. [2005]. Method, system and device for obtaining a representation of a text. WO2007057799.
- Geleijnse, G., Korst, J., Sekulovski, D. [2007]. Method and apparatus for enabling simultaneous reproduction of a first media item and a second media item. Filed.
- Korst, J. , Geleijnse, G. [2005]. Method of obtaining a representation of a text. WO2007057809.
- Korst, J. , Geleijnse, G., Pauws, S. [2006]. Method and electronic device for aligning a song with its lyrics. WO2007129250.

Summary

Information Extraction from the Web using a Search Engine

The web currently is the de-facto source to find an arbitrary piece of information. Intelligent applications can benefit from the collective knowledge of the internet community as to be found on the web. However, the vast majority of the information on the web is represented in a human-friendly format using natural language texts. Such information in natural language texts is not machine interpretable. As intelligent applications – for example recommender systems – may benefit from such structured information, we focus on the extraction of information from the web.

We present approaches to find, extract and structure information from natural language texts on the web. Such structured information can be expressed and shared using the standard semantic web languages and hence be machine interpreted.

Information Extraction focusses on the identification of instances (given names and terms like *Technische Universiteit Eindhoven*, *Carla Bruni* and *Amarillo*) of classes (e.g. *university*, *person*, or *location*). Apart from the identification of such instances, their relations are to be discovered in a collection of texts (e.g. the relation between *Amsterdam* and *the Netherlands*). Inspired by the semantic web community, we specify the information demand (e.g. *'Find the names of all countries in the world'*, *'Given a list of pop artists, which one is said to be most related to Amy Winehouse?'*) using an ontology. The information extraction problem is expressed in terms of an ontology population problem.

Other information extraction tasks focus on the corpus rather than on the ontology. Where their goal is to identify all instances and relation as expressed in the texts, our goal is to solely find the demanded information expressed in the initial ontology. As information on the web can be assumed to be redundantly available, we do not have to recognize each formulation of a fact of interest. For example, the statement *Amsterdam is the capital of the Netherlands* is expressed in many texts using diverse formulations. To extract this information from the web, we may not

have to recognize all of the encountered formulations.

In the thesis, a simple ontology population method using patterns is proposed. Patterns are commonly occurring phrases that are typically used to express a given relation. We combine such patterns and known instances into search engine queries. For example if *Anton Philips* is a known instance, *was a* is pattern expressing the relation between a person and his profession, we combine the two into the query *"Anton Philips was a"*. Subsequently, we extract instances and relations from the retrieved documents. The use of the constructed queries shows to be an effective mechanism to access highly relevant texts on the one hand and to identify relations on the other hand.

After discussing a general approach to populate an ontology, we focus on two subproblems: the identification of effective patterns and the recognition of the instances of the defined classes in texts. The presented approach contains a bootstrapping mechanism, as learned instances and patterns are used to formulate new search engine queries.

The approach is illustrated with several case-studies. In order to benchmark our method, we extract *facts* from the web. The second part focuses on the extraction of *inferable* information, i.e. information that is not present as such on the web, but can be derived by combining data from multiple documents. The last part of the thesis focuses on the discovery of community-based information. By combining content of multiple documents, the *wisdom of the crowds*, we create descriptions for instances such as books and popular artists.

We show that we can reliably extract information from the web using simple techniques. Furthermore, making use of the redundancy of information on the web, the recall of relevant information is high for the studied domains.

Acknowledgements

First of all, I would like to thank Jan Korst, my daily supervisor at Philips Research. I have greatly enjoyed our countless discussions and cups of coffee. Together we learned a lot about language, music, history and cooking. Jan has concisely read all my writings and has stimulated me to sharpen my argumentations and to aim for a deeper understanding.

I am grateful to my promotor Emile Aarts for his guidance and enthusiasm. Emile always knew to surprise me with a new insight, a different perspective or a challenging question. Our discussions were both fun and stimulating.

The members of the core committee, Franciska de Jong, Paul De Bra and Paul Buitelaar have thoroughly read the draft version of my thesis. I would like to thank them for their valuable feedback and remarks.

The *Media Interaction* group of Philips Research, currently named *User Experiences*, has shown me that there is more to an algorithm than its correctness and efficiency. I thank my roommates for making me feel at home in the office. Zharko, Evelien, Edgar, Dario, Maria, Greg, Thérèse, Annick, Carolien, Marco, Javed, Bram, Marjolein and Peter: thank you. A special thanks goes to Dragan for all the discussions and reviewing. As being a member of subsequently the *intelligent algorithms*, the *interaction algorithms* and the *computational intelligence* cluster I found myself in a stimulating and knowledgeable research environment. The algoritmenclub provided a inspiring platform to broaden my scope. I would like to thank all cluster and club members. I thank the management of Philips Research, in particular Maurice Groten and Reinder Haakma, for providing a great working environment.

Finally, I'd like to thank friends and family, especially Coen, Janneke, Hans and Stijn.

Biography

Gijs Geleijnse was born in Breda on June 12, 1979. In 1997 he received his VWO diploma from the Newmancollege in Breda. In the same year he started his computing science study at Eindhoven University of Technology. In the summer of 2004 he graduated with honors, with a specialization in formal methods. His master's thesis, supervised by Rob Nederpelt, handles the comparison between two formal languages for mathematics.

In September 2004, he started as a *Holst junior* in the Media Interaction Group of Philips Research in Eindhoven. His work was carried out in the context of the Dutch BSIK program *MultimediaN*. His research – that resulted in this thesis, several papers, inventions and patent applications – was carried out under supervision of Jan Korst and Emile Aarts.

In August 2008, Gijs jointed the User Experiences group of Philips Research as a research scientist.