Exploratory Search on Mobile Devices

Sven Schmeier

Sven Schmeier

Exploratory Search on Mobile Devices

Abstract

The research field *Exploratory search*, embedded in the field of *Human Computer Interaction (HCI)*, aims for a next generation of search interfaces beyond the document centered Google-like approaches. New interfaces should support users to find information even if their goal is vague, to learn from the information, and to investigate solutions for complex information problems.

The goal of this thesis is to provide a general framework (*MobEx*) for exploratory search especially on mobile devices.

The central part is the design, implementation, and evaluation of several core modules for on-demand unsupervised information extraction (*IE*) well suited for exploratory search on mobile devices and creating the *MobEx* framework. These core processing elements, combined with a multitouchable user interface specially designed for two families of mobile devices, i.e. smartphones and tablets, have been finally implemented in a research prototype. The initial information request, in form of a query topic description, is issued online by a user to the system. The system then retrieves web snippets by using standard search engines. These snippets are passed through a chain of the already mentioned NLP components which perform an on-demand or ad-hoc interactive *Query Disambiguation*, *Named Entity Recognition*, and *Relation Extraction* task. By on-demand or ad-hoc we mean the components are capable to perform their operations on an unrestricted open domain within special time constraints. The result of the whole process is a topic graph containing the detected associated topics as nodes and the extracted relationships as labelled edges between the nodes. The Topic Graph is presented to the user in different ways depending on the size of the device she is using.

It can then be further analyzed by users so that they can request additional background information with the help of interesting nodes and pairs of nodes in the topic graph, e.g., explicit relationships extracted from Wikipedia or extracted from the snippets as well as conceptual information of the topic in form of semantically oriented clusters of descriptive phrases. Various evaluations have been conducted that help us to understand the chances and limitations of the framework and the prototype.

Zusammenfassung

In dieser Dissertation wird ein Framework vorgestellt, das eine alternative Art der Informationssuche mittels mobiler Endgeräte, z.B. Smartphones und Tablets, ermöglicht. Als Grundidee wird die reine Dokumentensuche, die von herkömmlichen Suchmaschinen (Google, Bing, Yahoo Search, etc.) bekannt ist, durch eine explorative Suche ersetzt, in der die Suchergebnisse aus Themenwolken, die mit der ursprünglichen Suchanfrage inhaltlich verknüpft sind, bestehen.

Methodisch betrachtet unterstützen herkömmliche Suchmaschinen eine „One Shot Suche", inhaltlich ausgerichtete Interaktionen werden nicht weiter unterstützt. Dem Nutzer wird eine Liste von Dokumentenextrakten automatisch bereitgestellt, wobei das scheinbar relevanteste Ergebnis an erste Stelle steht. Jedes Extrakt, das sogenannte Snippet, in der Ergebnisliste ist unabhängig von anderen Snippets in der Liste. Die Sortierung erfolgt durch die Ranking-Algorithmen der jeweiligen Suchmaschinen. Bei diesem Vorgehen müssen Benutzer grundsätzlich genau wissen, was sie suchen. Die gefundenen Snippets und die dahinter verborgenen Dokumente sind im Prinzip die Antworten auf Anfragen, die konkrete Antwort darf der Suchende sich selbst erarbeiten. Dies bedeutet die Sichtung der Snippets, gegebenenfalls Untersuchung der entsprechenden Dokumente und im Fall der Nichtbeantwortung der Suchanfrage, der Neustart des gesamten Prozesses mit einer neu formulierten Suchanfrage.

Ein Grundbedürfnis des Suchens, das durch herkömmliche Suchmaschinen nicht geboten wird, ist die interaktive Suche. Dies ist besonders vor dem Hintergrund der Anwendung auf mobilen Endgeräten ein großer Nachteil. Der oben geschilderte Prozess ist akzeptabel auf herkömmlichen Computern, schnelles Navigieren und Eingaben per Tastatur sind - zumindest für westliche Sprachen - problemlos, auf mobilen Endgeräten sind diese Aktionen nicht ohne weiteres zielführend.

Interaktionen finden hier eher über Gesten auf Basis von grafischen Elementen statt, die auf dem Touchscreen adäquat präsentiert werden müssen.

Als ein zentrales Ergebnis dieser Dissertation wurde *MobEx*, ein Framework für explorative Suche auf mobilen Endgeräten entwickelt und implementiert. Es besteht aus verschiedenen online (auch ad-hoc oder on-demand) Informationsextraktionskomponenten, die auf Webinhalte ohne Beschränkung der Domäne angewendet werden, sowie einer multimodalen interaktiven Benutzerschnittstelle für mobile Endgeräte, die abhängig von der Art des mobilen Endgerätes unterschiedliche Ausprägungen hat. Üblicherweise haben diese Endgeräte verschiedene Bildschirmgrößen, so dass zunächst zwei Klassen zu unterscheiden sind, für die jeweils eigene Darstellungsoptionen entwickelt wurden: Für die Klasse der Tablets mit Bildschirmgrößen 7,10 und 12-Zoll werden Topic Graphen eingesetzt, die sich über oben genannte Interaktionsparadigmen auf mobilen Geräten bedienen lassen. Für Smartphones mit Bildschirmgrößen 3.5,4,4.3-Zoll werden die gefunden Topics und Relationen über navigationsbasierte Listen dargestellt. Der Kern des *MobEx* Frameworks besteht aus der Hintereinanderschaltung von austauschbaren KI-Modulen zur Verarbeitung natürlichsprachlicher Dokumente, die speziell für den Einsatz in einer Onlinebenutzung konstruiert und trainiert sind: Erkennung von Eigennamen sowie „Hot Topics" (NEI=Named Entity Identification); Extraktion von Relationen (RE=Relation Extraction); wissensbasierte Auflösung möglicher Ambiguitäten (QD=Query Disambiguation). Der **NEI** Ansatz identifiziert Entitäten, die basierend auf Suchergebnissen zu einer Suchanfrage miteinander verwandt sind. Dazu benutzt *MobEx* zunächst die Ergebnisse der herkömmlichen Suchmaschinen. Die ersten 1000 Elemente der Ergebnisliste werden mittels syntaktischer und semantischer Algorithmen untersucht, mögliche Kandidaten werden identifiziert. Der syntaktische Ansatz bestimmt zunächst die Wortarten der in den Texten gefundenen Tokens mit Hilfe eines Part-Of-Speech Taggers. So werden mit sehr hoher Genauigkeit Substantive, Verben, Adjektive, Artikel, Pronomen usw. bestimmt. Diese Wortarten werden zu Nomen- und Verbgruppen gruppiert.

Über Kollokationsbetrachtungen werden die Kandidaten identifiziert, die miteinander oder mit der ursprünglichen Suchanfrage in Verbindung stehen. Im Kern dieser Kollokationsbetrachtungen steht ein eigens entwickeltes mathematisches Maß zur Bestimmung der Pointwise Mutual Information (PMI), das neben den Kollokationshäufigkeiten von benachbarten Worten oder Wortgruppen auch den Abstand zwischen ihnen innerhalb der Texte berücksichtigt. Mit Hilfe eines mathematischen Verfahrens bestimmt der nachfolgende semantische Ansatz „Eigenwertzerlegung" (SVD=Singular Value Decomposition) mögliche latent-semantische Strukturen innerhalb der Texte und filtert die syntaktischen Kandidaten durch ein weiteres Netz.

Im Ergebnis erhält man einen Graphen von relational miteinander in Verbindung stehenden Themenbereichen (Topics). In abschließenden Evaluationen konnte gezeigt werden, dass das entwickelte Verfahren zur Extraktion von Named Entities und Topics vergleichbare bis bessere Ergebnisse liefert als andere State-Of-The-Art Verfahren.

Der Relationsextraktions-Ansatz **RE** ermittelt auf Basis der erkannten Topics die möglichen Beziehungen zwischen ihnen. Es findet eine Namensgebung der Kanten innerhalb des Graphen statt. Hierfür wurde der Kollokationsansatz auf Verbgruppen erweitert und über einen fuzzy matching Algorithmus können die Verbindungen zwischen den Kandidaten explizit formuliert werden.

Zur Evaluation des Verfahrens wurden in einem Batchlauf aus einer großen Menge von Snippets, die aus Suchanfragen erzeugt wurden, alle Relationen ermittelt und abschließend 300 Relationen randomisiert isoliert. Zwei Personen bestimmten danach die Qualität der extrahierten Relationen und es konnte gezeigt werden, dass die Akkuratheit je nach Messung 70% bzw. 88% betrugen.

Beide Ansätze benötigen vor dem eigentlichen Prozess die Disambiguierung der ursprünglichen Suchanfrage durch den **QD** Ansatz. Lautet die Suchanfrage zum Beispiel „Jim Clark", so muss das System wissen, welcher Jim Clark gemeint ist: Der Rennfahrer, der Gründer von Netscape, der Kriegsheld, der Football Spieler, der Sheriff, der Bankräuber, usw. Hierfür bedient sich der QD Ansatz der größten Wissensquelle, die es derzeit gibt, Wikipedia. Der gesamte textuelle Inhalt von Wikipedia wurde dafür in einen eigenen Suchindex überführt, der einen millisekunden-schnellen Zugriff erlaubt. Im Falle von Mehrdeutigkeiten werden dem Benutzer vor der eigentlichen Suche Teile der entsprechenden Artikel präsentiert, so dass im Anschluss eine genaue Identifikation von Eigennamen, Hot Topics und Relationen möglich ist.

In automatischen und manuellen Evaluationsrunden wurden hierbei Akkuratheitswerte zwischen 87% und 96% erreicht. Der Prototyp, der auf Basis von *MobEx* implementiert wurde, erlaubt explizit interaktiven on-demand oder ad-hoc-Zugriff auf Informationen und Wissen. Über vollständige Unterstützung von Multitouch-Gesten und die Verwendung von Graphen-Strukturen wird eine interaktive Form einer spielerischen Entdeckungsreise angeboten. *MobEx* ist skalierbar und anpassungsfähig in Bezug auf neue Domains und arbeitet theoretisch weitestgehend sprachunabhängig, getestet wurden die Sprachen Deutsch und Englisch.

In abschließenden Nutzerevaluationen waren insgesamt 26 verschiedene Personen beteiligt. 20 Personen testeten die Tablet Version, 6 Personen die Version auf dem Smartphone. Von den 26 Testern waren 8 aus dem Forschungsbereich *Language Technology*, 18 aus Bereichen, die nichts mit (Computer-)Forschung zu tun haben. 15 Tester hatten keinerlei Erfahrungen in der Benutzung des Tablets, 4 Personen der Smartphone Gruppe hatten keine Erfahrungen mit Smartphones. Innerhalb der Tests diente die Suchmaschine Google als Vergleichssystem, die Suchanfragen waren vorgegeben und beinhalteten Definitionen, Suche nach Personen und allgemeine Themen. Durch das Ausfüllen von Likert Skalen sowie Abschlussfragebögen konnten folgende Ergebnisse erzielt werden:

- Auf dem Tablet wurde bei allgemeinen Suchanfragen die Ergebnisdarstellung von Google bevorzugt. Bei Personenanfragen konnte die Darstellung der Ergebnisse als Topic Graph überzeugen.

- Die Beurteilung des Systems auf dem Smartphone hing sehr davon ab, wie vertraut die Tester mit dem Gerät waren. Je vertrauter, desto mehr lag die Präferenz auf den Google Ergebnissen

Insgesamt empfanden die Tester die Interaktivität des Prototypen als sehr angenehm und können sich vorstellen, es in Produktreife als das System der Wahl für die mobile Suche zu benutzen.

Die Forschung und Ergebnisse dieser Dissertation spiegeln nur einen kleinen Teil dessen wider, was in dem Themenbereich der explorativen Suche auf mobilen Endgeräten noch getan werden kann und wird. Ein großes Problem im Verlauf der Dissertation war der Zugriff auf Suchergebnisse aus Suchmaschinen. Zu Beginn der Arbeit wurde die Google API benutzt, die Suchergebnisse waren exzellent und performant. Zudem konnten Domänenbeschränkungen sowie Einschränkungen bzgl. der Sprache der Suchergebnisse angewendet werden. Der Zugriff über die API von Google wurde jedoch während der Entwicklung des Prototyps für Forschung allgemein eingestellt.

Die alternative Suchmaschine BING von Microsoft war die nächste Wahl für die Tests. Leider konnte die Ergebnisqualität nicht mehr erreicht werden, auch die Einschränkung von Domänen war hier nicht mehr möglich. Im August 2012 wurde auch BING für die Forschung abgeschaltet. Derzeit arbeitet das System auf der API des Suchmaschinenherstellers BLEKKO, der einzige Dienst, der noch frei für diese Art von Forschung ist. Allerdings besteht bei dieser API nicht mehr die Möglichkeit der Sprachwahl in den Suchergebnissen was maßgeblich die Gesamtevaluation beeinflusst hat.

Eine engere Kooperation zwischen universitären Institutionen - auch außerhalb der USA - und den privatwirtschaftlichen Anbietern wäre wünschenswert für künftige Themen dieser Art.

Acknowlegdements

I wish to thank all my colleagues, friends and relatives who have given me support and encouragement during my work on this dissertation.

Above all, I am deeply grateful to my supervisor Günter Neumann - not only for the fruitful discussions during the development of the thesis. His deep knowledge about the area of my research, his enthusiasm for research in general and his person as a whole inspired me to go on and never give up in finding adequate and innovative solutions to upcoming problems.

My sincere thanks go to Peter Adolphs, Michael Kruppa, Hans Uszkoreit, and Feiyu Xu - alphabetically ordered - who lend me an open ear for ideas, complaints, joy, and disappointments, and helped me a lot with their advices and positive thoughts.

I am very grateful to Ai Renlong, Nicolaas Bongaerts, and again to Günter Neumann, Hans Uszkoreit and Feiyu Xu for their great project and product work and support, allowing me to concentrate on thesis writing during the last years.

Finally I want to thank my beloved wife Jasmin for her love and support, for encouraging me to write faster and better, and better again.

Last but not least I thank my parents and my sister for everything.

Contents

A Evaluation of PCL(SIL)

B Evaluation of PCL(MIL)

C PCL(MIL) and linguistic processing

List of Figures

List of Tables

1

Introduction

Nowadays, the main tools for accessing information on the internet are standard search engines like Google, Bing, Yahoo, etc. The typical and simple workflow is that a user enters one or more keywords that represent the information of interest and receives a ranked list of documents or snippets. The search engine results contain concrete documents or web pages with the desired information, hopefully in the first n places. Search engines also present so called snippets, small texts consisting of parts of sentences in the result document that contain at least one term of the search query. Hopefully these snippets are expressful enough to the user so that she is able to pick the right document. The user has to read through the documents and then eventually reformulate the query in order to find new information. Seen in this context, current search engines seem to be best suited for "one-shot search" but do not support content-oriented interaction.

The following aspects are important in this context: 1) Users basically have to know what they are looking for. 2) The documents serve as answers to user queries. 3) Each document in the ranked list is considered independently. 4) The ranking is according to the search engines ranking algorithms.

If the user only has a vague idea of the information in question or just wants to explore the information space, the current search engine paradigm does not provide enough assistance for these kinds of searches.

A second aspect in the context of this thesis is that since the fast development of mobile internet and smartphones/tablets people want to use search engines via their mobile devices too.

Unfortunately, not only the user interfaces, but also the workflow described above, seems not to be well suited for a device with limited screensize and limited keyboard access. In case of mobile devices, the most convenient way to interact with a system is by touching buttons, swiping the screen, squeezing it with two or more fingers etc. With each interaction the user expects an immediate response by the systems. According to the generally accepted Human Interface Guidelines[1] the most convenient reactions to user touches are:

- Single Touch: Open something new out of the object that has been touched

- Double Touch: Switch or leave the current view for details

- Squeezing Fingers: Zooming in or out

- Swiping: Get a next page or flip something

- Touch for longer time: Go inside if some object

Therefore, we believe that interacting with linked structures like they are contained in the Web is perfectly suited for interactions like that.

1.1 Major contributions

The **MobEx** approach is an on-demand or ad-hoc information and knowledge accessing system, especially designed to run on mobile devices. It fully supports multitouch gestures, uses topic graph structures to present information nuggets and relationships between them, supports interactive "playful" knowledge dicovery, is scalable and adaptable with respect to new domains, relations and language independent at least in its basic form. All processes of information extraction are unsupervised or use general non domain independent knowledge bases and run extremely fast.

In its heart **MobEx** consists of a chain of interchangeable NLP components, specially designed and trained for application in an on-demand or ad-hoc way. Within this work we concentrated on three main components.

---

[1]http://en.wikipedia.org/wiki/Human_interface_guidelines

First, an ad-hoc *Named Entity Identification (NEI)* (chapter 3) approach that identifies named entities that are related to a given topic using snippets coming from a search engine in a fully unsupervised way. The basic concepts, underlying the *NEI* component, are based on syntactical and semantical observations. For the syntactical part we developed a sophisticated collocation algorithm based on chunks (in contrast to word based collocation) that makes use of a special pointwise mutual information (PMI) (98)), that explitely makes use of distance observations between the chunks. The semantical part is realised using a *Singular Value Decomposition (SVD)*(46) applied on snippets. The combination of the approaches show comparable and even better results to known systems for *NER* (like e.g.: (6), (22), etc.). Please note that the *NEs* are not classified, but only identified.

Second, a *Relation Extraction (RE)* (chapter 4) approach which extracts relations between the named entities, again fully unsupervised. Here, we make use of an extension of the collocation task in *NER*. Instead of working with chunk tuples, here, we deal with chunk triples, containing *NG-chunks* (noun-group chunks) and *VG-chunks* (verb-group chunks). In order to overcome the problem of sparseness of such chunk triples we developed a special fuzzy matching algorithm, that does a typed fuzzy matching on each of the arguments of the triples, depending on the argument type. We introduce the concept of penalties as it is also done in (55). We furthermore make use of Wikipedia to support the relation extraction process. This approach goes into the direction of (29) although in our case the NG- and VG-chunks are more simple and generic. Besides this unsupervised approach, we analysed a semi-supervised approach to *RE* based on *Multiple Instance Learning (MIL)*(32). Our analysis ended with negative results, which we find interesting enough to report it here. The detailed evaluation results are in the attachment of the thesis.

Third, for both approaches, *RE* and *NER*, we developed a *Query Disambiguation (QD)* system (chapter 5) which is based on statistical matches with one or more large knowledge bases - in our case Wikipedia. For ambiguous concepts in focus we explicitly not only solve such ambiguities, but furthermore we also reassign the associations and relations found by *NER* and *RE* to match the right semantic concept. Furthermore, we equipped the system by a Concept Extraction (*CE*) component based on (30) and adapted it for usage on mobile devices. The *CE* basically identifies and clusters descriptive sentences for the node of a topic graph that has been selected by the user.

We mainly use this component for concepts that are not contained in Wikipedia and ask definition questions.

We then developed the *MobEx* system on mobile devices with different screen sizes and hence we used different UI paradigms: For devices with screens large enough we used graphical topic graphs for presentation of results. Furthemore the system has been equipped with multi-touch and gesture controls. For smaller devices we use a navigation based representation built on a stack of touchable text.

Parts of this thesis have already been published or will be published in the following conferences, journals and books:

- *A Mobile Touchable Application for Online Topic Graph Extraction and Exploration of Web Content* presented at 21st July 2011 on the ACL (68)

- *Interactive Topic Graph Extraction and Exploration of Web Content* published in the book series *Theory and Applications of Natural Language Processing* (71)

- *Exploratory Search on the Mobile Web* presented at 8th February 2012 on the ICAART (4th International Conference on Agents and Artificial Intelligence) (69)

- *MobEx - a System Exploratory Search on the Mobile Web* will be published in the Springer book *Agents and Artificial Intelligence, ICAART2012, Revised Selected Papers*

- *Guided Exploratory Search on the Mobile Web* has been presented between 4th and 6th October 2012 on the KDIR (4th International Conference on Knowledge Discovery and Information Retrieval) (70)

- *Guidance in a System for Exploratory Search on the Mobile Web* will be published in the Springer book *Communications in Computer and Information Science (CCIS)*

1.2 Thesis Structure

The remainder of this thesis is structured into seven chapters:
Chapter 2 motivates the idea of **MobEx** in more details and presents related work.
Chapter 3 focusses on the *Named Entitiy Identification (NEI)* process, related work and evaluation.

Chapter 4 describes the extension of the previous approach by unsupervised and knowledge-based *Relation Extraction (RE)* method. Again, with related work and evaluation.

Chapter 5 presents our approach of *Query Disambiguation (QD)*, which leads to the concept of *Guided Exploratory Search*. Furthermore, it proposes an automatic method for its evaluation.

Chapter 6 presents a walk through the system showing all technologies developed in the previous chapters implemented on mobile devices.

Chapter 7 closes with a conclusion and outlook. It summarises the essential components and evaluation results of the system. Furthermore, it shows directions for future research.

2

Motivation and Related Work

In this chapter we show the motivation of our work, i.e. why we focused our research on **MobEx**, which we describe in the first part. In the second part, we show related work that has been done in recent years in selected fields of *Natural Language Processing (NLP)* and *Exploratory Search*.

2.1 Motivation

Searching for information on the internet is well known to nearly everyone who is using computers and the internet: We formulate queries consisting of one or more words and get presented a list of document extracts, so called Web snippets, as a search result. By reading through the snippets we decide whether the information behind these snippets, i.e. documents from which they have been extracted, may contain the desired information. If not we try to reformulate the query and hope to find more adequate results. In the end we inspect the information in detail by clicking on the links that lead us to the full documents and hope to find a solution to our query. Sometimes we need to iterate the whole process several times.

Implemented on small mobile devices like tablets, smartphones or mp3-devices reading through a bunch of documents or snippets can be a tedious task. Furthermore, the lack of the keyboard on such systems - usually they provide soft keyboards only - makes it hard to enter reformulations of the query. Hence the established process on ordinary computers is not so very well suited for mobile devices. The most convenient way to interact with such systems is by ttouching buttons, swiping the screen, squeezing it with two or more fingers etc.

Nowadays, users deal with lots of different mobile devices. We distinguish two families: Smartphones offer screen sizes of 3.5, 4.0 or 4.3 inches and Tablets with screen sizes varying between 7, 10 or 12 inches all coming with touchable screens. For the first the capabilities for displaying touchable text and graphics on one screen are very limited. Hence the presentation of search results or answers need to be different for these families.

The main goal of this thesis is twofold at least: First, exploring and developing new interactive ways to find information and acquire knowledge. As a main prerequisite the system has to work in an on-demand or ad-hoc way. On-demand or ad-hoc means we do not restrict the domain, we do not rely (too much) on precomputed data, the system should present the results in a reasonable time, i.e. processing including I/O operations should not exceed 5 seconds(10). Furthermore, as most users of today's search engines are used to enter search queries in form of few words the initial input to the system should work in the same way. We do not aim for building a new question answering system where the user has to formulate real questions. We also do not want to reinvent general search engines. Instead, the system should make use of today's most prominent search engines like Google, Bing, Yahoo, etc., use general information sources like Wikipedia, and it should be open to document pools like databases by using local search and indexing mechanisms based on Lucene (42).

Second, the results of all processes should be presented in an intuitive, interactive, and easy-to-navigate form to the user. The user interface should consist of a mix between well-known search result presentation mechanisms in form of snippets and documents and easy-to-use graphical interactive representations. Furthermore, the system should run on a mobile device so that the user is able to use it in nearly all everyday situations. This also means the graphical representation of results should support multitouch gestures as described above.

By providing more interactive elements to the search process we also support the idea to "find out about something". In summary this means the following:

1. We consider a user-query as a specification of a topic that the user wants to know and learn more about. Hence, the search result is basically a graphical structure of the topic and associated topics that are found.

2. The user can interactively explore this topic graph in order to either learn more about the content of a topic or to interactively expand a topic with newly computed related topics.

In the first step, the topic graph is computed on the fly from the set of web snippets that has been collected by a standard search engine using the initial user query. Instead of considering each snippet in isolation, all snippets are collected into one document from which the topic graph is computed. We consider each topic as an entity, and the edges between topics are considered as a kind of (hidden) relationship between the connected topics. The topic node also stores the set of snippets (the entity has been extracted from) and the documents retrievable via the snippets' web links.

For larger mobile devices (in our case an iPad), the topic graph is then displayed on the screen. The nodes in the topic graph do support multitouch operations and provide several ways of interacting:

- Single Touch: By single touching the node, the system will open new associated nodes.

- Double Touch: By double touching the node, the user can inspect the content of a topic, i.e, the snippets or web pages.

Of course the topic graph itself also supports multitouch operations:

- Pinch: Pinching the display zooms the topic graph.

- Double Touch: Double Touch brings the topic graph back to original size.

- Wiping: Wiping changes the center of the topic graph.

In case of smaller mobile devices the representation of results consists of a scrollable, navigation-based stack of touchable text and small symbols inside a text field. The ways of interacting are very simple and intuitive:

- Touch on the text, i.e. concept: The current screen slides to the left and the new screen coming in from the right shows the associated concepts.

- Touch on a special button besides a concept: The current screen slides to the left and the new screen incoming from the right shows the snippets from where this concept has been extracted from.

- Touch on the snippet shows the website to which the snippet is associated.

Coming back to the process of building up the topic graph there are several aspects to be considered: In such a dynamic information extraction situation, the user expects real–time performance from the underlying technology. The requested information cannot simply be pre– computed because we do not restrict the domain. Therefore most of the relevant information has to be extracted online relative to the current user request. That is why we assume that the relevant information to build up the topic graphs can be extracted from a search engine's *web snippets* and hence avoid the costly retrieval and processing time for huge amounts of documents. Of course, direct processing of web snippets also poses certain challenges for the Natural Language Processing (NLP) components.

Web snippets are usually small text summaries, which are automat- ically created from parts of the source documents, and are often only in part linguistically well–formed, cf. (59). Thus the NLP components are required to possess a high degree of robustness and run–time behavior to process the web snippets in real–time. Since, our approach should also be able to process Web snippets from different languages, the NLP components should be easily adaptable to many languages. Finally, no restrictions to the domain of the topic should be pre–supposed, i.e. the system should be able to accept topic queries from arbitrary domains. In order to fulfill all these requirements, we are favoring and exploring the use of shallow and highly data-oriented NLP components. Note that this is not a trivial or obvious design decision, since most of the current prominent information extraction methods advocate deeper NLP com- ponents for concept and relation extraction, e.g., syntactic and semantic dependency analysis of complete sentences and the integration of rich linguistic knowledge bases like WordNet.

To achieve our goal we need the following chain of NLP components, each working on the results of the previous components. Note that all of the components are either working with non specific and hence general knowledge base or in case of task specific information, everything is induced in an unsupervised way:

- Word- and Sentence-Tokenizer: The Tokenizer identifies the strings to be used by the later components and deletes unnecessary characters. It furthermore identifies sentences. This is not a trivial task (35), especially in Web Snippets sentences often are incomplete and truncated.

- PoS Tagger: The PoS Tagger classifies the tokens into certain classes. Again this task is pretty hard in our domain. It is known that PoS tagging performance of even the best taggers decreases substantially when applied to web pages (34). Web snippets are even harder to process because - said again - they are not necessary contiguous pieces of texts, and usually are not syntactically well-formed paragraphs due to some intentionally introduced breaks (e.g., denoted by ... betweens text fragments).

- Chunker: The chunker finds contigous noun- and verbgroups (*NG* an *VG*) based on PoS tags. For this we provide some general rules for the way how such groups are built-up by PoS Tags.

- Named Entity Identification *NEI*: In general, the nodes of the topic graphs should contain *NEs* or important *topics*. Due to our highly dynamic situation, *NEs* or *topics* need to be found ad-hoc and cannot rely on general rules or predefined base knowledges.

- Online Clustering: The *OC* clusters the snippets and labels the clusters. We use these labels to filter the *NEs* and *topics* detected by the *NEI*.

- Relation Extraction: The *RE* components finally identifies relationships between *NEs* and classifies them to semantic predicates. In our topic graph, predicates label the edges between the nodes.

- Query Disambiguation: Our *QD* approach is based on Wikipedia and aims to disambiguate the search query before starting the information retrieval process. With this we help the user to guide her inverstigations into the right direction.

- Concept Extraction: The *CE* identifies and clusters descriptive sentences for the node of a topic graph that has been selected by the user. With this we can also perform *QD* for *NEs* ot *topics* that are missing in Wikipedia.

In the beginning of the work on this thesis we had the idea to just reuse well-known technologies and combine them according to the above shown workflow. For some steps, i.e. Tokenizer and PoS Tagger, it was possible to adapt current technology to our needs. For the Chunker we needed special rules for coping with the special format of web snippets as they may omit certain parts of the original text which may lead to wrong *NGs* and *VGs*. For Named Entity Identification and Relation Extraction we needed to design and build up special components from the scratch. The main reasons for this are accuracy, expressivity, and performance in terms of speed. We will go into details in the corresponding chapters.

2.2 Related Work

Our approach is unique in the sense that it combines current research in the field of *Human Computer Interaction (HCI)*, especially the field of Exploratory Search, with recently developed technology from *Language Technology* especially Collocation and Concept Extraction and Unsupervised Information Extraction methods. As such, it learns from and shares ideas with other search results.

2.2.1 Exploratory Search

Nowadays, information has become more and more ubiquitous and the demands of searchers on search engines have been growing. Hence we need systems that support search behaviours beyond document oriented simple "one-shot" lookup. The research field *Exploratory Search* embedded in the field *Human Computer Interaction (HCI)* explorates the process of information seeking and tries to find solutions to support it. Exploratory search systems should, for example, discover new associations and kinds of knowledge, resolve complex information problems, or develop an understanding of terminology and information space structure. The general aim of this research is to come to a next generation of search interfaces to support users to find information even if the goal is vague, to learn from the information, and to investigate solutions for complex information problems.

"Exploratory search can be used to describe an information-seeking problem context that is open-ended, persistent, and multi-faceted; and to describe information-seeking processes that are opportunistic, iterative, and multi-tactical" (101). Exploratory searches are driven by curiosity or a desire to learn about or investigate something. According to Marchionini (60) a more detailed view on search is:

1. Lookup:

   - Fact retrieval, Known-item search, Navigation, Transaction, Verification, and Question answering[1]

2. Learn:

   - Knowledge acquisition, Comprehension/Interpretation, Comparison, Aggregation/Integration, and Socialize

3. Investigate:

   - Accretion, Analysis, Exclusion/Negation, Synthesis, Evaluation, Discovery, Planning/Forecasting, and Transformation

The current ranked list approach, i.e. todays most-used search interface, is well-suited for Lookup, it is hard to use it for Learning but for Investigating things it is to simple and does not support a discourse of questions and answers. Furthermore, it is known for the fact that information to the end of this list will often never be accessed (93).

In the following we will present some known approaches and describe how they influence the user's search perfomance.

2.2.1.1   Grouper

The clustering interface *Grouper* by Zamir and Etzioni (105) has been originally implemented for the HuskySearch engine and it has been compared with the ranked-list interface of the same (Fig. 2.1). This engine uses the clustering algorithm called Suffix Tree Clustering (STC), that groups the search results into coherent groups. Through the analysis of behaviour logs of the search engine with and without clustering it could be shown that no clustering is needed in finding specific documents that are ranked very high in the result set of the engine. Documents that are ranked in the mid range can be found better by using clustering.

---

[1]Answering specific question like: when, who, where, how much,etc - in contrast to: how, why,.....

Figure 2.1: The Grouper system

After some time working with the system people started to enjoy the clustering system although not in all cases.

2.2.1.2   Findex

The Findex system by Käki (50) again uses clustering to organize search results. An automatic computation of labelled categories/clusters based on the search results by Google is shown to the left side of the web interface. The clusters may be clicked to filter the overall search result set. The evaluation of the system has been based on analysis of weblogs and by final questionnaires for the testers. The results pretty much confirmed the results by Zamir and Etzioni: specific searches show less improvement than vague searches concerning user's performance. Also users need to get used to the new kind of result presentation, but they accept and even like it more after very short time. (see Fig. 2.2)
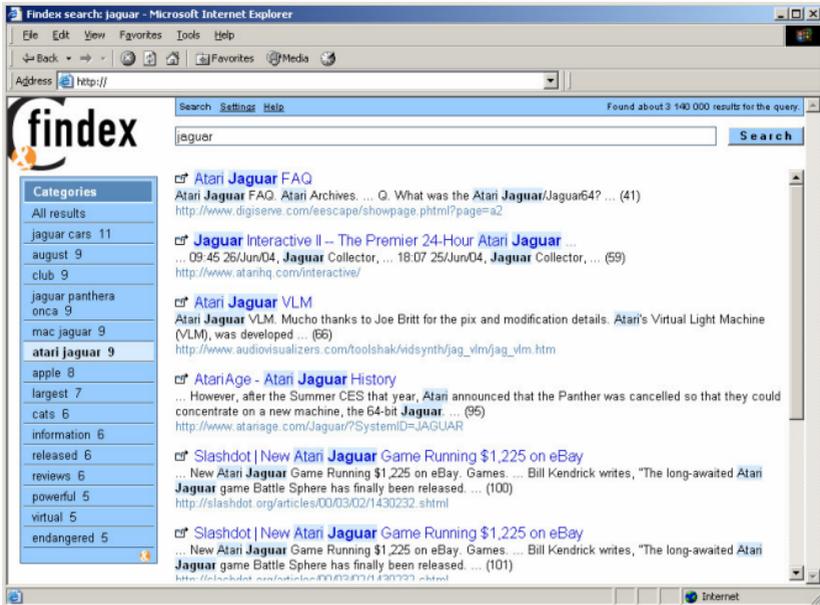
Figure 2.2: The Findex system

2.2.1.3   WordBars

The WordBars system (Fig. 2.3) by Hoeber and Yang (45) provides
active user interaction during the search process in contrast to the pre-
vious systems. It visually presents an ordered list of terms that occur in
the titles and snippets of the first 100 documents gathered by Google.
The user has the possibility to add or remove terms from his query and
thereby resort the search results. In fact WordBars helps the user to re-
fine her query and with this supports result exploration for both, specific
and vague initial queries. They report that one fundamental design of
their system is to create the right balance between computer automation
and human control(91). Hence WordBars does not simply expand the
original query but instead waits for user interaction before doing next
steps. The crucial part is to present the possible choices as good as
possible in order to create a real interactive web information retrieval
system. The authors show that for specific and for vague initial queries

their system is able to improve the overal result quality, although there
was no significant improvement concerning the user's performance.
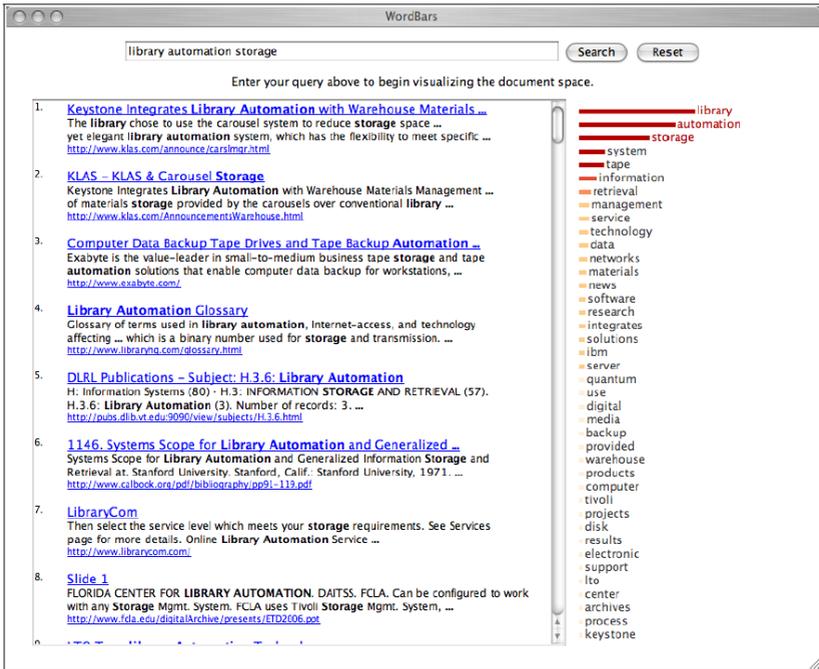
Figure 2.3: The Wordbars system

2.2.1.4   WebSearchViz

WebSearchViz by Nguyen and Zhang (72) uses the analogy to the solar
system for presenting the search results (Fig. 2.4). The query represents
the sun, the documents are the planets and location, speed, rotation,
color, and distance of objects represent the ranking of the result docu-
ments. The workflow inside the system is as follows:

1. A user enters a query into the system.

2. The system sends the query to the Google search engine and collects the found documents[2].

3. The collected pages are lexically analyzed. Then, the user is asked to provide the subjects of interest. The user assigns weights for keywords that correspond to each subject.

4. She/he can choose any subjects to be displayed in the visual space. The others will not be shown, still remain in the system unless the user explicitly deletes them.

5. During the interaction with the visual space, the user is able to modify, add, delete, or redefine subjects at will. The visual space will be updated accordingly.

2.2.1.5   Lighthouse

The *Lighthouse* system by Leuski and Allan (54) combines the ranked list representation and clustering visualization (Fig. 2.5). The documents are represented as spheres floating in space and they are positioned depending on their mutual relatedness. The more related the closer are the spheres. Hence the result space shows clustered documents and documents that do not belong to any clusters. The evaluation for measuring users' acceptance showed positive results. Users have proven to be more successful with the Lighthouse system than they are using ranked document lists.

2.2.1.6   3D Clustering

Akhavi, Rahmati,and Amini (2) apply the results of a clustering algorithm on the representation like a fractal tree (Fig. 2.6). You can zoom into the leaves of the tree and find more and more details down to the document itself. The thickness of a branch represents the density, i.e. semantic closeness of the documents.

---

[2]... not only the snippets
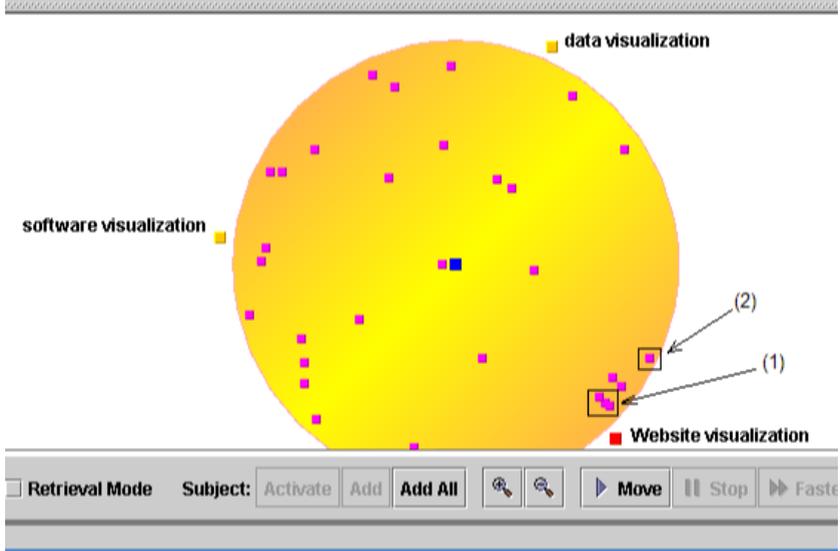
Figure 2.4: The WebSearchViz system

2.2.1.7 WhatsOnWeb

Di Giacomo, Didimo, Grilli, and Liotta (21) organise search results of
Web clustering engines (Fig. 2.7). The *WhatsOnWeb* - system uses
graphs instead of trees to present the clusters and subcluster of the result
document set for a query. According to their evaluations the graph-based
interface showed more or less similar successful result identification by
users compared to tree based systems. When it comes to find the correct
single documents the graph based approach was more appropriate.
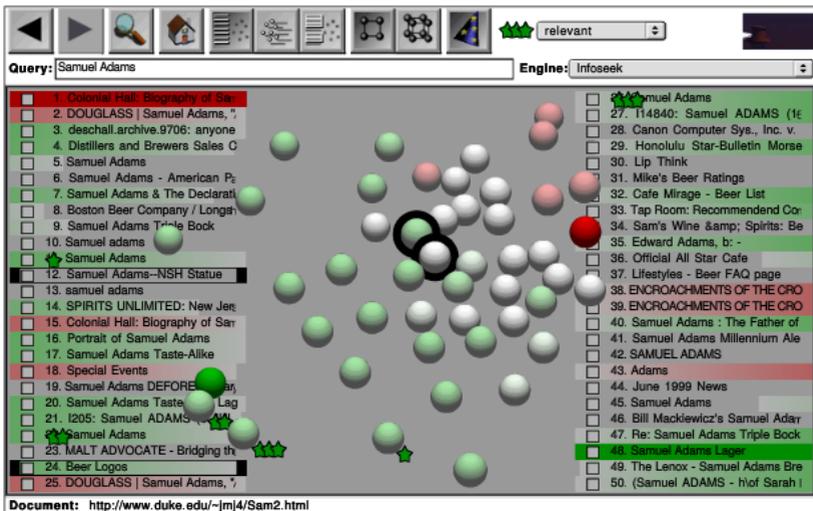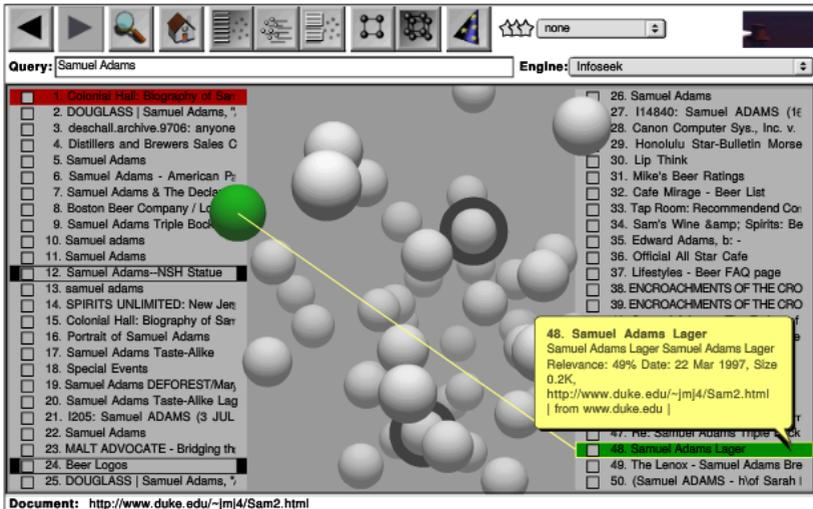
Figure 2.5: The Lighthouse system
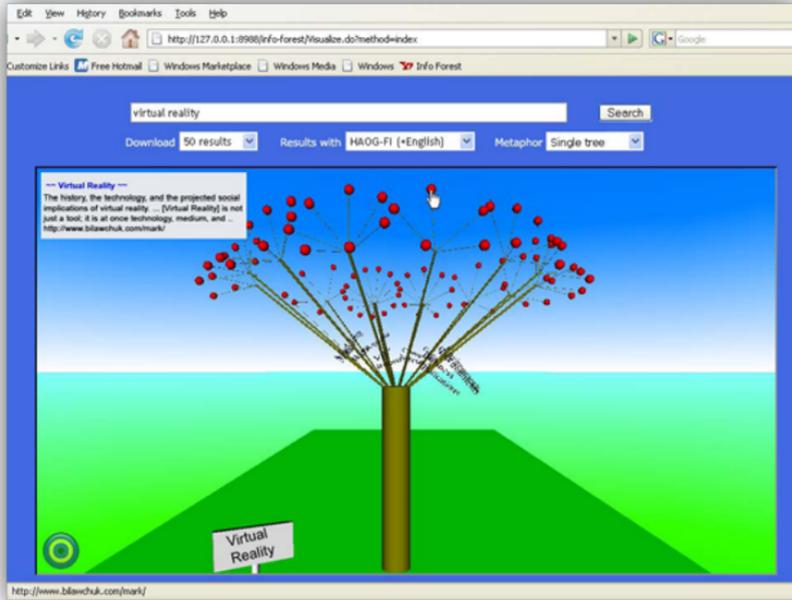
Figure 2.6: 3D Clustering of search results

2.2.2   Summary

The current research in the area of Exploratory Searchs aims to improve
the result presentation. Despite the use of different experimental visu-
alisations to avoid the simple list presentation, clustering is used as the
basic technique. Most approaches mentioned above make use of recently
developed clustering technologies, either based on Suffix Trees or on Sin-
gular Value Decomposition - both approaches are known to be able to
work as an online tool as they are fast and scalable up to the usual
numbers of retrieved documents for a search query. Furthermore, they
usually provide cluster labels that can be used for result presentation.
In the above mentioned three stages classification for search tasks these
approaches can be used for *Lookup* and *Learn* search whereas the *Investi-
gate* search is not really supported. For this personalized search systems
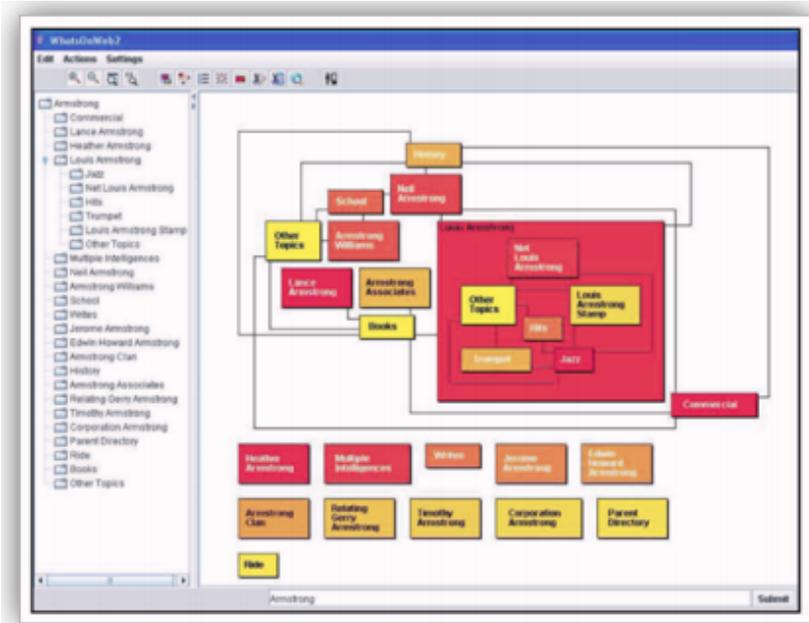
Figure 2.7: The WhatsOnWeb system

like *Dilia* (83) are more appropriate where you can store search results for further operation like tagging, categorizing, keywording, merging, cutting, etc. the documents.

Another research direction is focusing on data that nowadays still is hard to search in like blogs, forums or social media like Facebook or Twitter. For example O'Connor et al. (73) built a system called *Tweetmotif* which groups messages by frequent significant terms in order to allow some kind of thematic driven search. However, also in this approach word-based clustering is the dominating technique.

The approach presented in this thesis clearly concentrates on the *Learn* aspect with two fundamental differences: (1) our focus of research is on mobile devices and (2) we do no rely on clustering techniques only, but we rather make use of more state of the art methods from language technology.

2.2.3   Web Information Extraction, Relation Extraction, and Collocation Extraction

Web Information Extraction (WIE) systems have recently been able to extract massive quantities of relational data from online text. The most advanced technologies are algorithmically based on Machine Learning methods taking into account different granularities of linguistic feature extraction, e.g., from PoS-tagging to full parsing. The underlying methods of the learning strategies for these new approaches can range from supervised or semi-supervised to unsupervised. Currently, there is a strong tendency towards semi-supervised and, more recently, unsupervised methods.

For example, Shinayama et al (89) present an approach for unrestricted relation discovery that is aimed at discovering all possible relations from texts and presents them as tables. Sekine (84) has further developed this approach to what he calls "on-demand information extraction". Major input to the system is topic based in form of keywords that are used to crawl relevant Web pages. The system then uses dependency parsing as a major tool for identifying possible relational patterns. The candidate relation instances are further processed by specialized clustering algorithms. A similar approach has been developed by Eichler et al. (24) who further combines this approach with advanced user interaction.

Another approach of unsupervised IE has been developed by Oren Etzioni and colleagues, cf. (27); (26); (104). They developed a range of systems (e.g., KnowItAll, Textrunner, Resolver) aimed at extracting large collections of facts (e.g., names of scientists or politicians) from the Web in an unsupervised, domain-independent, and scalable manner. In order to increase performance, specific Machine Learning based wrappers have been proposed for extracting subclasses, lists, and definitions.

The bottleneck of Etzioni's and his colleagues' work is that they focus on the extraction of unary relations, although they claim these methods should also work on relations with greater arity. Rosenfeld (80) present URES, an unsupervised Web relation extraction system, that is able to extract binary relations, on a large scale, e.g., CEO_of, InventorOf,

MayorOf reporting precision values in the upper 80ies. Furthermore, Downey et al. (23) present a method that is able to handle sparse extractions.

Bunescu and Mooney (13) propose binary relation extraction based on Multiple Instance Learning (MIL). The process starts with some positive and negative instances of a relation, retrieves documents or snippets matching those instances and builds positive and negative training sets for a MIL algorithm. The generated model is then used to classify whether a text contains the relation or not.

Systems for extracting n-ary relations usually use parsers in combination with bootstrapping methods. See for example, the approaches presented by Greenwood and Stevenson(38), Sudo et al. (94), McDonald et al. (62). One main technique we use in our system is a special form of collocation extraction. A large amount of work has been done in this area. There are a lot of different approaches ranging from very shallow methods, that try to find word collocations on pure statistical means on tokenized strings, to full parsing-based methods. Usually, collocation extraction is meant to find multi word expressions in order not to split them up in further processing. Typical examples are e.g. *great difficulty*, *grow steadily*, *proof of concept*, *pay attention*, *reach consesus*, etc.. Such phenomena can be found in nearly all languages.

Earlier methods generally deal with n-grams (adjacent words) only, and use the plain co-occurrence frequency as an association measure (17). Justeson et al (49) make use of Part of Speech (PoS) Tagging to filter candidates. Curch and Hanks (18) extract phrasal verbs by again PoS-Tagging the text and applying the *Mutual Information* measure to sort the candidates. The Xtract system by Smadja (92) detects several collocations like rigid noun phrases or predicative collocations using the *z-score* measure combined with heuristics as for example cooccurences having the same distance in all texts in focus. Furthermore, they use a parser to validate their results. Collocations with flexible distances are found in more recent systems by using shallow, dependency or full parsers. Examples for such systems are (18), (51), or (57). Finally Seretan and Wehrli (85) and (86) use a full parser and achieve to extract long-distance collocations.

3

Named Entity Identification in MobEx

The main task of the Named Entity Identification (*NEI*) component in **MobEx** is to determine an initial set of correlated entities from the input topic. Such a set of correlated entities corresponds to an association graph, which is the basis for the **topic graph**. In contrast to Name Entity Recognition system we do not classify the identified NEs into classes like *person*, *location*, *event*, etc.

3.1 Collocation Extraction on Web Snippets

As already mentioned, the motivation behind the construction of the topic graph is to provide an initial first overview of possibly relevant topic related phrases, which turn out to be NEs in most cases (see section 3.3), their potential inter–relationship[1] and secondly to allow the user to interact with the system by gestures (see Fig. 3.1 showing a topic graph visualized on the iPad).

---

[1]...and even suggestions for further data exploration in the **MobEx** framework
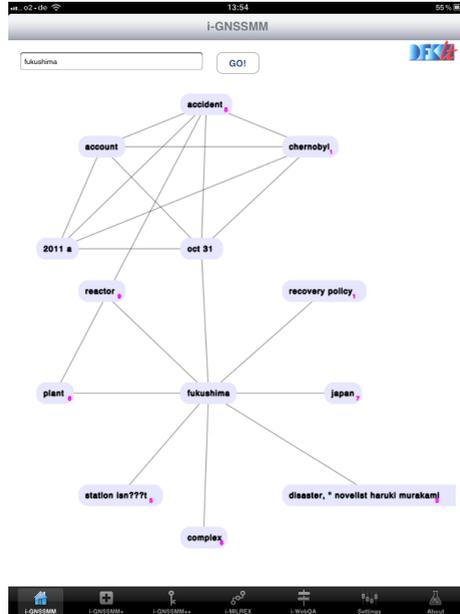
Figure 3.1: A Topic Graph for the query *fukushima*

Since the topic graph initializes the process of **MobEx** framework it is important that the Topic Extraction Process (*TEP*) is

- fast: because it is supposed to be used on-line.

- on-demand: the user should be able ask about anything, i.e., there should be no restrictions to the scope of a topic

- up-to-date: the most actual information should be identified

- indicative: the information should point to relevant information

The core idea of our *TEP* is to compute a set of chunk–pair–distance elements for the N first web snippets returned by a search engine for the topic Q, and to compute the topic graph from these elements.[2] In general for two chunks, a single *chunk–pair–distance (cpd)* element stores the distance between the chunks by counting the number of chunks in–between them.

---

[2]For the remainder of the thesis we set N=200.

We distinguish elements which have the same words in the same order, but have different distances. For example, (Peter, Mary, 3) is different from (Peter, Mary, 5) and (Mary, Peter, 3).

The motivation for defining a *cpd model* is our assumption that a *cpd* element with a high frequency covers some syntactic co-relation, and hence, some relational information of the corresponding word pairs. Next, if we not only consider arbitrary words, but noun and verb groups and their relative distance as source of a chunk–pair–distance representation, we might detect candidate relevant structural dependencies, which indicate possible semantic relationships.

Following this consideration, the major steps performed by *TEP* are as follows:

1. Retrieval of N web snippets

2. Linguistic analysis of web snippets

3. Computation of chunk–pair–distance model

4. Computation of topic graph

We begin by creating a document S from the N-first web snippets so that each line of S contains a complete snippet. In our research we use the *BING* search engine by Microsoft. We decided in favour of BING, because they provide an excellent developer API and do not restrict the number of search queries per day. Each textline of S is then tagged with Part–of–Speech using the SVMTagger (36) and chunked in the next step. The chunker recognises two types of word chains. Each chain consists of longest matching sequences of words with the same PoS class, namely noun chains or verb chains, where an element of a noun chain belongs to one of the extended noun tags[3], and elements of a verb chain only contain verb tags. We finally apply a kind of "phrasal head test" on each identified chunk to guarantee that the right–most element only belongs to a proper noun or verb tag. For example, the chunk "a/DT british/NNP formula/NNP one/NN racing/VBG driver/NN from/IN scotland/NNP" would be accepted as proper NP chunk, where "compelling/VBG power/NN of/IN" is not.

---

[3]Concerning the English PoS tags, "word/PoS" expressions that match the following regular expression are considered as extended noun tag:
"/(N(N|P))|/VB(N|G)|/IN|/DT". The English Verbs are those whose PoS tag start with VB. We are using the tag sets from the Penn treebank (English) and the Negra treebank (German).

Performing this sort of shallow chunking is based on the assumptions: 1) noun groups can represent the arguments of a relation, a verb group the relation itself, and 2) web snippet chunking needs highly robust NL technologies. In general, chunking crucially depends on the quality of the embedded PoS tagger especially for web snippets. On the other hand, we want to benefit from PoS tagging during chunk recognition in order to be able to identify, on the fly, a shallow phrase structure in web snippets with minimal efforts. In section 3.2 we present our evaluation of the performance of the SVMTagger and our algorithmic solution to some problems.

The chunk–pair–distance model is computed from the list of noun group chunks.[4] This is done by traversing the chunks from left to right. For each chunk c_i, a set is computed by considering all remaining chunks and their distance to c_i, i.e., $(c_i, c_{i+1}, dist_{i(i+1)})$, $(c_i, c_{i+2}, dist_{i(i+2)})$, etc. We do this for each chunk list computed for each web snippet. The distance $dist_{ij}$ of two chunks c_i and c_j is computed directly from the chunk list, i.e., we do not count the position of ignored words lying between two chunks.

Finally, we compute the frequencies of each chunk, each chunk pair, and each chunk pair distance. The set of all these frequencies establishes the chunk–pair–distance model CPD_M. It is used for constructing the topic graph in the final step. Formally, a topic graph $TG = (V, E, A)$ consists of a set V of nodes, a set E of edges, and a set A of node actions. Each node $v \in V$ represents a chunk and is labelled with the corresponding PoS tagged word group. Node actions are used to trigger additional processing, e.g. displaying the snippets, expanding the graph etc.

The nodes and edges are computed from the chunk–pair–distance elements. Since the number of these elements is quite large (up to several thousands), the elements are ranked according to a weighting scheme which takes into account the frequency information of the chunks and their collocations.

---

[4]Currently, the main purpose of recognizing verb chunks is to improve proper recognition of noun groups. The verb chunks are ignored when building the topic graph.

More precisely, the weight of a chunk–pair–distance element $cpd = (c_i, c_j, D_{ij})$, with

$D_{i,j} = \{(freq_1, dist_1), (freq_2, dist_2), ..., (freq_n, dist_n)\}$, is computed based on PMI (98) as follows:

$$PMI(cpd) = log_2((p(c_i, c_j)/(p(c_i) * p(c_j))))$$

$$= log_2(p(c_i, c_j)) - log_2(p(c_i) * p(c_j))$$

where relative frequency is used for approximating the probabilities $p(c_i)$ and $p(c_j)$. For $log_2(p(c_i, c_j))$ we use the (unsigned) polynomials of the corresponding Taylor series[5] using $(freq_k, dist_k)$ in the k-th Taylor polynomial and adding them up:

$$PMI(cpd) = (\sum_{k=1}^{n} \frac{(x_k)^k}{k}) - log_2(p(c_i) * p(c_j))$$

$$, where \ x_k = \frac{freq_k}{\sum_{k=1}^{n} freq_k}$$

The visualized topic graph TG is then computed from a subset $CPD'_M \subset CPD_M$ using the m highest ranked cpd for fixed c_i. In other words, we restrict the complexity of a TG by restricting the number of edges connected to a node.

3.2   Semantic Filtering of Noisy Chunk Pairs

The motivation for using the chunk–pair–distance statistics is the assumption that the strength of hidden relationships between chunks can be covered by means of their collocation degree and the frequency of their relative positions in sentences extracted from web snippets, and as such, are emphasizing syntactic relationships. Figueroa and Neumann (30) demonstrated the effectiveness of this hypothesis for web–based question answering. In general, chunking crucially depends on the quality of the embedded PoS tagger. However, it is known that PoS tagging performance of even the best taggers decreases substantially when applied to web pages (34). Web snippets are even harder to process because they are not necessarily contiguous pieces of texts. For example, an initial manual analysis of a small sample revealed, that the extracted

---

[5]In fact we used the polynomials of the Taylor series for $ln(1 + x)$. Note also that k is actually restricted by the number of chunks in a snippet.

chunks sometimes are either incomplete or simply wrong. Consequently, this also deteriorated the "readability" of the resulting topic graph due to "meaningless" relationships. Note that the decreased quality of PoS tagging is not only caused by the different style of the "snippet language", but also because PoS taggers are usually trained on linguistically more well–formed sources like newspaper articles (which is also the case for our PoS tagger in use which reports an F–measure of 97.4% on such text style).

In order to tackle this dilemma, investigations into additional semantical–based filtering seems to be a plausible way to go.

About the Performance of Chunking Web Snippets

As an initial phase into this direction we collected three different corpora of web snippets and analysed them according to the amount of well–formed sentences and incomplete sentences contained in the web snippets. Furthermore, we also randomly selected a subset of 100 snippets from each corpus and manually evaluated the quality of the PoS tagging result. The snippet corpora and results of our analysis are as follows (the shortcuts mean: #s = number of snippets retrieved, #sc = well–formed sentences within the set of snippets, #si = incomplete sentences within the snippets, #w = number of words, F(x) = F–measure achieved by the PoS tagger on a subset of 100 snippets with x words).

Fukushima   this corpus represents snippets mainly coming from official online news magazines. The corpus statistics are as follows:

#s	#sc	#si	#w	F(2956)
240	195	182	6770	93.20%

Justin Bieber   this corpus represents snippets coming from celebrity magazines or gossip forums. The corpus statistics are:

#s	#sc	#si	#w	F(3208)
240	250	160	6420	92.08%

New York   this corpus represents snippets coming from different official and private homepages, as well as from news magazines. The corpus statistics are:

#s	#sc	#si	#w	F(3405)
239	318	129	6441	92.39%

Summarised, 39% of all tagged sentences are incomplete and the performance of the PoS–tagger decreases by about 5% F–measure (compared to the reported 97.4% on newspapers). Consequently, a number of chunks are wrongly recognized. For example, it turns out that date expressions are systematically tagged as nouns, so that they will be covered by our noun chunk recognizer although they should not (cf. section 3). Furthermore, the genitive possessive (the "'s" as in "Japan's president") is classified wrongly in a systematic way, which also had a negative effect on the performance of the noun chunker. Very often nouns are wrongly tagged as verbs because of wrongly identified punctuation. Thus, we need some filtering mechanism which is able to identify and remove the wrongly chunked topic–pairs resulting from errors in the PoS–tagger process.

A promising algorithmic solution to this problem is provided by the online clustering system *Carrot2* (76), that computes sensible descriptions of clustered search results (i.e., web documents). The Carrot2 system is based on the Lingo (74) algorithm. Most algorithms for clustering open text follow a kind of "document–comes–first" strategy, where the input documents are clustered first and then, based on these clusters, the descriptive terms or labels of the clusters are determined. The Lingo algorithm actually reverses this strategy by following a three–step "description–comes–first" strategy (see (74) for more details):

1. extraction of frequent terms from the input documents

2. performing reduction of the (pre–computed) term–document matrix using Singular Value Decomposition (SVD) for the identification of latent structure in the search results

3. assignment of relevant documents to the identified labels.

The specific strategy behind the Lingo algorithm matches our needs for finding meaningful semantic filters very well: we basically use step 1) and 2) to compute a set of meaningful labels from the web snippets determined by a standard search engine as described above. According to the underlying latent semantic analysis performed by the Lingo algorithm, we interpret the labels as semantic labels. We then use these labels and match them against the ordered list of chunk–pair–distance elements computed in the *NEI* step described above. This means that all chunk–pair–distance elements that do not have any match with one of the semantic labels are deleted.

The idea is that this filter identifies a semantic relatedness between the labels and the syntactically determined chunks. Since we consider the labels as semantic topics or classes, we assume that the non-filtered pairs correspond to topic–related (via the user query) relevant relationships between semantically related decriptive terms.

Of course, the quality and usefulness of the extracted topics and topic graph remains to be evaluated. In the next sections we will discuss two directions: a) a quantitative evaluation against the recognition of different algorithms for identifying named entities and other rigid identifiers, and b) a qualitative evaluation by going into some details of the extracted topics. In section 6.2 we also present an evaluation concerning the user experience with the final system. Indirectly, this evaluation can be seen as a qualitative evaluation by means of the analysis of user experience.

3.3   Evaluation

Our *NEI* process is completely unsupervised and web–based, so evaluation against standard gold corpora is not possible, because they simply do not yet exist (or at least, we do not know about them). For that reason we decided to compare the outcome of our *NEI* process with the outcomes of a number of different recognisers for named entities (NEs).

Note that very often the extracted topics correspond to rigid designators or generalized named entities, i.e., instances of proper names (persons, locations, etc.), as well as instances of more fine grained subcategories, such as museum, river, airport, product, event (cf. (65)). So seen, our *NEI* process (abbreviated as *TEP*) can also be considered as a query–driven context–oriented named entity extraction process with the notable restriction that the recognised entities are unclassified. If this perspective makes sense, then it seems plausible to measure the degree of overlap between our *NEI* process and the recognized set of entities of other named entity components to learn about the coverage and quality of *TEP*. For the evaluation of *TEP* we compared it to the results of four different NE recognizers:

1. SProUT(6): The *SProUT*–system is a shallow linguistic processor that comes with a rule–based approach for named entity recognition.

2. AlchemyAPI[6]: *AlchemyAPI*–system uses statistical NLP and machine learning algorithms for performing the NE recognition task.

3. Stanford NER(22): The *Stanford NER*–system uses a character based Maximum Entropy Markov model trained on annotated corpora for extracting NEs.

4. OpenNLP[7]: A collection of natural language processing tools which use the Maxent package to resolve ambiguity, in particular for NE recognition.

We tested all systems with the three snippet corpora described in section 3.2.

The tables 3.1, 3.2, and 3.3 show the main results for the three different corpora; table 3.4 shows the results summarised. All numbers denote percentages that show how many relevant[8] NEs of the algorithm in the row could be extracted by the algorithm in the column. For example, in the dataset "Justin Bieber" *TEP* extracted 85.37% of the NEs which have been extracted by *SProUT*. *AlchemyAPI* extracted 75.64% and *StanfordNER* extracted 78.95% of the NEs that have been extracted by *SProUT*. The numbers with preceding "#" show the number of extracted NEs. The following roman numbers are used to denote the different algorithms:

- I *SProUT*

- II *AlchemyAPI*

- III *StanfordNER*

- IV *OpenNLP*

- V *TEP*

---

[6]http://www.AlchemyAPI.com
[7]http://incubator.apache.org/opennlp/
[8]Relevance here means that a NE must occur more than 4 times in the whole dataset. The value has been experimentally determined.

Table 3.1: Results for query *Justin Bieber*.

	I	II	III	IV	V
I	#136	75.64	78.95	78.48	85.37
II	69.01	#143	93.97	86.00	97.17
III	76.71	97.52	#172	92.86	96.09
IV	74.70	89.19	88.52	#196	95.10
V	28.66	40.88	42.40	44.96	#675
	67.77	79.61	80.66	81.13	#157

Table 3.2: Results for query *Fukushima*.

	I	II	III	IV	V
I	#121	81.03	83.61	81.35	87.5
II	80.26	#129	93.46	87.36	98.48
III	85.00	94.59	#131	91.67	92.22
IV	74.65	89.13	85.26	#178	91.58
V	27.45	26.89	33.33	35.31	#543
	72.93	80.04	83.19	82.26	#132

Keeping in mind that our approach always starts with a topic around which all the NEs are grouped, i.e. NE recognition is biased or directed, it is hard to define a gold standard, i.e. manually annotate all NEs which are important in a specific context. In context of the query "Fukushima" most people would agree that word groups describing the nuclear power plant disaster clearly are NEs. Some would also agree that terms like "earthquake" or "tsunami" function as NEs too in this specific context. Given a query like "New York" people probably would not agree that "earthquake" should function as a specific term in this context. Of course there are NEs of generic type like "persons", "locations", or "companies", but it is questionable whether they suffice in the context of our task.

We compared the systems directly with the results they computed. The main interest in our evaluation was whether the NEs extracted by one algorithm can also be extracted by the other algorithms. Furthermore, we set a very simple rating scheme telling us that detected NEs with more occurrences are more important than those with lower frequencies.[9]

---

[9]Except for the *TEP*, where we used the PMI as described above.

Table 3.3: Results for query *New York*.

	I	II	III	IV	V
I	#175	81.39	88.24	85.15	71.05
II	76.60	#169	93.53	86.51	74.36
III	90.00	95.79	#280	92.35	73.28
IV	84.43	92.72	93.17	#230	83.49
V	26.64	26.83	17.71	33.07	#388
	81.11	83.90	73.77	79.87	#166

Table 3.4: Summary for NER Evaluation.

	I	II	III	IV	V
I	#432	79,25	83.6	81.66	81.31
II	75.29	#441	93.65	86.62	90.00
III	83.90	95.97	#583	92.29	87.19
IV	83.90	95.97	583	#604	87.19
V	27.58	31.53	31.15	37.78	#1606
	73.94	81.18	79.21	81.09	#455

The numbers show that *TEP* extracts nearly four times more NEs than the other systems. Therefore the numbers in the fourth row of all tables are pretty low. Hence we did a second run on the three datasets by deleting all extracted NEs that are below a certain threshold in terms of occurrences. For each dataset we computed the threshold so that the number of the remaining NEs is roughly the same as in the other methods.

The results show that, looking at the numbers and percentages, no system outperforms the others, which confirms our approach. Please note that the *TEP* approach works for query-driven context-oriented named entity recognition only. This means that all approaches used in this evaluation clearly have their benefits in other application areas. Nevertheless by going into details we saw some remarkable differences between the results the systems produced. All systems were able to extract the main general NEs like locations or persons. For terms that are important in the context of actuality and current developments, we saw that the *TEP* approach is able to extract more relevant items.

In case of "Fukushima", the *SProUT* system did not extract terms like "earthquake", "tsunami" or "nuclear power plant". Of course this is because the underlying ruleset has not been developed for covering such types of terms. The *AlchemyAPI* and *StanfordNER* systems were able to extract these terms but failed in detecting terms like "accident" or "safety issues". For "Justin Bieber" relevant items like "movie", "tour-dates" or "girlfriend" could not be detected by all systems except *TEP* . For the snippets associated with the query "New York" all systems identified the most important NEs, and differed for less important NEs only.

Last but not least, the runtime, which plays an important role in our system, varied from 0.5 seconds for the *SProUT* system, to 2 seconds for *TEP*, 4 seconds for *StanfordNER* to 15 seconds for *AlchemyAPI*.

3.4   Related Work

Named Entity Ecognition (NER) has a long history in NLP research. As a key part of *Information Extraction (IE)* it became a stand-alone task in the Message Understanding Conference (MUC-6)(39) in 1996. At that time the main *NER* activity has been to extract person, organisation and location names as well as time, date, and money expressions, units, and percent expressions. (79) is among the first scientific publications that describes a system to extract company names. For more than fifteen years *NER* has been a hot topic among researchers and still it is at least for other languages than English. However, two main techniques have been settled as the core of all systems: (1) Handcrafted rules and gazeteers, (2) supervised learning based systems. Both approaches require large collections of documents to be analysed (by hand) to obtain sufficient knowledge to either design the rules or let them induced.

More lightweighted and domain independent processes like semi- and unsupervised have become of interest in the last years. Not just because they are more easy to handle, but also because the results are comparable to the supervised and handcrafted systems (66). Semi-supervised systems usually are based on bootstrapping. The bootstrapping processes start with a small sets of so-called seeds and generate possible candidates using context clues. The candidates are then proven by some process and added to the seeds. Then the whole process is repeated until the process converges (or reaches some threshold).

A very interesting approach by Pasca et al. (77) uses the combination of a technique presented in (56) to find words of a similar class and they apply it to very large corpora with more than 100 million web documents. They started from a seed of 10 sample facts and generated 1 million facts with a precision of 88%. Other systems like (9) make use of lexical features of words implemented by regular expressions. The main idea behind this is to use conventions lots of documents or websites are following. So for example to extract booktitles a possible regular expression looks like $([A-Z]0, 1[a-z,]+)+(Paperback)$ matches expressions like "Winnetou I, Karl May (Paperback)" or "Donald and Daisy, Walt Disney (Paperback)". Often such regular expression approaches are mixed with gazeteers and lexica[10]. Unsupervised systems usually use clustering techniques to detect NEs. Often enough such systems rely on either lexical resources like WordNet (3) or Wikipedia or they are based on robust systems based on supervised algorithms like detected POS tags. Other approaches - similar to our approach - consider the *NER* as a specific *empirical collocation extraction task*. Shinyama and Sekine (90) use the coccurrences of NEs that appear in multiple news sources as the indicator for a positive match. Etzioni et al. (28) make use of PMI-IR (pointwise mutual information and information retrieval) by (98). However, instead of extracting collocations between words, which is still the dominating approach in collocation extraction research (e.g. (5)), we are extracting collocations between chunks, i.e. word sequences. Furthermore, our measure of association strength takes into account the distance between chunks and combines it with the PMI (pointwise mutual information) approach.

---

[10]Some people may argue that such systems are handcrafted and not semi-supervised

4

Relation Extraction in MobEx

In the previous chapter we showed how we find associated topics to a query formulated by a user. With our syntactically and semantically based processes we are able to produce related concept and present them in a visual Topic Graph. However the kind of relation between the topics has not been made explicit yet. Therefore the user needs to have either basic knowledge or a good intuition to find the appropriate path to satisfy her interest. In this chapter we show how we equip our topic graph with meaningful relations between the nodes. The relations will label the edges in the graphical representation. We will start to introduce the main research directions in the topic of relation extraction. After that we will present the two approaches we followed during our research whereas the first approach based on semi-supervised technologies failed for our purpose and secondly the successful approach, completely unsupervised and again based on collocations. We believe it is necessary and interesting to present the unsuccessful approach too and explain why it failed in the end. Please note that most of the evaluations for that approach are to be found in the Appendix.

4.1   Related Work

Today's methods on extracting relationships mostly rely on machine learning methods combined with linguistic processing like parsing and more. In general, documents in which we search for interesting relationships are processed in the following way:

1. Perform some kind of – not necessarily linguistic – preprocessing in order to find connections between text fragments. These connected fragments are the *relation candidates.*

2. Use formerly acquired models to instantiate a certain relationship or dismiss the relation candidate.

The linguistic preprocessing usually is highly language and domain independent. In general deeper or more sophisticated linguistic analysis requires good linguistic resources and models. Depending on the underlying documents often enough systems for relation extraction use nearly no linguistic preprocessing or perform some kind of dependency parsing (see next sections). Acquiring the model for instantiating relation candidates can be done by three main methods: supervised, semi-supervised and completely unsupervised methods.

4.1.1   Supervised Relation Extraction

The most prominent supervised technique is *SIL* (Single Instance Learning). According to the *SIL* paradigm systems based on this technology build their models on labelled instances. The extraction or recognition of relations is reformulated as a classification task. Given a set of positive and negative training examples, i.e. examples that express a certain relation and examples that do not, *SIL* algorithms induces for a sentence like

$$S = w_1 w_2 ... e_1 ... w_i ... e_2 ... w_n$$

, where w_k are the words (all words of the sentence except the entities) and e_1 and e_2 are the arguments of the relation, the following function:

$$f(T(S)) = \begin{cases} +1 & relation\ is\ found \\ -1 & relation\ is\ not\ found \end{cases} \tag{4.1}$$

The form of the sentence varies according to the function T. It can be a set of features extracted from the sentence - in the above example the features are the words and the named entity - or some kind of

structured representation like a labelled sequence, parse trees, etc. Machine Learning algorithms that are able to induce the function f are for example:

1. Muggleton and de Raedt(64) propose inductive logic programming (ILP) as means to learn relations, i.e. logic predicates

2. Roth and Wen-Tau Yih(81) describe a way of relational learning using propositional means and compare it to ILP

3. Zelenko et al.(106) experiment on different kernel methods for SVMs

A main problem of *SIL* approaches is the need of a more or less huge set of labelled instances that serve as the training base which is often not easy to achieve. Furthermore, this set needs to be balanced and hopefully the examples are expressive enough so that the learning algorithm is able to generalise.

4.1.2 Bootstrapping or semi-supervised methods

Nowadays bootstrapping is one of the most used semi-supervised methods and has been applied both on binary and on n-ary relations. These methods largely rely on redundancy, i.e. most instances for a relation are assumed to be formulated in similar ways. Examples are:

1. Brin(9) starts with a small set of seed examples. In a next step patterns are induced using a labelled data set. These patterns are applied to some unlabelled document collection and new seed examples are generated. Then the whole process is repeated until it converges.

2. In the Snowball system Agichtein and Gravano(1) again start with a handful of training examples of tuples of interest. In contrast to the previous approach, here the arguments of the relation are required to be NEs. These examples are used to generate extraction patterns, which in turn result in new tuples being extracted from some document collection. The tuples are then clustered using a special similarity function:

$$M(tuple_i, tuple_j) = (pre_i * pre_j) + (suff_i * suff_j) + (mid_i * mid_j)$$
(4.2)

with tuple = [e$_1$,e$_2$,pre,suff,mid] and *pre, suff, mid* are the parts of the sentence occurring before e$_1$, after e$_2$ and between e$_1$ and e$_2$.

3. Yangarber and Grishman (103) start from seed patterns obtained by user input. The corpus itself is tagged with named entities (*NEs*) denoting people, companies and locations. Then a parser is used to extract all clauses from each document into tuples containing the head of the subject, the verb, the head of the object, locative and temporal modifiers and certain other verb arguments. The clause structures are normalised to produce uniform tuple structures, i.e. passive constructions are put to active, relative clauses are put to main clauses, etc. Using the *NEs* the seed patterns are generalised, for example the seed patterns of the topic "Managment Succession" are: "xtramind GmbH appoints Mr Smith", which is generalised to the tuple "*company* appoints *person*" and "Mr Smith resigns" which is generalised to the tuple "*person* resigns". The generalised patterns are then used to retrieve relevant documents. In a next step the patterns in the retrieved documents are weighted and the one with the highest information gain for discriminating relevant from irrelevant documents is added to the set of seed patterns.

4. Xu(102) starts from so-called "semantic seeds", which is a small set of instances of the target relation. In contrast to previous approaches, the seeds are given as dependency structures instead of textual patterns. In a next step rules are extracted from dependency structures and annotated sentences which match these seeds. By applying these rules on unseen text further relation instances are generated, which are added to the initial seeds. Then the whole process starts again until no new rules or instances can be found.

Another semi-supervised, and not based on bootstrapping, method is *MIL* (Multiple Instance Learning): Bunescu and Mooney(13) propose binary relation extraction based on multiple instance learning. The process starts with some positive and negative instances of a relation, retrieves documents matching those instances and builds positive and negative training sets for some *MIL* algorithm. The generated model will then be used to classify whether some text contains the relation or not. See chapter 4.2.1.2 for more details.

4.1.3 Unsupervised methods

Unsupervised RE means to extract relations - usually from the Web - with no training data and no list of relations.

Hasegawa et al. (41) determine relationships among NEs by using co-occurrence paired with context obeservations . For this, they build bag-of-word vectors containing two NEs of the same type and context words filtered from stop words. Those vectors are clustered in a next step. The clusters are labelled using the intersection of the most important words in the vectors of each cluster. These labels describe the found relation.

Turney (100) starts with a pair of words and uses corpora to collect possible yet unspecified relations between them. By using relational similarity based on the Latent Relational Analysis (99) it is possible to rank the most *telling* pattern highest. This means found relationships are not classified, but the best patterns are used to describe them.

Davidov et al. (20) start with seeds containing some example word pairs representing some class. For each word of such word-pairs - they call it concept words - examples are extracted from the corpus containing the concept word and a so-called target word. In a next step the sentences are clustered into concept-word-independent clusters. These clusters now can be matched to the initial seed and hence used to identify the relation classes for new unknown examples.

The IDEX system described in (25) performs the *RE* task on the basis of simplified dependency structures. By clustering those structures, based on several indicators and relaxations like synonym observations in Wordnet, fuzzy matching using extracted dependency information plus context information and furthermore, by comparing coreference sets of two relations, the resulting set of relations is extracted. Applied on a manually annotated corpus the system was able to extract 11 out of 15 potentially interesting verb relations.

Etzioni et al. (27) show ideas on how to extract information from large amounts of data, i.e. the web, by using unsupervised methods. Furthermore, they present the results of the system *Textrunner*, a system that has been developed in a project called *KnowItAll*. Textrunner is provided with detailed web information by *Google* and uses it to automatically extract (binary) relations. The starting point is an extensible ontology containing extraction templates for each relation to be extracted. An example of such a template are Hearst patterns ((44)) "NP1 such as NPList2", "NP1 , including NPList2", or "NP1 is a NP2" describing that the head of each simple noun phrase (NP) in NPList2

is an instance of the class named in NP1. In this way the system may identify sentences as "We provide tours to cities such as Paris, Nice, and Monte Carlo" by instantiating the template above. Other template patterns are allowed too of course, like "NP1 is the NP2 of NP3" or "the NP1 of NP2 is NP3". With such templates the system is able to extract relations, as well as named entities (NE), in a completely unsupervised way. In a next step the instantiated templates are used to generate text extraction rules which are used as input for the search engines like Google, Alta Vista, Fast, etc. to obtain more instances. A rule for the mentioned example looks like "cities such as x, y, and z". The hit counts delivered by the search engines are used to prove the result similar to Turney's PMI-IR algorithm(98). The original *KnowItAll* system then used bootstrapping techniques to update the instances for the templates and produce more text extraction rules. In *Textrunner* this step is omitted by training a self-supervised classification algorithm, *Naive Bayes*, based on a small set of positive and negative training documents. The *Reverb* system (29) introduces two very simple syntactic rules that improve the F-measure of the extracted relations by eliminating incoherent extractions and uninformative relations. An example for incoherent is: "The guide contains dead links and omits sites" yields the incoherent relation "contains omits"; an example for uninformative is: "is" extracted from "is the author of" or "has" extracted from "has the information about".

However all of these unsupervised approaches need a lot of data in order to minimise the errors.

4.2 Relation Extraction in MobEx

In the following two sections we present our way of performing the *RE* task. We started with the idea of learning some relations in a semi-supervised way first. The main advantages we had in mind were that we could really classify found relations to a fixed set of relation classes. This would lead to a better understanding and representation on a mobile device as the needed representation place would have been fixed. We also planned to enhance the User Interface of the system on the mobile device in a way so that the user let the system learn more relationships while using it. However as mentioned above, our first choice turned out to be not too useful, at least in our implementation.

Concerning the way of how we did our *NEI* presented in the previous chapter we also had in mind to implement the *RE* as an unsupervised module. In fact instead of using one of the above-mentioned systems, we decided to enhance the *CPD* - model introduced in chapter 3. Due to the overall system architecture (implementation-wise) it would have been easy to replace the module in case of failure. Additionally, we used an Online Clustering tool *OC* (see (75)) to improve the quality of the extracted relationships.

4.2.1   Relation extraction using Multiple Instance Learning (MIL)

(Semi-)supervised approaches require at least some offline work, i.e. a learning phase, before they can be applied on the final extraction task. In this respect, following such an approach slightly contradicts our main goal of the thesis, i.e. to provide an on-demand or ad-hoc system. Nevertheless, we thought that it might be useful, that at least some relationships between our *NEs* are general enough to be learned in advance and applied in the running system. In the end, the online behaviour of the system will be ad-hoc. For this, we worked on the above mentioned approach by Bunescu and Mooney(13), as it promises in general good results without the need of collecting and manually labelling lots of training material, as it would be the case with *SIL* methods. In contrast, the automatic extraction by using unsupervised methods does need no manual work on the data but the semantics of the extracted relationships is not always clear. Hence the manual works shifts from labelling data to labelling relations.

The idea behind *MIL* is the following: Instead of taking a set of instances, which are labelled positive or negative as the initial training base, a *MIL* learner uses a set of bags that are labelled positive or negative. Each bag contains many instances, which may be unlabelled as individuals. Important in this context is the following rule: a bag is labelled negative if **all** the instances in it are negative. A bag is labelled positive if there is **at least one** instance in it which is positive. As it is the case with *SIL* learners, the *MIL* learner produces a model during its learning phase, which then can be used to classify unseen instances. Bunescu and Mooney used this kind of learning in the following way: The starting point was that as little supervision as possible should go into the system and furthermore this knowledge should be expressable by non-experts on RE or linguistics.

The best way to produce large amounts of instances that approve or deny a certain relationship is to formulate search queries on a basis of well-known relationships. This means for example, to produce the training data for the relation *Corporate-Acquisition* the following queries have been formulated[1]:

Google * * * * * * * Youtube (1375)

Adobe Systems * * * * * * * Macromedia (622)

Viacom * * * * * * * Dream Works (323)

Novartis * * * * * * * Eon Labs (311)

Yahoo * * * * * * * Microsoft (163)

Pfizer * * * * * * * Teva (247)

whereas the first four queries should contain at least one sentence expressing the relation of a *Corporate-Acquisition*, the last two clearly do not contain this relation. The numbers behind the queries repesent the number of obtained snippets. Then the resulting snippets are stored without any change except that the entities are replaced by placeholders into positive and negative labelled files. These files are the abovementioned bags which serve as input to the learner. No sentence isolation or other linguistic technologies are applied.

To check the results the following queries have been formulated and each snippet has been manually tagged whether it contains the desired relation or not:

Pfizer * * * * * * * Rinat Neuroscience (50, 41 true positives)

Yahoo * * * * * * * Inktomi (433, 115 true positives)

Google * * * * * * * Apple (281)

Viacom * * * * * * * NBC (231)

The evaluation of this approach showed encouraging results varying between f–measures from 0.55 up to 0.72. Bunescu used a *MIL* version of Support Vector Machines (SVMs, c.f. (4)) and formulated a new Kernel function based on subsequence kernels (SSK, c.f. (12)).

---

[1]The stars between the entities mean there may be 7 other words between the entities

Furthermore, two biases have been introduced, the first consists of a relaxation of words that are correlated with either of the two named entities. In fact, those words are given lower weights. The second one concerns with words that are specific for the named entities used as starting point. Words from these elements, like "stock", or "October", are likely to occur very often in the Google-Youtube bag, and because the training dataset contains only a few other bags, subsequence patterns containing these words will be given too much weight in the learned model.

Bunsecu did several tests on the collected data with very varying results:

- SSK-MIL: This corresponds to the *MIL* formulation with modified subsequence kernel.

- SSK-T1: This is the SSK-MIL system augmented with word weights, so that the Type I bias is reduced.

- BW-MIL: This is a bag-of-words kernel and shows the performance of a standard text-classification, e.g. SVM with linear kernel, approach to the problem (Baseline).

See figure 4.1 showing the ROC curves for the relation *Corporate-Acquisition*

As we did not succeed in retrieving or reimplementing Bunescu's and Mooney's system we implemented our own approach by using a special ML technique we developed in the course of this thesis. This new approach can be used for *SIL* problems - not only for Relation Extraction, but also for general document classification - as well as for *MIL* problems.

4.2.1.1   Machine Learning based on probabilistic centroids (PCL)

The main idea for this algorithm has been inspired by *kNN* (37) and *Rocchio* (15) algorithms which have been successfully applied in Information Retrieval and Document Classification. Our innovative part of *PCL* (Probabilistic Centroid Learning) consists of a cf/idf (category frequency * inverse document frequency) and hit frequency based weighting technique, that strengthens the robustness of the machine learning algorithm in cases where the training data is noisy or unbalanced according to its category-sizes.
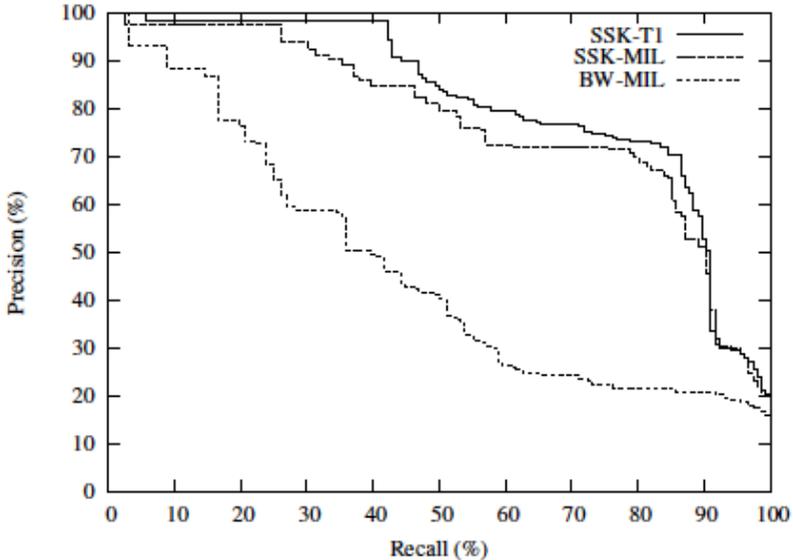
Figure 4.1: ROC curves on Corporate-Acquisiton data

Furthermore, learning and classification speed is very fast compared to other approaches.

In its supervised - single instance learning *(SIL)* - version the training basis of PCL consists of labelled documents or text. The basic work steps are as follows:

- *Learning phase*

    For each category add up all training data and build a feature vector (87) x (Table 4.1): The feature vector may consist either of - sometimes linguistically processed - tokens or of letter n-grams occurring in the text. A combination of both is possible. This process is called *preprocessing*

    Compute the norm of each category cat_i according to its size:

    $$norm(cat_i) = \sum |x_{k_i}| \qquad (4.3)$$

    where x_{k_i} denotes all k features of category i

Feature	#occ	#weighting
james	210	0.24
jim	394	0.28
clark	394	0.28
driver	129	0.18
army	77	0.15
netscape	112	0.17
robbery	58	0.13
cash	97	0.16
accident	139	0.19
.	.	.
.	.	.
.	.	.
antarctica	17	0.07

Table 4.1: Example feature vector

Weight the features in the vectors according to their number of occurrences in the category, which in principle represents the ambiguity of the features with respect to the categories:

$$w_{training}(x_{k_i}) = \#x_k \ \in \ cat_i \qquad (4.4)$$

The main advantage of this weighting scheme is that it does not use term frequencies inside of documents which makes it especially suitable for short documents like snippets (see (97)). As a result we now have the following: (1) We have the *norm* of each category, where small categories, i.e. categories with a small number of training documents and features, have lower norms than larger ones. This is important for unbalanced training data. (2) For each category we have the *weight* of a feature (0 if it is not contained). In this way the algorithm automatically filters out stop words which are prominent in each document and thus have nearly the same weight in all categories. They just do not play any role.

With this the learning phase is already completed. No further computations are neccesary in this step. This is why the algorithm performs very fast.

- *Classification phase*

  Build a feature vector of the data to be classified: The prepro-
  cessing should be the same as in the learning phase.

  Weight the features according to their relevances in the categories.
  This gives the mutual information gain - as used in the above
  mentioned *knn*–approaches - for each token and each cate-
  gory:

  $$w'(x_{cat_i}) = \max\left(0, \frac{\sum cat_i + 1}{\sum(x_k \ \in \ cat_i)}\right) \qquad (4.5)$$

  $$w(x_{cat_i}) = \max\left(0, w'(x_{cat_i}) * w_{training}(x_{k_i})\right) \qquad (4.6)$$

  and store whether the token has occurred in a category or
  not. With this value we add the entropy of each token and
  each category. In our context the entropy expresses the num-
  ber of features needed to decide the class to which the docu-
  ment belongs[2]. This resembles the computation of the (blind)
  relevance feedback as used in the above mentioned *Rocchio*–
  approaches :

  $$hit(x_{cat_i}) = \sum \left\{ \begin{array}{cc} +1 & x_k \ \in \ cat_i \\ -1 & not \ x_k \ \in \ cat_i \end{array} \right\} \qquad (4.7)$$

  For each category we then compute the relevances for each fea-
  ture in the vector of the document to be classified. We first
  multiply the entropy with the mutual information gain. Then
  we smooth that value by dividing it with the norm of the cat-
  egory multiplied with the length of the document vector. As
  mentioned this step makes the relevance computation more
  robust against unbalanced training data:

  $$\forall \ cat_i \ \ relevance(cat_i) = \sum \frac{w(x_{cat_i}) * \sqrt{hit(x_{cat_i})}}{norm(cat_i) * length(x)} \qquad (4.8)$$

  The classification is done according to the category with the best
  relevance

---

[2]In Shannons Information Theory, the entropy rate of a data source means the average
number of bits per symbol needed to encode it (see Wikipedia)

The main properties of this classification method are its simplicity, speed and robustness even for unbalanced or noisy training data. The classification result is also equipped with reliable confidence values by computing a leave one out classification for all data in the training set. Furthermore, it allows direct control on the mutual information gain of each feature in the feature space which will be important for extending the approach to solve Multiple Instance problems. See Appendix-A for the evaluation of this approach.

4.2.1.2 PCL(MIL) - Extension to Multiple Instance problems

The extension of this algorithm to solve *MIL* problems, is based on the special properties those kind of problems have. On the one hand we know that all data inside negative bags are really negative, on the other hand there is at least one example in the positive bags that is really positive - but we do not know the true positive examples. Hence the task for the ML approach is to seperate the false positives from the true positives in the positive training set. The resulting algorithm based on the above method works as follows:

- *Learning phase*

    Add up all positive and negative bags to one big positive and negative feature vector.

    Process the Learning Phase of the *SIL* approach.

- *Rebalancing phase*

    Compute a leave-one-out calculation (52) to identify false positive examples.

    For the set of false positive training examples calculate the features with the highest relevance for the positive bag.

    Reweight some percentage of the features in the positive feature vector.

    Recompute the leave-one-out calculation using the new weighting and relabel false positive examples to negative. The result is a smaller set of false positives examples that have been originally labelled as negative. As side effect, the number of false positives inside the positive bags is reduced too.

    Repeat the two steps several times[3].

---

[3]A higher number of iterations shifts the *MIL* towards higher precision

> Reorder the trainings bags according to the labels asigned dur-
> ing the last leave-one-out calculation and repeat the whole
> process (again several times).

See Appendix B for the evaluation of this approach and Appendix C for
more results when using linguistic preprocessing.

4.2.2   Discussion

The results, especially shown in Appendix C, may lead to the conclusion
that this kind of Relation Extraction is very much usable for the whole
system. Unfortunately after implementing the online component into
the system we saw strange effects[4]:

1. Obvious relations between two *NEs* have not been extracted be-
   cause the classifier was trimmed towards precision instead of recall.

2. Trimming the classifier more towards recall resulted - depending
   on the desired relation - in immediate misclassifications without
   increasing the number of right classifications.

Going through the collected training data and the steps of the learning
phase, we realised that more than 70% of the true positive examples are
expressed in an identical way. So for example the query "Google * *
* * * * * Youtube" was expressed as: "Google [has]acquired Youtube
[for the amount of $1.65 billion]" in more than 80% of the snippets (the
brackets denote that this part is optional). The query "Yoko Ono * * *
* * * * John Lennnon" was expressed as: "Yoko Ono, [former] wife of
[Beatle] John Lennon" in more than 77% of the snippets. Other queries
showed the same effects. This means in summary:

1. The training set, as well as the test set, is way too unbalanced:
   The classifiers could not cope with overfitting problems, i.e. some
   linguistic expressions could not be learned because others where
   too dominant. Other examples belonging to the same relation type
   (for example Corporate-Acquisition) also had dominant linguistic
   expressions. Unfortunately, not the same.

---

[4]Please note that we now speak of the observable behavior of the system running
on the iPad. This does not mean that the PCL(MIL) approach does not work in
general...

2. Possible syntactic surfaces do not exist in the training examples: We observed that for some search queries there was a complete lack of different ways how to express something. Like in the above mentioned example "Yoko Ono * * * * * * * John Lennon" we could not find the following expressions: "Yoko Ono's husband, John Lennon", "Yoko Ono married John Lennon", "Yoko Ono and John Lennon married .... ". This means the classifier will be unaware of these expressions in future examples, for example "Sinead O'Connor married Barry Herridge"

3. Use of subjunctive in snippets: Subjunctive expressions could be distinguished from indicative ones in high precision setting only. This means a loss of recall that on the other hand led to problems described in (2).

   So either "Microsoft could buy a company like Netflix in 2012" or "Microsoft, Dell May Acquire Research In Motion" would be classified as *Corporate-Acquisition(Microsoft,Netflix)* or *Corporate-Acquisition(Microsoft,Research in Motion)* or "Google Inc. is snapping up YouTube Inc. for \$1.65 billion in a deal" will be classified as *Corporate-Acquisition(Google,YouTube)* - but not both!

Although we could try to cope with these problems using appropriate linguistic preprocessing, we cannot be sure to really catch the problems and we would risk an additional source of possible errors. Furthermore, we would be very language-dependent.

4.2.3   Relation Extraction using Collocation Chains

Starting from the theory described in 3 we extended the *CPD* model to a *CTD* - a chunk triple distance - model. In contrast to observing the statistics of *NP* chunks only, we now also look at (parts of) the verb group *VG* lying in between the NP chunks. In general, for two NP chunks, a single chunk-triple-distance element stores the distance between the first NP chunk and the VG, as well as the distance between the VG and the second NP chunk. For example (Peter, loves, Mary, 1, 1) is different from (Peter, loves, Mary, 3, 1) and (Mary, is loved by, Peter, 1, 1). Please note that this process again meets our main requirements, i.e. to be fast, on-demand, up-to-date and indicative.
The *CTD* model requires some more effort for not running into the sparse data problem. In fact it is necessary to use a fuzzy strategy to perform

the match between VGs in a *CTD*. During our research we noticed that
the VGs are lot more often expressed using verb synonyms and adverb
synonyms[5]. Examples for this are: "Justin Bieber has recently received
the award X by (person) Y" vs. "Before Justin Bieber started his concert
he received the award X by (person) Y" vs. "Justin Bieber was very
happy to receive his new trophy X". Hence the construction of the *CTD*
model is a bit more complicated than the construction of the *CPD* model.

4.2.3.1   Construction of the *CTD* model

Starting from the motivation as in chapter 3, i.e. to assume that a *cpd*
element with high frequency covers some syntactic co-relation, and to
conclude from structural dependencies possible semantic relationships,
our process is pretty much the same: (1) Retrieval of N Web snippets, (2)
linguistic analysis of the Web snippets, i.e. sentence detection, tokenizing
and POS Tagging, (3) computation of the chunk triple distance model
CTD and (4) computation of the topic graph. The really challenging
part is point (3) as this time we not only compute statistics between
two NGs, but we rather compute NG* - VG* - NG* triples[6], with a
matching strategy that contains some *penalties*. If the resulting score
of two triples is below a threshold, we merge them. Merging means (1)
recompute the statistics and (2) use the chunks with higher numbers of
occurrences in the merged triple. Fed with two possibly matching *CTDs*
(CT1[] and CT2[]) the algorithm works as follows:

```
// The fuzzy matching algorithm with penalty
10 let CT1[] and CT2[] chunk triples;
20 if (CT1[] == CT2[]) return true;
30 else {
30   let NG_left_1 = CT1[0]; let NG_left_2 = CT2[0]
// the NG left to VGs
40   let VG_1 = CT1[1]; let VG_2 = CT2[1]
// the VG between the NGs
50   let NG_right_1 = CT1[2]; let NG_right_2 = CT2[2]
// the NG right to VG
60   let PENALTY = 0;
70   if (NG_left_1==NG_left_2 && NG_right_1==NG_right_2) {
80     let NewVG = VG_1 + '' OR '' + VG_2;
90     return new CT(NG_left_1,NewVG,NG_right_1);
```

---

[5]At least this is the case in English and German. As we are handling with snippets
we usually do not have to deal with long distance dependencies, at least not in all
cases of statistical relevant relations.

[6]The asterisks mean that these NG and VG might not necessarily be found in this
form in the original snippets.

```
100  }
110  PENALTY += computePenaltyNG(NG_left_1,NG_left_2);
120  PENALTY += computePenaltyVG(VG1,VG2);
130  PENALTY += computePenaltyNG(NG_right_1,NG_right_2);
140  let TOKNUM = (len(CT1[]) + len(CT2[]))/2;
150  if (PENALTY-TOKNUM<=0) {  // the condition depends
                              // on the size of NG or VG
160    return intersectCT(CT1[],CT2[]);
170  } else return NIL;  // no match possible
```

So if two chunk triples (CT) match exactly, we return *true*, and the *CTD* construction process can simply proceed (20). Otherwise we decompose the CTs into the NGs left and right to the VG and the VG. If both NGs match exactly we return a new CT containing both VG *ORed*; else we compute *penalties* according to the following listing. If the number of words occuring in the CTs is greater than the computed penalty we return the intersected CT[7]. This means the syntactic difference of the NGs and VGs of the two CTs is small enough to be semantically similar.

```
// the method computePenaltyNG(G1,G2);
10 let G1 and G2 the word groups to be compared
20 let N1=nouns_in(G1); let N2=nouns_in(G2);
30 let Adj1 = adjectives_in(G1); let Adj2 = adjectives_in(G2);
40 let Det1 = determiners_in(G1); let Det2 = determiners_in(G2);
50 if (intersection(N1,N2)==0) return infinite;
60 let PenaltyN = union(N1,N2)-intersection(N1,N2);
70 let PenaltyAdj = union(Adj1,Adj2)-intersection(Adj1,Adj2);
80 let PenaltyDet = union(Det1,Det2)-intersection(Det1,Det2);
90 let PENALTY = 3*PenaltyN + 2*PenaltyAdj + 1*PenaltyDet;
100 return PENALTY

// the method computePenaltyVG(G1,G2);
10 let G1 and G2 the word groups to be compared
20 let V1=verbs_in(G1); let V2=verbs_in(G2);
30 let Adv1 = adverbs_in(G1); let Adv2 = adverbs_in(G2);
50 if (intersection(V1,V2)==0) return infinite;
60 let penaltyV = union(V1,V2)-intersection(V1,V2);
70 let PenaltyAdv = union(Adv1,Adv2)-intersection(Adv1,Adv2);
90 let PENALTY = 3*penaltyV + 2*PenaltyAdv;
100 return PENALTY;
```

---

[7]intersection will be done on the NG and VG level.

If there is no match between nouns in the NG or verbs in the VG, the penalty gets infinite, which means the CTs do not match (50). Otherwise, penalties are assigned if there are words that do not match in the word groups (60-80). The final penalties are computed on the basis of a syntactically typed and *empirically determined factors*. However, although these values brought the best results in internal evaluations they are still not finally determined, if ever : the noun and verb penalties get a factor of three, the adjective and adverb penalties get a factor of two and determiners get a factor of one. This means, we distinguish explicitly between low, medium, and high penalties. Editing a low penalty word can be done once in a chunk consisting of 2 words, twice for 3 word chunks, e.g. "the (good) husband" vs. "a (good) husband". Medium penalty words can be edited once in a 3-word-chunk, twice in a 5-word-chunk, e.g. "very good husband" vs. "really good husband". High penalty words can be edited once in a 4-word-chunk, e.g. "a very good husband" vs. "a very good man". Hence, as the chunks usually do not exceed 5 words, editing nouns, proper names or verbs is possible in principle, but only once, editing adjectives or adverbs only twice[8]. Furthermore, matching verb groups also is harder as usually they are much shorter. However, there is one special rule we introduced in order to make the resulting topic graph more compact: If both NGs have *penalty* = 0, i.e. they completely match, then we perform the merge, but we keep the two VGs in parallel in the new chunk, e.g. "Justin Bieber has been awarded with the XY prize" and "Justin Bieber got the XY prize" will be merged to "Justin Bieber (has been awarded with / got) the XY prize" . Table 4.2.3.1 shows possible candidates and merging results. The first three examples "jim clark drove lotus powered by ford", "jim clark successfully drove lotus powered by ford", and "jim clark won in lotus powered by ford" merge to "jim clark won in / drove lotus powerd by ford". The second two examples "jim clark was a famous f1 driver in the 1960's era" and "driver jim clark was famous in the f1 in the 1960's era" merge to "jim clark was famous in the f1 in the 1960's era". Finally, the examples "jim clark lost life on 7th april 1968", "jim clark was born 1936 in scotland" and "jim clark wins grand prix in francorchamps" do not merge, even not pairwise.

---

[8]... determiners usually are no problem as there are no chunks containing more than 2 determiners.

jim clark	drove	lotus powered by ford	59;22;10
jim clark	successfully drove	lotus powered by ford	59;11;10
jim clark	won in	lotus powered by ford	59;8;10
merges to			
jim clark	won in / drove	lotus powered by ford	59;41;10
jim clark	was	a famous f1 driver in the 1960's era	59;245;2
driver jim clark	was	famous in the f1 in the 1960's era	28;245;4
merges to			
jim clark	was	famous in the f1 in the 1960's era	59;245;4
jim clark	lost life on	7th april 1968	59;11;11
jim clark	was born	1936 in scotland	59;7;2
jim clark	wins	grand prix in francorchamps	59;34;5
does not merge			

Table 4.2: An excerpt of the CTD model of Jim Clark showing candidates that can possibly be merged. The numbers show the number of occurrences of the groups in the retrieved snippets.

4.2.4 Computing the Chunk Triple Distance Model CTD_M

The computation of the extended CTD_M is straightforward: we temporarily reduce the triples back to tuples and use the same formula as we did to compute the CPD_M in section 3:

$$PMI(cpd) = (\sum_{k=1}^{n} \frac{(x_k)^k}{k}) - log_2(p(c_i) * p(c_j))$$

$$, where\ x_k = \frac{freq_k}{\sum_{k=1}^{n} freq_k}$$

This can easily be done because of our triple merging strategy described above:

- There will never be more than one triple containing the (approximately) same NGs. They are reduced to one triple (see first example above).

- Omitting the verb information does not change the statistics.

- The original *CTDs* are created from the *CPDs* and hence a final CTD_M can be built.

In the next section we will show evaluation experiments.

4.3   Evaluation

In chapter 3 we compared our *NEI* approach with well-known systems that perform a similar task although with different prerequisites. Unfortunately, things are not as easy in *RE* as there are no off-the-shelf componenets available for this task. Although there exist a couple of unsupervised approaches the prerequisites are quite different compared to our case. As shown in section 4.1.3, those systems need a totally different trainingset, e.g. we did not find any unsupervised approach working on snippets only, and hence the results will be of course different in many respects. Hence this time we concentrated on randomly picking results from several test runs and checking the correctness of the extracted relations manually. This approach is most convenient to evaluate unsupervised methods, e.g. in (27), (23), (80), etc. To gather enough examples we ran the system in a batch mode using lists of named entities as the source for our search queries. In our experiments we took the entries of "List of celebrity guest stars on Sesame Street"[9] (*Set*1) and the "List of film and television directors"[10] (*Set*2). These are the same lists as we will use in chapter 5, which introduces the Query Disambiguation *QD*. In a next step we randomly took 300 extracted relations, and the snippets from which they have been extracted, and let them be checked independently by two members of our lab, who are not connected with this thesis. Each evaluator first judged whether the extracted relation is correct or not. A relation r is considered to be correct if there is some pair of entities X and Y such that (X, r, Y) is a relation between X and Y . For example, (diana krall, reschedule, a february 2012 concert) extracted from the snippet "according to a press release issued by ruth eckerd hall earlier today, a scheduling conflict has forced diana krall to reschedule a february 2012 concert" contains a correct relation. Also (diana krall, was born on, november 16) extracted from the snippet "diana krall was born on november 16, 1964 in nanaimo, british columbia, canada. diana and her younger sister michelle grew up in a musical household." is considered to be correct, although it is not complete or

---

[9]http://en.wikipedia.org/wiki/List_of_celebrity_guest_stars_on_Sesame_Street
[10]http://en.wikipedia.org/wiki/List_of_film_and_television_directors

underspecified (but still makes sense). (madeline kahn, was, a similar talent) extracted from "madeline kahn was a similar talent who worked four times with mel brooks (young ... 'american idol' to 'the vampire diaries': the top-rated tv shows on each network ..." is also considered to be correct although it does not make very much sense. We consider it to be correct because it serves as a helper to promote the complete relation according to our merging strategy. (jessica alba, shows off, itty-bitty waist in la) extracted from "jessica alba shows off her itty-bitty waist in la ... whose weeny waist? 8:15am, oct 13, 2011 ... all eyes were on this tiny waist at the american film ..." is correct although it is not complete. In contrast this time the information is not contained in the snippet but it is similar to the case before.

(jessica alba, is, mexican american) extracted from "13 helpful answers below ... jessica alba father is mexican american and the mother is french danish... but jessica alba" is not correct as the object "mexican american" does belong to "jessica alba father" instead of "jessica alba" - although in this case it might be the truth, in general it is not. Table 4.3 summarises the results of this analysis.

# queries	407
# snippets	81.400 (snippets limited to 200 per query)
# relations	2013 (relations limited to 9 per topic)
# random	300
Correct complete	173
Incorrect	23
Correct incomplete extraction	18
Correct incomplete in snippet	23
Correct but no sense	63

Table 4.3: Batch relation extraction statistics

The results show very convincing numbers. The accuracy rate is 88% for the strict view, i.e. correct and complete vs. incorrect, 70% for the more relaxed view[11]. We manually inspected some snippets whether they contain undetected relations and we found out that the recall rate is about 80% for those relations that are entailed in correct formulated sentences. From section 3.2 we have learned that the ratio between complete and incomplete sentences varies between 50% and 70%.

---

[11]..or 58% for correct and complete vs. all

This means the real recall is roughly around 40% to 60%. However, we believe that this still is a very good result, if you keep in mind that not all sentences in the snippet, complete or incomplete, describe a new relation and there are also enough relations that are expressed by complete and incomplete sentences in the same corpus.

4.4 Background Knowledge for Facts and Relations

In the previous sections we extracted concepts and relations between the concepts on a purely statistical way. As already outlined these methods heavily rely on redundancy inside the snippets. The advantages have become clear: speed, actuality, and domain independence. On the other hand there are also disadvantages: often enough snippets do not contain the necessary background knowledge or they are just not prominent enough on the web so the redundancy is not given. Sometimes only actual or very recent facts are present in the found snippets, whereas necessary knowledge is not accessible. However, in the current search engines like BING, Google, or DuckDuckGo, we see more and more that especially this knowledge is in the first places of the result list (see Fig. 4.2, Fig. 4.3, and Fig. 4.4) - but only once so we cannot use redundancy. Either it is the correlated Wikipedia article or maps - in case the query contains a place or institution - or pictures - especially when it comes to person names. Although this is a very useful extension to common search engines used on ordinary computers, we consider this kind of knowledge-presentation can be improved when it comes to mobile, especially small screen devices.

In general we need to have a way to get specific background knowledge about the topic in focus. And we again want to present this in the same exploratory way to the user as we did before. The main idea to acquire background knowledge is to connect to knowledge bases, that have been built manually, and of course they should be maintained and kept up to date, i.e. to follow the path of the great search engines. Hence we connected the Wikipedia database to our system. We also tested DBpedia (7) as a background source. However, it turned out that currently it contains too much and also redundant information. For example, the Wikipedia facts for "Justin Bieber" contain eleven basic relations, whereas DBpedia has fifty relations containing lots of redundancies, which are really hard to filter out.
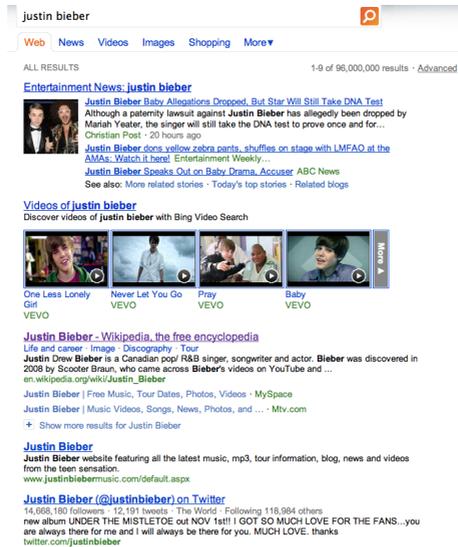
Figure 4.2: Search result for query *Justin Bieber* on BING
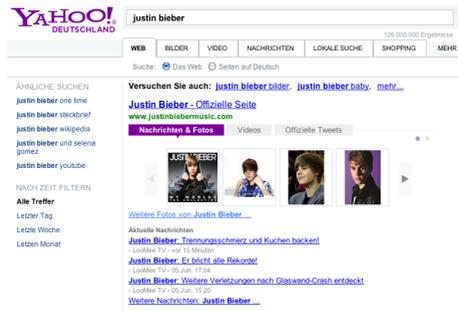
Figure 4.3: Search result for query *Justin Bieber* on Google

Furthermore, currently there does not exist an ontology for the possible relations and relation names in DBpedia. This means that on the one hand there is no fixed set of relations that are used to characterise instances of the same category.
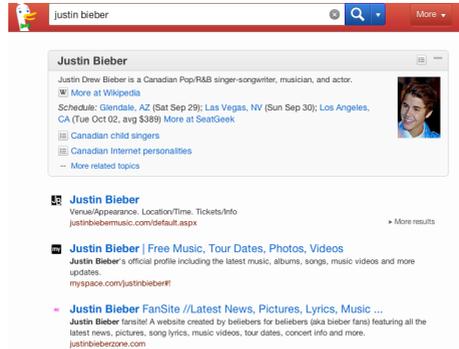
Figure 4.4: Search result for query *Justin Bieber* on DuckDuckGo

For example the relation "birthplace(person,place)" or
"dateOfBirth(person,date)" are not always given for instances of the
category "person". On the other hand the same semantic relationships
are expressed by different predicates like "birthplace", "placeOfBirth",
etc. and sometimes they even occur together in the same instance. We
decided in favour Wikipedia, despite the fact that same problems occur,
but since Wikipedia pages are manually constructed and maintained, we
believe that such errors will be repaired during time.

4.4.1 Wikipedia Infoboxes

Currently a wikipedia page is split into several sections. Usually an
article starts with a title followed by a short abstract. After that you will
find a table of contents - in case the article is long enough - and then,
according to this table, follows the whole document split into several
sections. On the right side in most cases there is a so called *infobox*
containing the most general and important facts of the subject, at least
to the authors' opinions.

As all Wikipedia pages are represented in XML and the infoboxes are
part of such XML documents, they can be easily extracted using XML
parsers. For example, the Wikipedia infobox for Justin Bieber contains
eleven basic relations: *Birthname, Name, Origin, Birthdate, Occupation,
Background, Genre, (record) Label, URL (of website), Associated Acts,
Years active* (See Fig. 4.5). Please note that the relations in infoboxes
are given by the Wikipedia user who enters the article.

Background information	
Birth name	Justin Drew Bieber
Born	March 1, 1994 (age 17)[1] London, Ontario, Canada
Origin	Stratford, Ontario, Canada
Genres	Pop, R&B, teen pop[2][3][4]
Occupations	Singer, musician, actor
Instruments	Vocals, guitar, piano, percussion,[5] trumpet[6]
Years active	2009–present
Labels	Island, RBMG
Associated acts	Usher
Website	justinbiebermusic.com

Figure 4.5: The Wikipedia infobox for *Justin Bieber*

Hence the articles

1. contain relationships that are specialised to the topic in focus and not just for the kind of topic like "person", "place", "organisation". In our example we find "record label" or "associated acts", which will not be found in the infobox for "Barack Obama".

2. contain the same relationships as other articles, but with different names. For example the relation "birthplace" is also written as "place of birth", "origin", etc.

3. vary in the number of relationships given. From only four relationships for actor "Jim Carey" to 699 relationships for the Chinese figure skater "Chen Lu"[12]

Concerning (1) and (2) we simply leave the names of the relationships as they are.

---

[12]The relationships contain all victories, skating figures, dates and places of competitions, etc.

As we do not plan to do more than displaying them, we think they
should be understandable for the user (as they were created by other
users before). Point (3) clearly is a problem as we do not have the space
to present more than 12-15 relationships on our mobile device. For
this we defined a set of basic relationships we always want to present
if available. We did this for following types of entities considering the
main variations in the naming :

- *person*: name, birthdate, birthplace, relatives, occupation, place
  of death, date of death, alma mater, homepage

- *place*: name, city, country, capital, language, governor (or similar),
  number of citizens, foundation date, homepage

- *organisation*: name, founder, date of foundation, business, ceo,
  location, homepage

We downloaded a snapshot of the whole English Wikipedia database
(images excluded), extracted the infoboxes for all articles, if available,
and built a Lucene index running on our server. We ended up with
1,124,076 infoboxes representing more than 2 million different searchable
titles. The average access time is about 0.5 seconds. Currently, we only
support exact matches between the user's query and an infobox title in
order to avoid ambiguities.

In Fig. 4.6 and Fig. 4.7 we present the results of the unsupervised and
knowledge-based *RE* approeaches respectively. The knowledge-based
approach shows some general information about Justin Bieber like his
URL, his label, the birthname etc. whereas more recent and gosspi-like
topics like "Justin Bieber is a teenage singing sensation" or "J.B. arrives
... 2011 american music awards".
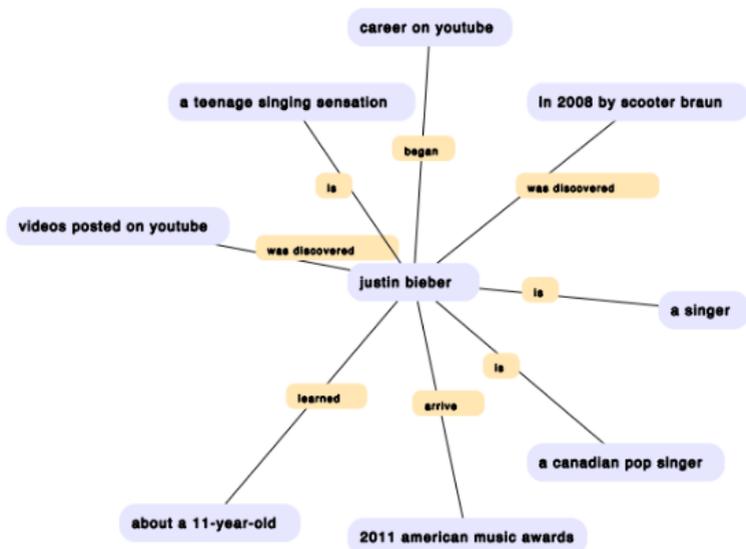
Figure 4.6: Topic graph of *Justin Bieber* using unsupervised *RE*
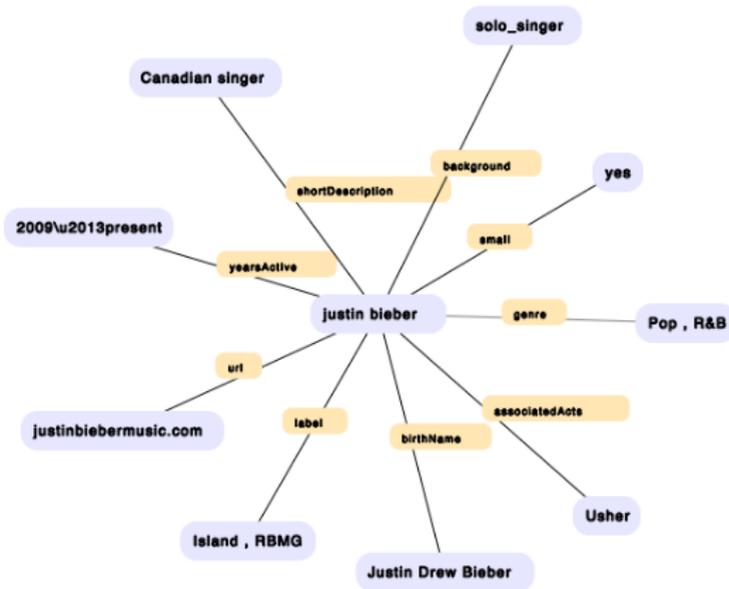
Figure 4.7: Topic graph of *Justin Bieber* based on Wikipedia Infoboxes

5

Disambiguation of Topics

Quite a frequent case of search on the web is the search for named entities, i.e. famous personalities, places, events etc. In fact, as our system relies on redundancy, the query for named entities is well-suited to be presented in topic graphs. However, the success of working with the system heavily depends on the quality of the associated topics presented in the topic graph. One source of insufficient quality is the possible ambiguity of the search query. For example, if the user looks for the person *Jim Clark*, she/he thinks either of the racing driver or the Netscape founder[1]. As the retrieved snippets may contain information about both persons, the topic graph will show associated topics that lead the user into wrong directions while exploring (Fig 5.1).

Hence there is a need of leading the user to the right path so that she gets not frustrated by meandering in the search results. In (82) it is reported that between 7% and 23% of frequent queries in the logs of two search engines are ambiguous. This does not only include queries that are ambiguous as the contained words have several meanings like the word "bank", "jaguar", etc., but also queries that may lead to a different solution space of a search engine's document pool. Hence in this context the disambiguation task is strongly correlated to the automatic determination of the user's intension or goals. For the search engines used today these tasks become very tricky to solve, since often enough both problems are correlated and occur at the same time for users' search queries.

---

[1]... or the baseball player, the football player, the bank robber, the film editor, the war hero,...

Figure 5.1: Ambiguities when searching for Jim Clark.

Regarding our solution for exploratory search on mobile devices, the disambiguation part is less hard to solve. As described in the earlier chapters, our system is designed in a way that the user may decide on the answers from the associated topics that are presented in the topic graph[2]. Hence our solution divides the above mentioned problems, query disambiguation and user's goal determination, in a natural way: The user himself decides on the goal using the associated topics around the original query. The system needs to determine the right sense to the query in order to present the right associations.

5.1 Methods for Detecting Ambiguities

For the query sense diambiguation task we focused on two main directions: (1) knowledge-based methods, (2) statistical methods. The goal was to find out which method provides the best recall and precision values. For this, we started with several queries that we know have lots of senses. Please note that we did our very best to find all senses, but there may exist more than we could imagine:

- Blythe

    - This is Blythe, a photo book by Gina Garan featuring the doll Blythe
    - Blythe Katherine Danner, an American actress, mother of Gwyneth and Jake Paltrow
    - Blythe, a city in Riverside County
    - Blythe, a city in Burk County
    - Blythe, a river in the English Midlands
    - Blythe, a fashion doll created in 1972
    - ..... (57 different senses)

---

[2].. or the stack of topics in the smartphone version

- Jim Clark

  - Jim Clark, Scottish Formula One world drivers' champion in the 1960s
  - Jim Clark, founder of Silicon Graphics and Netscape
  - Jim Clark (film editor), Oscar-winning editor of The Killing Fields
  - Jim Clark (American football), American football player
  - .... (24 different senses)

- Phantom of the Opera

  - Le Fantome de lOpera (English: The Phantom of the Opera) is a novel by French language/French writer Gaston Leroux.
  - The Phantom of the Opera is a musical theatre/musical by Andrew Lloyd Webber, based on the French novel The Phantom of the Opera/Le Fantôme de lOpéra by Gaston Leroux.
  - The Phantom of the Opera is a 2004 film adaptation of Andrew Lloyd Webbers 1986 musical The Phantom of the Opera (1986 musical)/of the same name, which in turn was based on the French novel The Phantom of the Opera/Le Fantôme de lOpéra by Gaston Leroux.
  - The Phantom of the Opera is a 1925 American Silent film/silent horror film adaptation of the Gaston Leroux The Phantom of the Opera/novel of the same title directed by Rupert Julian.
  - ..... (16 different senses)

- Mission to Mars

  - Mission to Mars is a 2000 science fiction film directed by Brian De Palma from an original screenplay written by Jim Thomas, John Thomas, and Graham Yost.
  - Mission to Mars was an attraction located in Tomorrowland at Disneyland and at Walt Disney Worlds Magic Kingdom.
  - Mission to Mars is a 1955 science fiction novel by Patrick Moore, published by Burke.
  - Backyardigans: Mission to Mars is a steel roller coaster/steel family roller coaster/family roller coaster located at Movie Park Germany.

- Sven Schmeier

  - Sven Schmeier, myself
  - Sven Schmeier, an artist from Germany
  - Sven Schmeier, drummer of the band around Rio Reiser
  - Sven Schmeier, mountainbiker from Switzerland

5.2 Finding Ambiguities using Encyclopedic Knowledge

Our first approach is based on Wikipedia and aims on disambiguating the search query before starting the information retrieval process. Beside the fact that Wikipedia is known to cover a huge number of possible senses for a very large number of topics, we also consider Wikipedia as a suitable means of a human–computer interface in the sense that both the human and the computer can directly communicate in natural language (NL). We again indexed a snapshot of whole Wikipedia into a Lucene index. This time we did not extract the infoboxes, but we indexed the title and the first two sentences of each article. The index contains 2,999,597 articles with 4,320,497 different terms and has a size of 7.63 GB on the disc. The average access time is about 0.5 seconds. The original search query is used to search in the title field only. All search terms must occur in the title field - if they pass Lucene's *SimpleAnalyzer*, which lowercases and tokenises the query and the articles. Hence the query for "Jim Clark" also matches "James (Jim) Clark", "Clark Jim", "The famous Jim Clark", etc. For each query we count the number of found articles which represent different instances or meanings. See Table 5.1 for the results.

Query	#senses	#articles found	precision	recall
Blythe	57	57	1.0	1.0
Jim Clark	13	12	1.0	0.92
Phantom of the Opera	16	16	1.0	1.0
Mission to Mars	4	6	0.66	1.0
Sven Schmeier	4	0	1.0	0

Table 5.1: Disambiguation performance using Wikipedia

In summary the results show:

1. Disambiguation with knowledge is possible only if the search query can be found in Wikipedia. As the person "Sven Schmeier" has no article in Wikipedia, disambiguation is not possible.

2. In case there is a match, disambiguation is very accurate. The precision is very high, which is very likely for manually built approaches, which are double checked by reviewers as it is the case in Wikipedia. The recall is high too mainly because a lot of people add content on a daily basis (which is again double checked before released). Also the way we use the local search engine Lucene seems to be very effective. Recall values below 1.0 are for example caused by the fact that the "Netscape" founder "Jim Clark" is written as "James Clark" in Wikipedia.

3. The labeling of the found clusters is done simply by presenting the first two sentences of the Wikipedia article. Usually there is enough information to understand whether the topic of interest has been found or not.

As these results were very encouraging we equipped our system with this approach in the following way: To begin with we will use the topic "Jim Clark" as a running example and describe the process from the user's point of view.

A user starts her exploratory search by entering a query q consisting of one or more keywords used to represent the topic in question (in our example, just the two words "Jim" and "Clark"). Instead of directly computing and presenting a topic graph for q, possible senses of q are identified and enumerated by using our knowledge-based disambiguation approach. This means that the search strategy determines all possible senses (i.e., Wikipedia pages) that entail q as part of the Wikipedia title (i.e., the NL name of the concept described in the Wikipedia page). All found readings are then sorted according to the algorithm explained below and presented to the user who should select her preferred one.

Let us assume that the user selects the "British racing driver" sense, then the major content of the Wikipedia concept (basically the first sentence s of a Wikipedia page which usually defines the concept) is used to create a new expanded query q' from q and s.

Now, using q' an initial topic graph is computed on the fly from a set of Web snippets that have been collected by a standard search engine (currently, we are using Bing[3]).

The topic graph is then displayed on a tablet computer (in our case an iPad) as touch–sensitive graph. Note that if the user expands a node, the new query sent to the search engine is created from the label of the selected node and the "sense" information s created above from Wikipedia. Thus, each search triggered by a selected topic node is guided towards the user's preferred reading. This is why we also call our approach *guided exploratory search*.

In detail our query disambiguation algorithm works as follows:

```
10  let Q=user's query;
20  let TG=produce_TG(Q); // initial topic graph TG
30  let LI=Lucene Index;
40  let q[]=SA(tokenize(Q));
50  let query=(title:+q[1] ... +q[n]);
60  let results[]=search(LI ,query);
70  if (num(results[]) > 1) {
80    let ass[]=SA(associated_topics(TG));
90    let Qexp=(title:+q[1] ... +q[n]) OR
          (body:+ass[1] ... +ass[m]);
100   let docs[]=search(LI, Qexp);
110   if (user chooses docs[i]) {
120     let s=definition_sentences(docs[i]);
130     let TGnew=produce_TG(Q + s);
140     return TGnew;
140 }} else {
150     return TG;} // return initial TG
```

We start to compute an initial Topic Graph TG with the original user query (20) using the TG construction process described in section 3[4]. The steps (30) to (60) then compute the degree of sense ambiguity using Wikipedia in the following way. Firstly (40), we tokenise the query and apply Lucene's SimpleAnalyzer SA, which lowercases and tokenises the query. In a next step Lucene retrieves all documents that entail all tokens of the query in the titles of the articles (50+60). This way it is guaranteed that we find all instances for an entity.

---

[3] http://www.bing.com/

[4] This topic graph is just for sorting the Wikipedia articles (see below) and will not be shown to the user

The title of an article uniquely identifies each instance, because it typically describes the entity in the article and is further qualified by parenthetical expressions. For example, the query for "Jim Clark" also matches "James (Jim) Clark", "Jim Clark (sheriff)", "Jim Clark (film editor)", etc. If only a single title matches or if there is no match at all, we return the initial topic graph TG (150). Otherwise (70) we know that the query matches different Wikipedia articles, and hence, that the query is potentially ambiguous.

In principle, we could now present the different concepts to the user just in the order determined by Lucene. However, the problem is that this ordering actually ignores the information already expressed in the initial topic graph TG. It could happen that the higher ranked elements in the ranked list are unrelated with the information used by the search engine and covered in TG. On the other hand, the initial TG already expresses some interesting latent semantic information computed via the use of PMI, e.g. expressing that neighbouring nodes of a node n are semantically more related to n than nodes with larger distance. Thus, in order to achieve a more user query and TG related ordering, we perform the following steps (80) to (140). Firstly, we perform a query expansion by adding topics from TG that are determined by a $1NN$ strategy (80) to the original query, i.e. we use only the directly associated topics. In the next steps (90 ff) we again formulate a query against our Wikipedia index. This time we use the associated topics to also search in the articles' body. The result is an ordered list according to the main topics in the initial TG where the most probable meaning is listed first. Please note that the set of retrieved articles will stay the same as we *ORed* the original query with the associated topics. The abstracts of the articles are presented to the user to choose from. We extract the most important terms (using the function definition_sentences() defined more precisely in the next listing) from the chosen article (120) and produce the final TG using the combination of the terms and the original query (130). The initial TG is thrown away.

```
10 let first=article.firstSentence
20 let first_pos=POS_Tagging(first)
30 let sep=first_pos.indexOf(((is|was)(a|the)));
40 let isa_part=substr(first_pos,sep);
50 return filter_pos(''N'',isa_part);
```

According to Wikipedia article guidelines[5] an article usually contains a definition in the first sentence (10). Therefore we first tag the sentence with *PoS* information (20). If we find the definition phrases "is a", "is the", "was a", or "was the", we choose its right adjacent substring (30+40). If the definition phrase cannot be found, we choose the whole sentence. We filter out all tokens that are not tagged as nouns and return the remaining list (50).

5.3   Automatic evaluation

In the experimental evaluation we present an automatic way of how to determine the accuracy of the knowledge–based disambiguation algorithm. In a first step we use the above mentioned algorithm. Please note we evaluate really ambiguous queries only. Then we alter the original algorithm in the following way:

```
110 let right=0; all=0;
120 foreach(doc in docs) {
130  let s=definition_sentences(doc);
140  let TGnew=produce_TG(Q + s);
150  let ass[]=SA(associated_topics(TGnew));
160  let Qexp=(title:+q[1] ... +q[n]) AND
             (body:+ass[1] ... ass[m]);
170  let articles[]=search(LI,Qexp);
180  if(doc==articles[0]) {
190    right++;
200  }
210  all++;
220 }
230 final_accuracy=right/all ;
```

The idea behind this automatic evaluation is as follows: the topic graph produced, starting from a disambiguated document, results in a new Topic Graph *TGnew*. A search against the Wikipedia index using the original query for the title–field and the $1NN$ associated topics from *TGnew* should have the disambiguated document as its best result.

---

[5]http://en.wikipedia.org/wiki/Wikipedia:Lead_section#
 Introductory_text

In our experiments we took the entries of "List of celebrity guest stars on Sesame Street"[6] ($Set1$) and the "List of film and television directors"[7] ($Set2$). Furthermore, we evaluated both kinds of the topic graph construction process: Topic retrieval based on collocations only ($TopCol$) and its combination with the cluster descriptions ($TopClus$). Table 5.2 shows the results on the two datasets and the two different TG construction approaches (The first column says: 1:Set1; 2:Set2; A:TopCol; B:TopClus).

Set	All	Ambig	Good	Bad	Acc
1+A	406	209	375	54	87.41%
1+B	406	209	378	51	88.11%
2+A	1028	229	472	28	94.4%
2+B	1028	229	481	19	96.2%

Table 5.2: Accuracy of disambiguation.

5.4 Manual evaluation

To doublecheck the results of the previous section we also did manual evaluations on datasets by randomly picking results from several test runs and let two human judges check the correctness of the topics for the chosen senses. This approach is often used to evaluate unsupervised methods, cf. (29). The general setup was to count the number of correct vs. incorrect topics for a given sense. Furthermore, we gave the judges the chance to intuitively decide whether they would have followed a wrong path while exploring the solution space, i.e. the task of guiding the exploratory search would have failed. Table 5.3 shows the results. The first column denotes the kind of topic retrieval like in the automatic evaluation. The next column shows the number of examples or senses that have been checked[8]. Column 3 shows the total number of extracted topics. The combined retrieval delivers less topics but as you can see in column 4, the quality seems to have improved as the ratio between

---

[6] http://en.wikipedia.org/wiki/
List_of_celebrity_guest_stars_on_Sesame_Street
[7] http://en.wikipedia.org/wiki/
List_of_film_and_television_directors
[8] Each judge checked the same examples independently

correct and incorrect topics decreases for both testers. The last column shows whether the guidance towards topics for the chosen sence has been successful. Please note the values in the columns 3–5 are highly subjective. For the second judge a lot of tokens do not make sense in her opinion, but on the other hand she would not have followed them during exploration. Hence although she generally judged more topics not to fit she rated the algorithms original sense, i.e. guiding the search towards the right direction, as more successful than the first judge.

However, we see that the manual evaluations seem to proove the results and the method of the automatic evaluation.

Set	All	Topics	Good	Bad	Success
A	20	167	132	35	ca. 95%
B	20	145	129	16	ca. 95%
A	20	167	108	59	¿ 97%
B	20	145	105	40	¿ 97%

Table 5.3: Manaual evaluation.

5.5 Finding Ambiguities using Statistical Approaches

The main problem of the knowledge-based approach arises when the ambiguous query cannot be found in Wikipedia using our strategy. So for example the query "Famous Jim Clark" would not be found as we require all words to appear in the article's title. Even if we could cope with this using a modified, fuzzy search strategy we still would not find out ambiguities in queries that simply are not present in Wikipedia.

5.5.0.1 Seperating Senses by Statistical Clustering

Statistical clustering approaches have become very popular in recent time. They combine several advantages like speed, reasonably good disambiguation performance at least on person names, usually no manual work ist necessary (besides implementing the algorithm), and furthermore, it is relatively easy to implement it as a browser plugin or as an internet service between user and search engine (see section 5.6). The process for this task on the mobile device should again result in either

short, but telling, topics or short sentences that can easily be checked by
the user. Hence we again used the *Carrot2* clustering algorithm as de-
scribed in chapter 3.2. We first collected the snippets delivered by BING
for the search query, clustered them and after that manually inspected
the produced clusters and the related labels. Please see table 5.4 for the
detailed results. In summary the results can be interpreted like that:

1. Snippets: In all cases the found snippets did not contain all pos-
   sible senses. The main reason for this is the limitation of our
   algorithm to retrieve 1000 snippets max., which is due to perfor-
   mance reasons[9]. On the other hand less prominent meanings, for
   example the different persons behind the name *Jim Clark*, *Blythe*,
   etc., would not be in the snippets provided by the search engine at
   all without further specification

2. Disambiguation performance: The precision and recall numbers
   show (a) whether the identified clusters really have been semantic
   clusters (precision) and (b) the ratio between the clusters found
   and the clusters expected (recall). In most cases the values have
   been very low. The most likely explanation is that the clusters
   represented by the snippets are usually very unbalanced, i.e. lot of
   snippets characterize one cluster, only some snippets characterise
   the other clusters. The clustering algorithm therefore is not able
   to identify the small clusters. Furthermore, the snippets seem to
   be too small in terms of length for the task of disambiguation.

3. Quality of labels: This point is strongly correlated to the second
   point. Whenever the precision and recall values were relatively
   high the quality of labels improved. However, the labels lead to
   confusion instead of clarification and tended not to be useful as a
   description.

5.6 Related Work

There are several approaches for web query disambiguation. As men-
tioned above the task is not only to detect disambiguities in the words
of the query but also to decide the right direction in the solution space.
Some approaches like (16) try to automatically learn a user's interest
based on the click history.

---

[9]Usually the number of snippets is below 200

Query	#senses	#in snippets	precision	recall
Blythe	57	7	0.64	0.13
Jim Clark	13	8	0.53	0.62
Phantom	16	6	0.66	0.375
Mission	4	3	0.6	0.75
Schmeier	4	2	0.5	0.25

Table 5.4: Disambiguation perfomance of statistical clustering

To achieve this, they provide a three-step algorithm: (1) a model representing user's interest based on the click history; (2) a process that estimates the user's hidden interest based on the click history; (3) a ranking mechanism that reranks the search engine result on the base of (1) and (2). Other approaches like (88) or (33) follow the same principle but with different learning and ranking algorithms. Another approach is based on hyperlink structures of the web and aims for a personal PageRank that modifies the search engines' PageRanks. Examples for this approach are (43), (48) or (78). A more generalising approach consists of collaborative filtering methods. Here, the search history of groups with similar interests are used to refine the search. This method has been used in (95) or (96). In the first approach users' profiles are constructed using a collaborative filtering algorithm (8); the second analyses the correlation among users, queries, and clicked web pages. With this information future ambiguous queries are disambiguated using the correlations. The advantage for the user is the increased completeness of the search results, because the knowledge-base for the filtering process is already filled by other users - provided there are users with similar interests.

In contrast to that, there has also been much research on trying to post-process the search results using clustering algorithms. (58) propose a very promising approach for disambiguation of person names. This approach does not require user models or a learning and personalisation phase. The results from a search process are clustered by taking different document properties into account: Title, URL, metadata, snippet, context window (around the original query), context sentence, and the bag of words of the whole document. The main property of this algorithm is robustness and speed, and hence the disambiguation performance. However, it lacks the labelling or definition of the clusters. So again, the user has to check by reading at least some snippets inside a cluster (19).

Approaches that make use of Wikipedia are for example (11), (19), (40), and many more. All of these approaches have been dominated by the idea to compare the local context of a named entity and compare it to Wikipedia articles. The articles are sorted by some similarity measure and the most similar article is used to classify the named entity to the associated meaning. These approaches are pretty similar to our approach, except that we do not use any original context, but the reduced one given by the identified topics.

5.6.1   Concept Extraction

Another statistical approach is Concept Extraction (*CE* for short) by (31). As we included this system in our mobile version we give an extended summary here.

The *CE* approach aims at finding answers to definition questions from Web snippets in an usupervised way just like our *NEI* and *RE* approach. The major advantages are that it : (a) avoids downloading full documents, (b) does not need specialised wrappers that extract definition utterances from definitional websites, and (c) uses the redundancy provided by Web snippets to check whether the information is reliable or not. *CE* achieves these goals by rewriting the query in such a way that it markedly increases the probability of aligning well-known surface patterns with web snippets. Matched sentences are therefore ranked according to three aspects: (a) the likelihood of words to belong to a description, (b) the likelihood of words to describe definition facets of the word being defined, and (c) the number of entities in each particular descriptive sentence. For this ranking purpose, *CE* takes advantage of a variation of Multi-Document Maximal Marginal Relevance and distinguishes descriptive words by means of Latent Semantic Analysis (LSA), cf. (53).

Potential Sense Identification   An important feature of *CE* is a module that attempts to group descriptive utterances by potential senses, checking their correlation in the semantic space supplied by LSA. This is the reason why we can use *CE* for disambiguation of the concept in question by clustering the extracted facts according to some hidden semantic relationship. And again, the final disambiguation is done by the user.

There are many-to-many mappings between names and their concepts. On the one hand, the same name or word can refer to several meanings or entities. On the other hand, different names can indicate the same meaning or entity. To illustrate this, consider the next set S of descriptive utterances recognised by the system:

1. John Kennedy was the 35th President of the United States.

2. John F. Kennedy was the most anti-communist US President.

3. John Kennedy was a Congregational minister born in Scotland

In these sentences, "*US President John Fitzgerald Kennedy*" is referred to as "*John Kennedy*" and "*John F. Kennedy*", while "*John Kennedy*" also indicates a Scottish congregational minister.

CE disambiguates senses (*sense* is one meaning of a word or one possible reference to a real-world entity) of a topic δ by observing the correlation of its neighbours in the reliable semantic space provided by LSA. This semantic space is constructed from the term-sentence matrix M (by considering all snippets as a single document and each snippet as a sentence), which considers δ as a *pseudo-sentence*, which is weighted according to the traditional *tf-idf*. *CE* builds a dictionary of terms W from normalised elements in the snippet document S, with uppercasing, removal of html-tags, and isolation of punctuation signs. Then *CE* distinguishes all possible unique *n-grams* in S together with their frequencies. The size of W is then reduced by removing *n-grams*, which are substrings of another equally frequent term. This reduction allows the system to speed up the computation of M as UDV' using the *Singular Value Decomposition*. Furthermore, the absence of syntactic information of LSA is slightly reduced by taking strong local syntactic dependencies into account.

For the experimental evaluation a baseline system was implemented, in which 300 snippets were retrieved by processing the input query (the topic in question) using the same query processing module as the one used in *CE*. The baseline splits snippets into sentences and accounts for a strict matching of the topic in question. In addition, a random sentence from a pair, that shares more than 60 % of its terms, and sentences that are a substring of another sentence were discarded. The baseline and *CE* were then tested with 606 definition questions from the TREC 2003/2001 and CLEF 2006/2005/2004 tracks. Overall, *CE* consistently outperformed the baseline.

The baseline discovered answers to 74% of the questions and *CE*
to up to 94%. For 41.25% of the questions, the baseline found one to
five descriptive sentences, whereas *CE* found 16 to 25 descriptive sen-
tences for 51.32% of the questions. More specifically, results show that
CE finds nuggets (descriptive phrases) for all definition questions in the
TREC 2003 set, contrary to some state-of-the-art methods, which found
nuggets for only 84%. Furthermore, *CE* finds nuggets for all 133 ques-
tions in TREC 2001 question set, in contrast to other techniques, which
found a top five ranked snippet that conveys a definition for only 116
questions within the top 50 downloaded full documents. Concerning the
performance of the sense disambiguation process, *CE* was able to distin-
guish different potential senses for some topic δs, e.g., for "*atom*", the
particle–sense and the format–sense. On the other hand, some senses
were split into two separate senses, e.g., "*Akbar the Great*", where "*em-
peror*" and "*empire*" indicated different senses. This misinterpretation
is due to the independent co-occurrence of "*emperor*" and "*empire*" with
δ, and the fact that it is unlikely that they share common words. How-
ever, we decided to make use of the *CE* in our system and adapted it in
the way that the results are shown as a touchable list that lead to the
webpage from which the definitions have been extracted (c.f. section 6).

6

A Final Walkthrough through the System

In the previous chapters we presented the modules that are needed to perform exploratory searches on mobile devices. For each component we summarised the current state of the art in research; we explained our approaches and the theory behind them in details; we did extensive evaluations of our modules and compared them to related works. What is still missing is the presentation of the system that has been implemented in the course of this thesis. For this we will start with a running example that shows the whole functionality of the system as the user may experience it. After that we will present user experiments and the evaluations of the system running on a tablet - Apple iPad - and on a smartphone - Apple iPhone.

6.1 Running Example

In this section we go through all components, screens and settings of the system and show how the system performs . We will show screenshots of the tablet version and in case of significant differences also of the smartphone version.

Figure 6.1: The user enters the query using the soft keyboard. In our case the query is "Jim Clark" which is passed to the disambiguation unit in Figure 6.2.

Figure 6.2: The disambiguation unit presents the first sentences of the Wikipedia abstract for each possible instance. In most cases it contains a definition of the term in question. The user now can choose one of the instances by simply touching it. The first entry can be chosen if no desambiguation is desired.
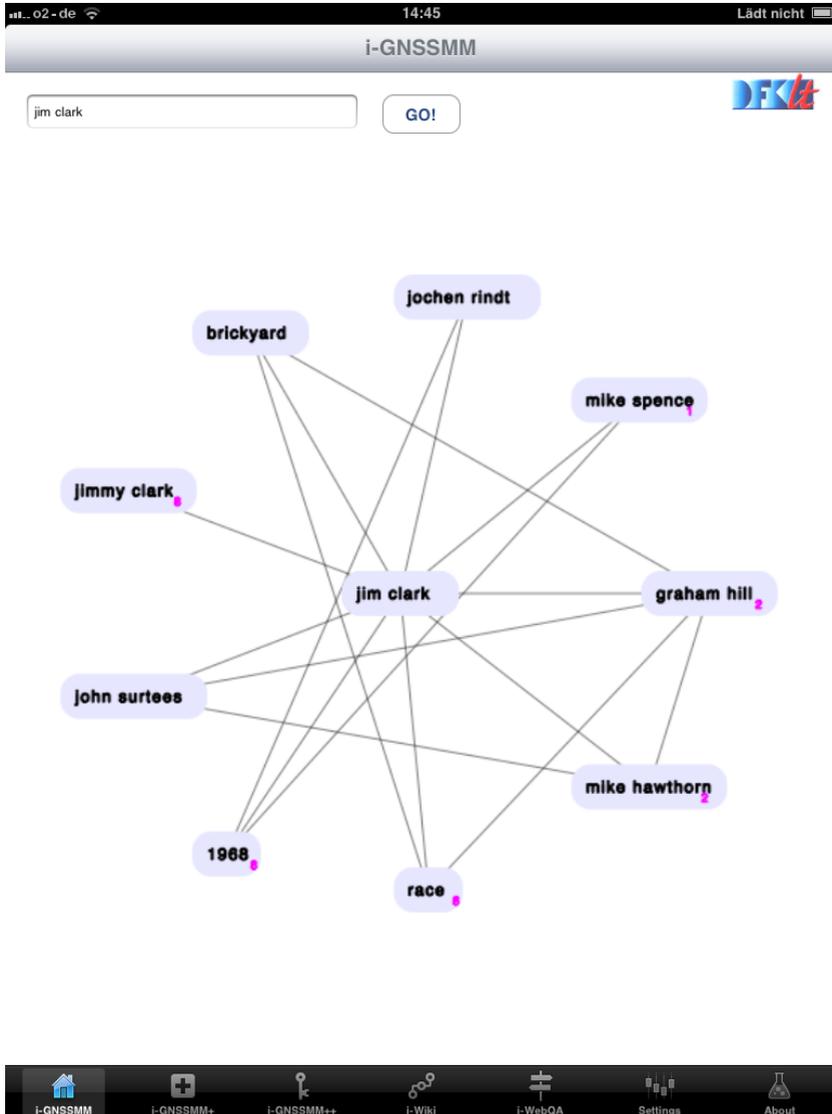
Figure 6.3: With the additional information provided by the user the system creates the topic graph for this instance - in our case for the formula one racing driver. The topic graph in this screenshot shows the extracted NEs based on the pure collocation chain approach (CPD_M).
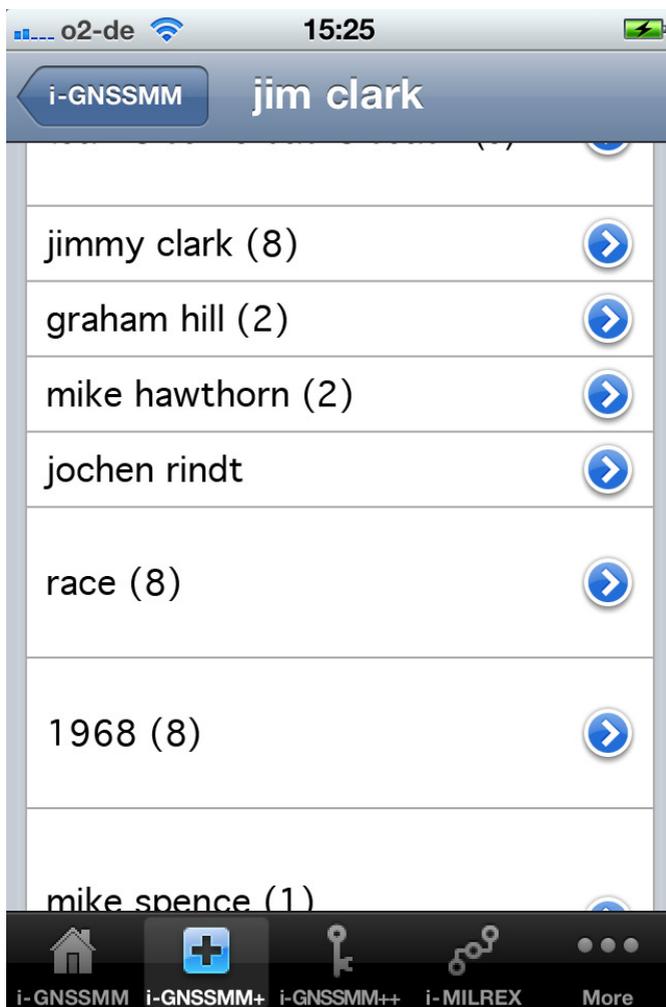
Figure 6.4: The visualisation of the topic graph on an iPhone is shown in this screenshot. The numbers behind the topics show how many topics are associated with this node. The blue arrows lead to the snippets from which the *NE*s have been generated from (Figure 6.9) .
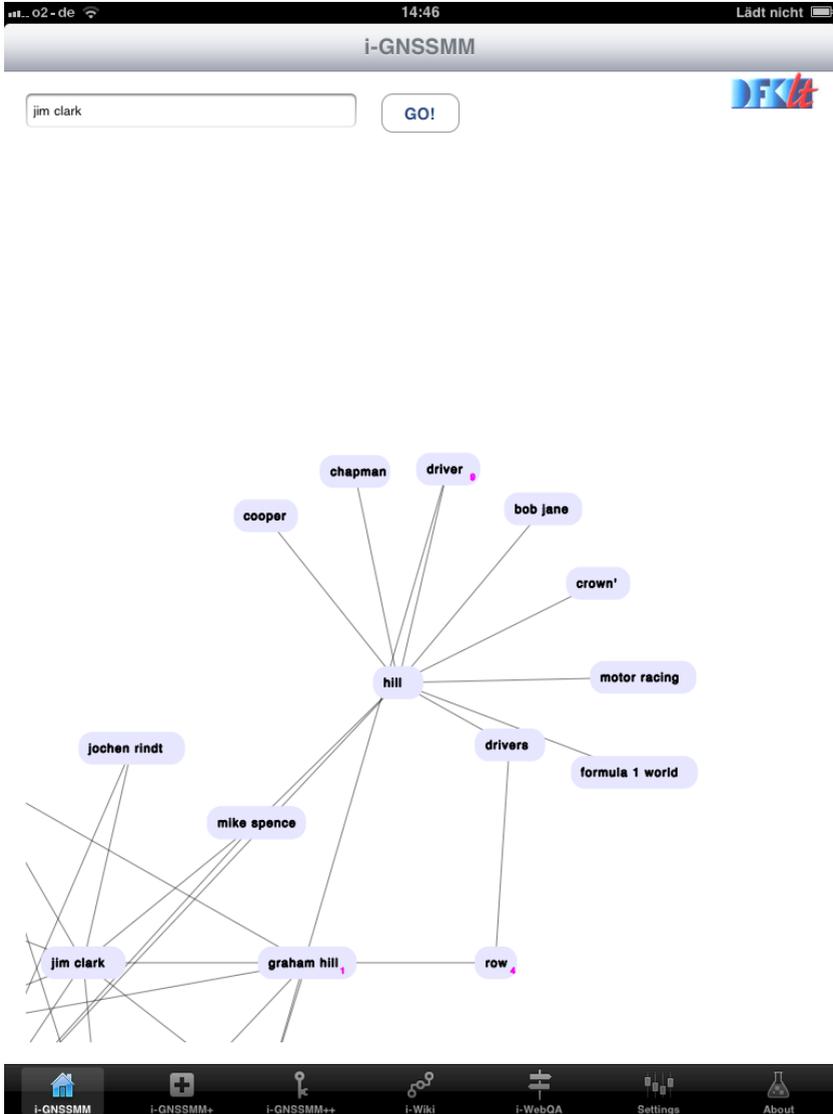
Figure 6.5: Touching a node opens a new branch containing *NE*s that are associated with the label of this node. The small numbers inside the nodes show how many topics are associated with this node. If there is no number, touching the node will start a new topic extraction process.
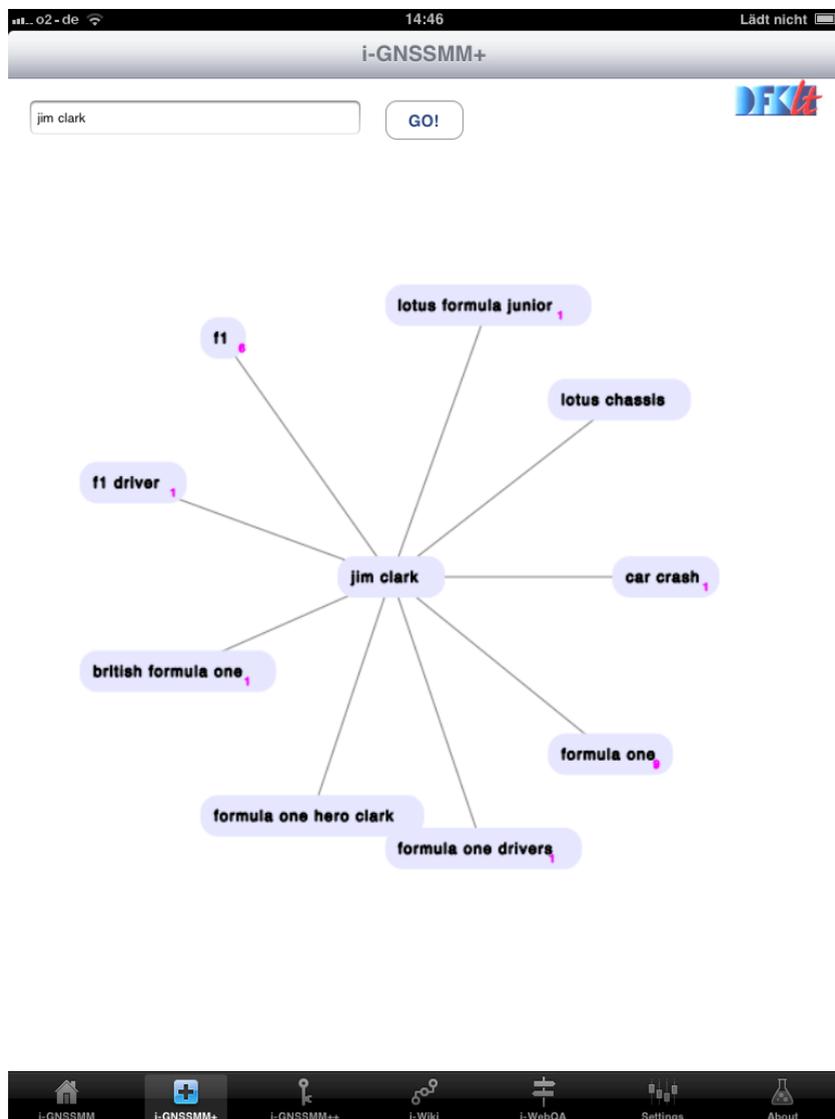
Figure 6.6: In comparison: this is the topic graph generated using Collocation Chains plus Singular Value Decomposition (SVD).
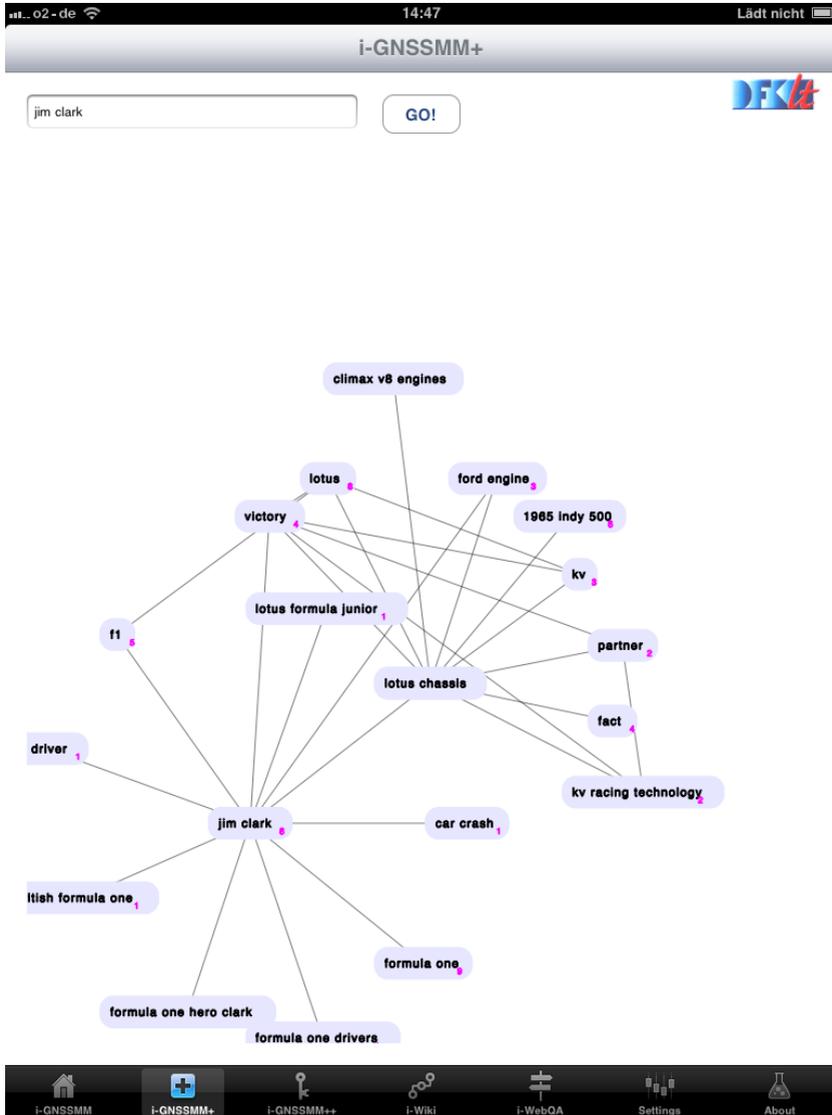
Figure 6.7: Again a new branch will be opened by touching the node. The NEs are also generated by CPD_M and SVD. On the iPhone the view is similar to the one in Figure 6.4.
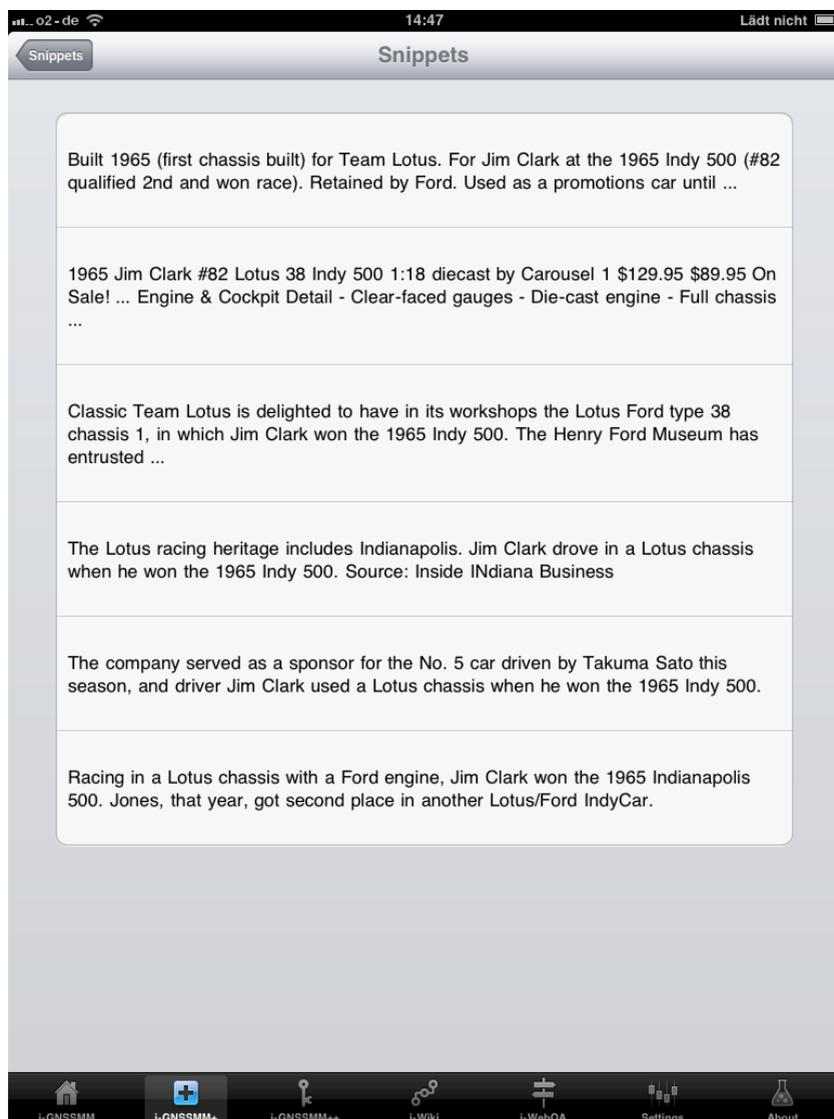
Figure 6.8: Double touching a node opens a new view. This view shows the underlying snippets from which the NEs have been generated from.
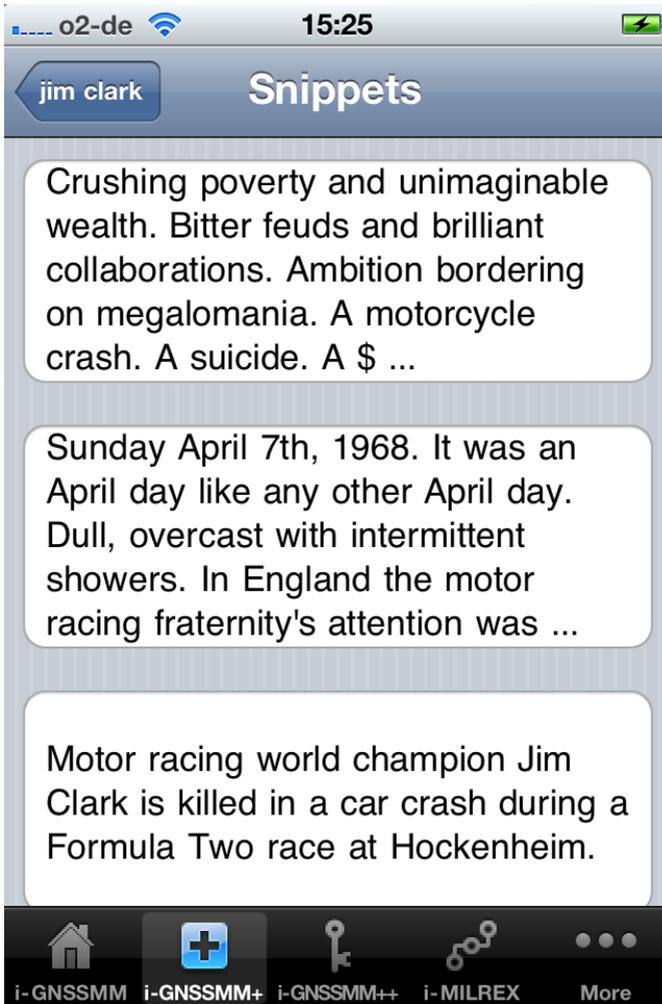
Figure 6.9: In case of the Smartphone the Snippets are shown according to the action described in Figure 6.4, i.e. touching the small blue arrow.

Figure 6.10: Touching a snippet opens the corresponding website in a new browser view.
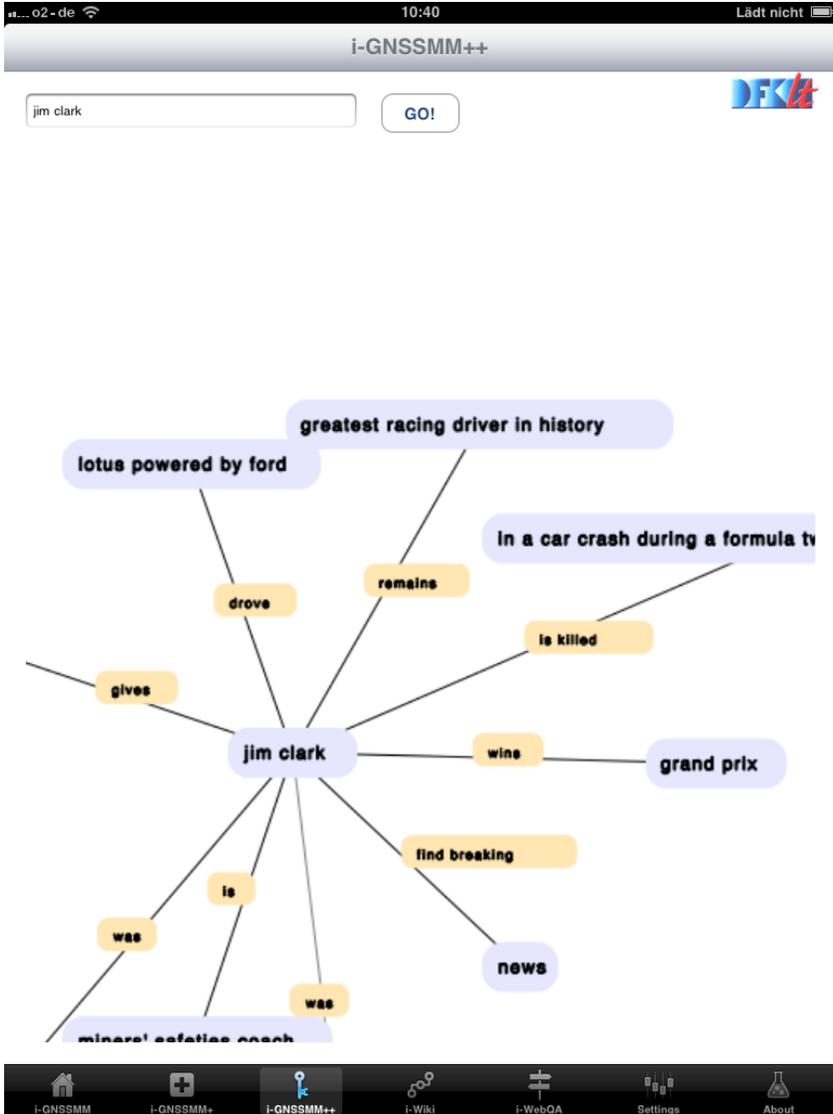
Figure 6.11: This topic graph has been constructed by the unsupervised relation extraction approach based on *CTDs* and *SVD*. This time the presentation in the iPhone is the same (only iPhone with Retina display is supported).
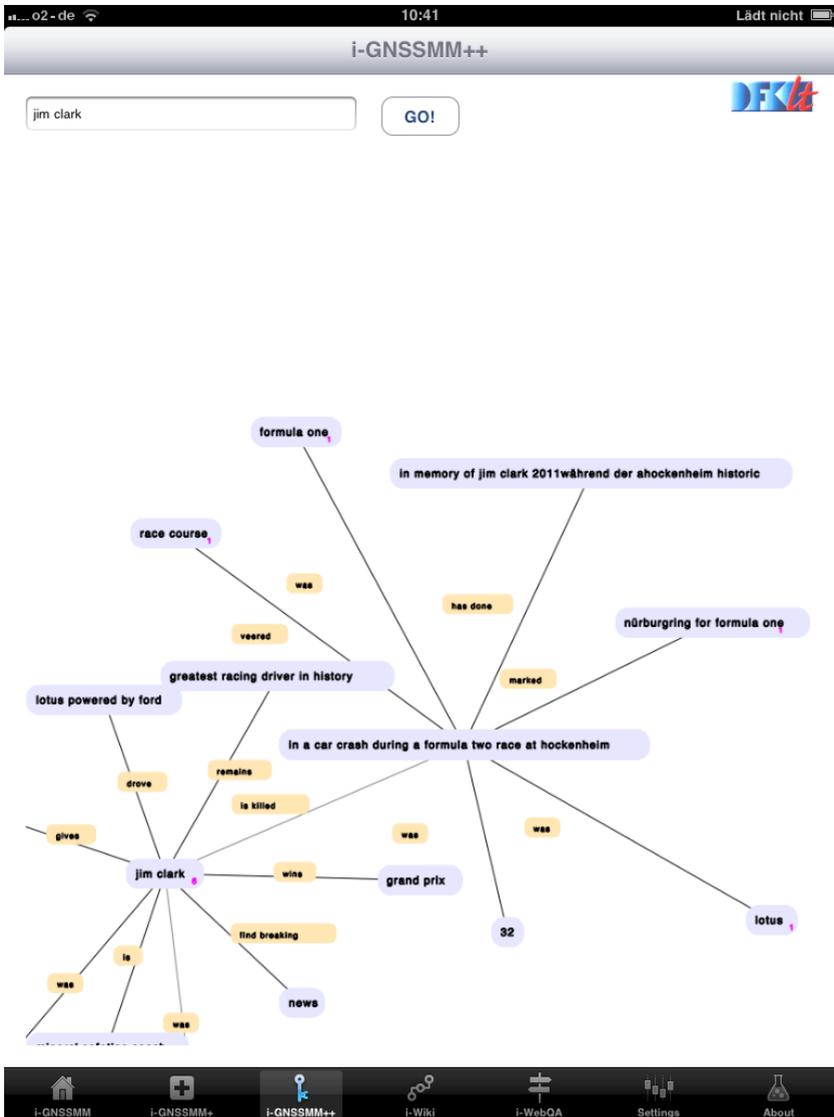
Figure 6.12: Touching a node shows the relations to other topics. This is the same as in Figure 6.5 or Figure 6.7.
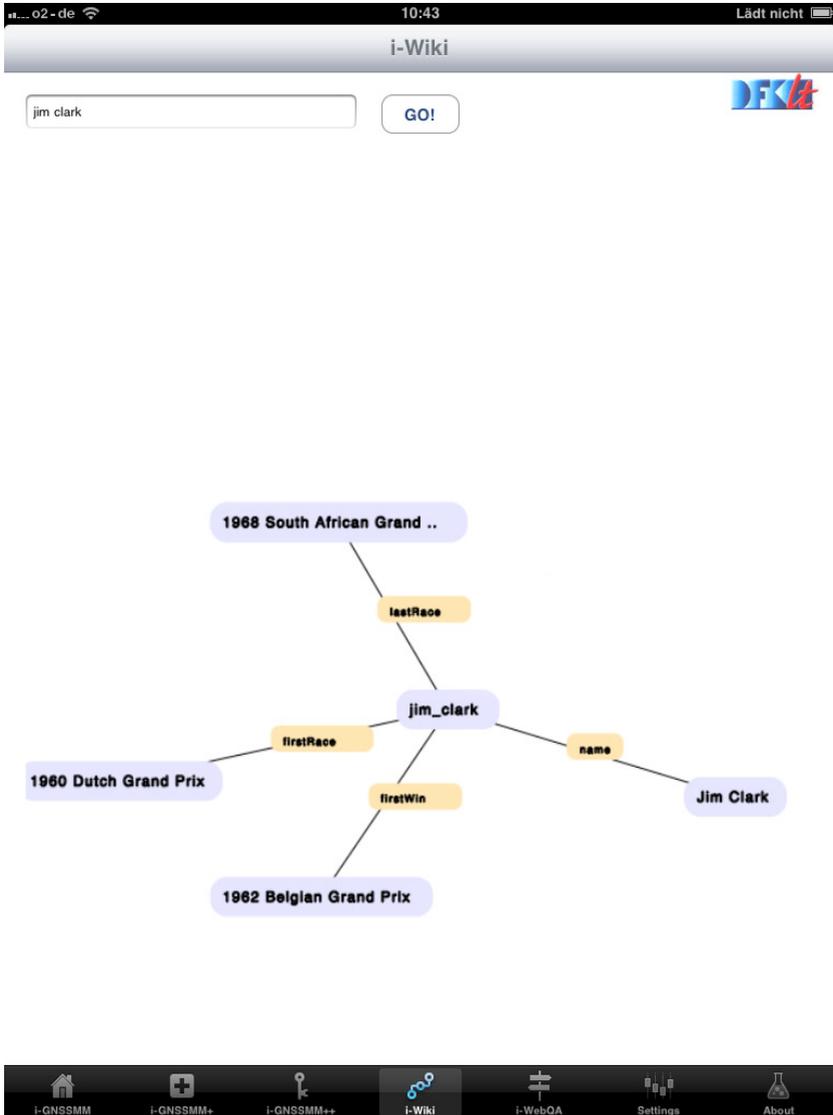
Figure 6.13: This view shows the resulting topic graph using information from Wikipedia Infoboxes. This time the topic graph has been generated by using Wikipedia infoboxes only.
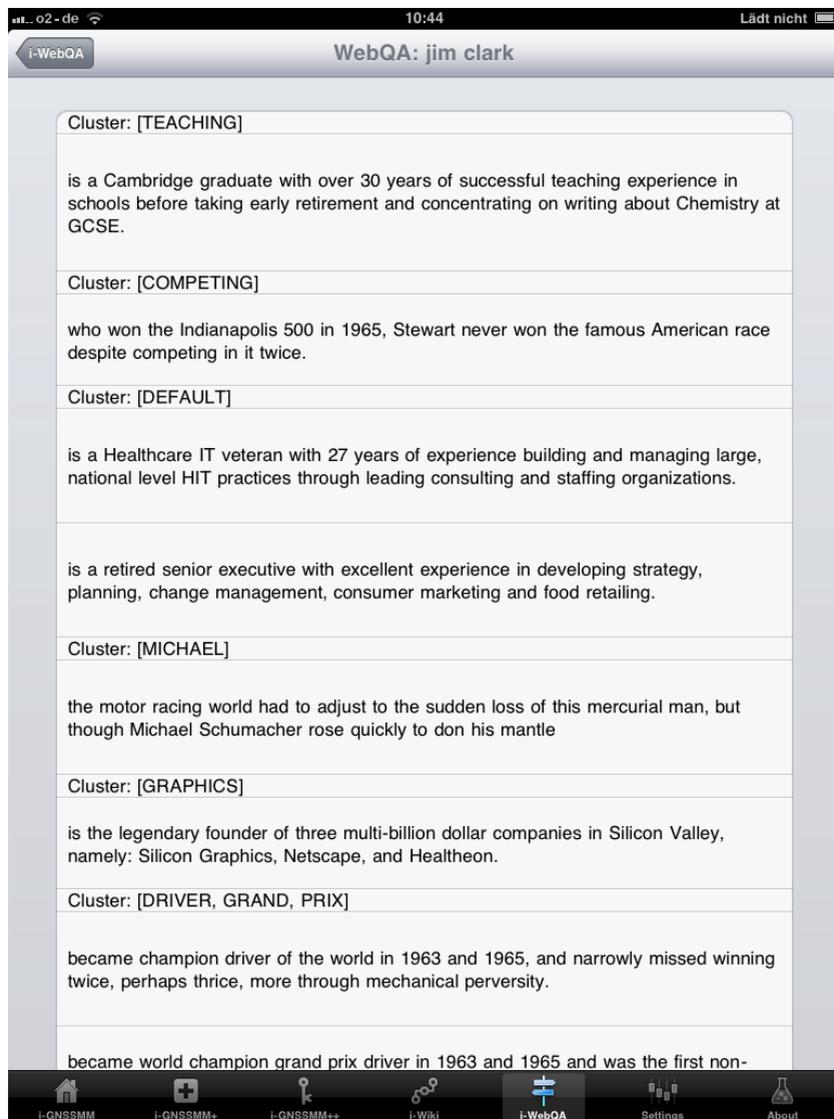
Figure 6.14: This view shows the results generated by the Concept Extraction (CE) unit. The query has been reformulated to a definition question: "define Jim Clark".

Figure 6.15: Clicking on a cell of the CE opens a new Browser view.
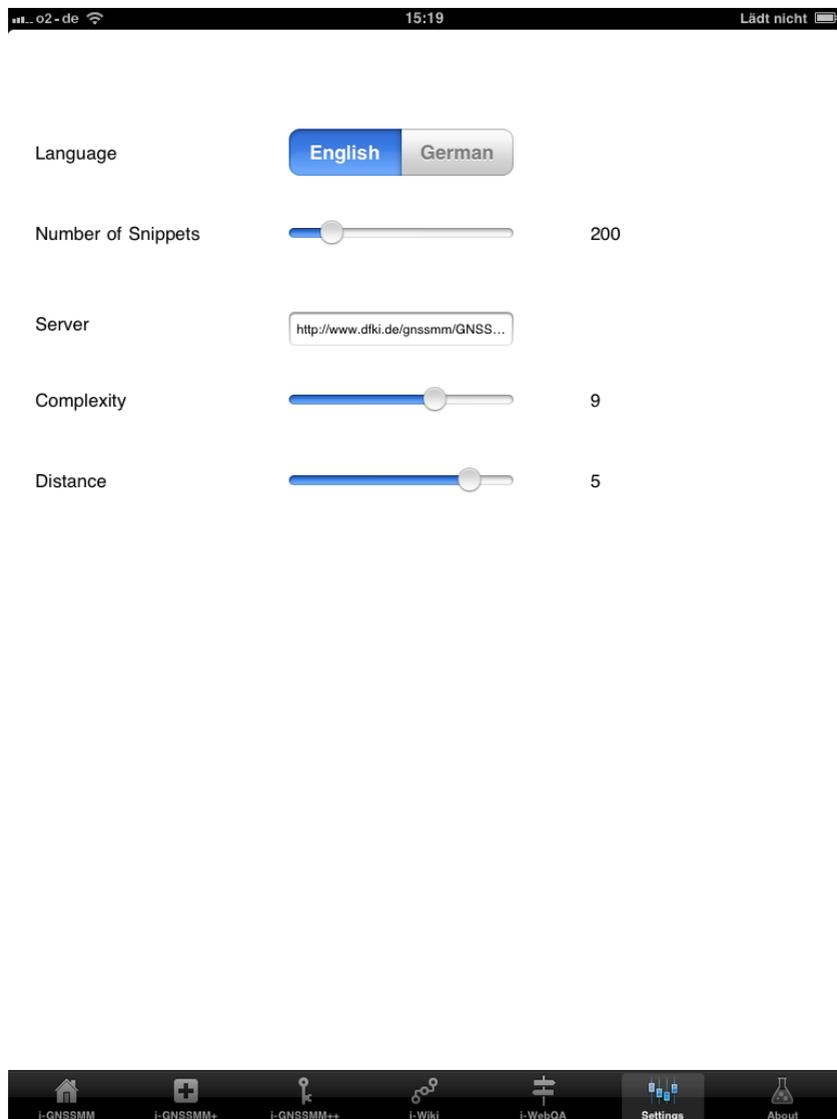
Figure 6.16: The Settings view allows the user to change the language, the number of snippets, the max. number of nodes associated with a NE in the topic graph, the max. distance between the query and associated topics in the snippets and of course the address of the server.
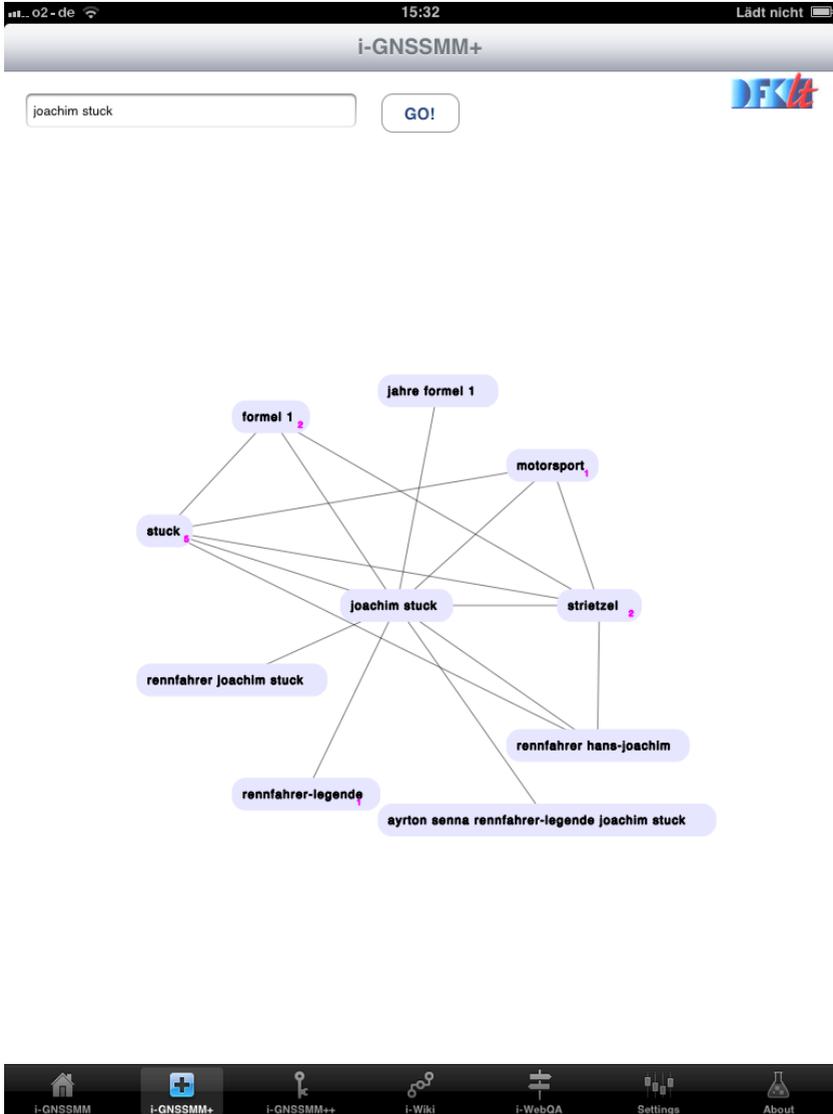
Figure 6.17: As described in the previous chapters all modules in the system are theoretically language-independent, tested for German and English. Except the PoS Tagger, which needs a trained model for the used language. Here is an example for a German topic graph generated by the query "Joachim Stuck".
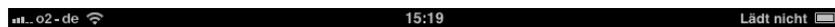
Figure 6.18: Finally of course the About page...

6.2   User Evaluation of the MobEx System

For information about the user experience we had 26 testers — 20 for testing the iPad app and 6 for testing the iPhone app: 8 came from our lab and 18 from non–computer science related fields. 15 persons had never used an iPad before, 4 persons have been unfamiliar with smartphones. More than 80 searches have been made with MobEx and with Google respectively.

After a brief introduction to our system (and the mobile devices), the testers were asked to perform three different searches (using MobEx on the iPad/iPhone and Google on the iPad/iPhone) by choosing the queries from a set of ten themes. The queries covered definition questions like *EEUU* and *NLF*, questions about persons like *Justin Bieber*, *David Beckham*, *Pete Best*, *Clark Kent*, *Wendy Carlos* , and general themes like *Brisbane*, *Balancity*, and *Adidas*. The task was not only to get answers on questions like "Who is ..." or "What is ...", but also to acquire knowledge about background facts, news, rumours (gossip) and more interesting facts that come into mind during the search.

Half of the iPad–testers were asked to first use Google and then MobEx in order to compare the results and the usage on the mobile device. We hoped to get feedback concerning the usability of our approach compared to the well known internet search paradigm. The second half of the iPad–testers used only our system. Here our research focus was to get information on user satisfaction of the search results. The iPhone–testers always used Google and MobEx, mainly because they were fewer people.

After each task, both testers had to rate several statements on a Likert scale and a general questionnaire had to be filled out after completing the entire test. The tables 6.1, 6.2, 6.3, and 6.4 show the overall result.

#Question	v.good	good	avg.	poor
results first sight	43%	38%	20%	-
query answered	65%	20%	15%	-
interesting facts	62%	24%	10%	4%
suprising facts	66%	15%	13%	6%
overall feeling	54%	28%	14%	4%

Table 6.1: MobEx on the iPad

#Question	v.good	good	avg.	poor
results first sight	55%	40%	15%	-
query answered	71%	29%	-	-
interesting facts	33%	33%	33%	-
suprising facts	33%	-	-	66%
overall feeling	33%	50%	17%	4%

Table 6.2: Google on the iPad

#Question	v.good	good	avg.	poor
results first sight	31%	46%	23%	-
query answered	70%	20%	10%	-
interesting facts	45%	36%	19%	-
suprising facts	56%	22%	11%	11%
overall feeling	25%	67%	8%	-

Table 6.3: MobEx on the iPhone

#Question	v.good	good	avg.	poor
results first sight	23%	63%	7%	7%
query answered	70%	20%	10%	-
interesting facts	33%	33%	33%	-
suprising facts	36%	-	27%	37%
overall feeling	25%	33%	33%	9%

Table 6.4: Google on the iPhone

The results show that people prefer the result representation and accuracy in the Google style when using the iPad. Especially for the general themes, the presentation of web snippets is more convenient and easier to understand. The iPhone–testers could be divided into two groups: in case they were unfamiliar with smartphones the testers preferred our system, because it needs much less user interaction and the results are more readable. Testers being familiar with smartphones again preferred the Google style, mainly because they are used to it.

However, when it comes to interesting and suprising facts users enjoyed exploring the results using the topic graph (iPad) or the navigation-based representation (iPhone/iPod). The overall feeling was in favour of our system, which might also be due to the fact that it is new and somewhat more playful.

The replies to the final questions: *How successful were you from your point of view? What did you like most/least;? What could be improved?* were informative and contained positive feedback. Users felt they had been successful using the system. They liked the paradigm of the exploratory search on the iPad and preferred touching the graph instead of reformulating their queries. For the iPhone they prefered the result representation in our system in general and there have been useful comments on how to improve it. One main issue is the need of a summary or a more knowledge based answer to the search query as Google often does it by offering a direct link to wikipedia as a first search result. This will be part of our future research.

Although all of our test persons make use of standard search engines, most of them can imagine to use our system, at least in combination with a search engine on their own mobile devices. The iPhone test group even would use our system as their main search tool (on the smartphone) if the proposed improvements have been implemented.

7

Conclusion and Outlook

The research in this thesis has focused on *exploratory search* for mobile devices. The central part was the design, implementation, and evaluation of several core modules for on-demand unsupervised information extraction (*IE*) well suited for usage on mobile devices. These core processing elements, combined with a multitouchable user interface specially designed for two families of mobile devices, i.e. smartphones and tablets, have been finally implemented in the research prototype *MobEx*. The evaluation results for each component and also the feedback given by testers of *MobEx* have been very positive and encouraging.

In this final chapter we will summarise the core features of the *MobEx* system, discuss open issues, and propose some future directions

7.1 Summary

Like in ordinary search engines a topic is issued online and the whole process is started by the user.

7.1.0.1 Query Disamiguation – QD

In a first step the user's query is disambiguated by using a knowledge base, i.e. currently Wikipedia articles, as the main source for disambiguation.

Beside the fact that Wikipedia is known to cover a huge number of possible senses for a very large number of topics, we also consider Wikipedia as a suitable means of a human–computer interface in the

sense that both the human and the computer, can directly communicate in natural language (NL). If there are more than one matching Wikipedia article, the user gets presented the abstracts of those articles - partly truncated if they exceed the size of 80 words. The user is expected to choose one of these abstracts by touching it on the mobile device or skipping this step if she/he wants to explore the solution space without any restriction. In case he/she chooses a certain meaning the system expands the original query with a set of detected nouns in the Wikipedia article that are able to keep the next process focussed on the chosen meaning. In large scale automatic tests we could prove that the accuracy of the system is about 90%. Smaller scaled manual tests confirmed this.

7.1.0.2   Named Entity Identification – NEI

The main task of the *Named Entity Identification* component in MobEx is to determine an initial set of correlated entities from the (expanded) input topic. Such a set of correlated entities corresponds to an association graph, which is the basis for the topic graph. So seen *NEI* is equivalent to a Topic Extraction Process (*TEP*) in our system. We begin by creating a document S from the N-first web snippets so that each line of S contains a complete snippet. In our research we use the BING search engine by Microsoft to retrieve the snippets. Each textline of S is then tagged with Part–of–Speech using the SVMTagger (36) and chunked in the next step. We then compute the chunk-pair-distance model (CPD_M), which contains all available collocation information between the noun chunks. In a final step we determine the most important associations between the chunks by using a special Pairwise Mutual Information measure (*PMI*). The visualised topic graph TG is then computed from a subset $CPD'_M \subset CPD_M$ using the m highest ranked *cpd* for fixed c_i. In other words, we restrict the complexity of a TG by restricting the number of edges connected to a node. Note that the whole process works in a completely unsupervised way and very efficiently. The average processing time is about three seconds for N=200. Furthermore it is completely domain and theoretically language independent, tested for German and English. The models of the POS Tagger need to be trained for each language seperately. For the evaluation of *TEP* we compared it to the results of four different NE recognisers: SProUT(6), AlchemyAPI[1], Stanford NER(22), and OpenNLP[2]. We could prove that

---

[1]http://www.AlchemyAPI.com
[2]http://incubator.apache.org/opennlp/

the achieved results are comparable and sometimes outperform the other approaches. However, when going into details we noticed that especially topics which are highly context-dependent and can be extracted by our system, but not by the others. Please note that the *TEP* approach works for query-driven context-oriented named entity recognition only. This means that all approaches used in this evaluation clearly have their benefits in other application areas.

Changing the view from the core to the system as an App on an iPad or iPhone the topic graph is then displayed either as touch–sensitive graph - on a tablet computer, in our case an iPad - or it is displayed as a stack of touchable text on a smartphone - in our case an iPhone or an iPod touch. By just selecting a node or a text box, the user can either inspect the content of a topic (i.e, the snippets or web pages) or activate the expansion of the topic graph through an on-the-fly computation of new related topics for the selected node. Note that each new query sent to the search engine is created from the label of the selected node and the "sense"-information as created above from Wikipedia. Thus, each search triggered by a selected topic node is guided towards the user's preferred reading.

7.1.0.3   Relation Extraction – RE

The *Relation Extraction* component in MobEx also works in an unsupervised and highly efficient way. We extended the *CPD* model to a *CTD* - a chunk triple distance - model. In contrast to observing the statistics of *NP* chunks only we now also look at (parts of) the verb group *VG* lying in between the NP chunks. In general for two NP chunks, a single chunk-triple-distance element stores the distance between the first NP chunk and the VG as well as the distance between the VG and the second NP chunk. Please note that this process again meets our main requirements: fastness, on-demand, up-to-date and indicative. The construction of the *CTD* model is a bit more complicated than the construction of the *CPD* model in order not to run into a sparse data problem. In fact it is necessary to use a fuzzy strategy to perform the match between VGs in a *CTD*. So we developed a special fuzzy matching algorithm that does not require exact matches between the elements of a triple. It is motivated by the Levenshtein algorithm, which penalises replacements, insertions, and deletions of characters in order to compute possible matches between arbitrary strings. Instead of penalizing characters, we penalise replacements, insertions, and deletions of PoS-types in VGs and NGs.

We defined types of non exact matches like missing determiner in a NP chunk or different adverb in the VG, etc. and valued them using natural numbers (see 4.2.3 for the details). Whenever one of those types occurs, we penalise the possible match by the corresponding value. If a certain penalty threshold is exceeded, the triples do not match. The computation of the extended CTD_M is straightforward: we temporarily reduce the triples back to tuples and use the same PMI as we did to compute the CPD_M. For the evaluation we concentrated on randomly picking results from several test runs and checking the correctness of the extracted relations. This approach is often used to evaluate unsupervised methods, e.g. in (27), (23), (80), etc. To gather enough examples we ran the system in a batch mode using lists of named entities as the source for our search queries. We defined five different categories for the matches: (a) correct relation, (b) correct relation but underspecified NP chunk, (c) correct relation but incomplete because of not being contained in the snippets, (d) correct, but no sense i.e. an explaining part has not been extracted, and finally (e) incorrect. The numbers for the classes are quite impressive. We found 173/300 for (a), 18/300 for (b), 23/300 for (c), 63/300 for (d) , and 18/300 for (e). During the TEP research we have learned that the ratio between complete and incomplete sentences varies between 50% and 70%. This means the real recall is roughly around 40% to 60%. However, we believe that this still is a very good result if you keep in mind that not all sentences in the snippet, complete or incomplete, describe a new relation and there are also enough relations that are expressed by complete and incomplete sentences in the same corpus.

7.1.0.4   Addtitional Modules

During our research we found some drawbacks we tried to catch with additional modules, i.e. a QD component based on statistical methods and singular value decomposition and a RE component purely based on explicit facts of Wikipedia. We call them additional as they are integrated in the MobEx system, but they are not thought of being used regularly. First because the computing time is beyond the limit of five seconds (10) in case of the QD task, and second it is very incomplete in case of the RE part. Nevertheless both approaches have their advantages.

As the statistic *QD* does not rely on a hand crafted knowledge base but instead works on the entire internet it may have a higher coverage and hence provides senses to topics that are not contained in Wikipedia. The *RE* based on Wikipedia may deliver background facts that are not to be found in the search results of an ordinary search engine.

7.2   Next Steps and Future Research

This thesis has reported on a complete system and a complete evaluation. Nevertheless, during our research we came up with a lot of new ideas, which could not be covered here. We divided them into three different areas: (1) Limit the system to other domains, i.e. apply it to large databases instead of the whole internet for an alternative search in intranets; (2) add knowledge sources like *DBPedia*, *Wordnet*, *Open Linked Data*, *specialized Onthologies*, *Thesauri*, etc; (3) add more core linguistic modules like *Dependency Parsers*, *Coreference Resolution* and modules that do not need that much redundant information like the current components in *MobEx*.

7.2.1   Domain Dependency

During the research of this thesis the *MobEx* system has been presented in different scientific and public conferences: *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (68), *4th International Conference on Agents and Artificial Intelligence (ICAART 2012)* (69), *Informare!*[3], and in further smaller discussion groups. One major feedback has been whether the system could be tailored in a way that hidden information in intranet or database structures could be made visible in the same manner. *MobEx* is open to interface with different data sources in principle by its defined interface design. Future projects based on this work will hopefully prove the usefulness of the approach even in tailored domains.

---

[3]http://informare-wissen-und-koennen.com/

7.2.2   More Knowledge Sources

For the moment Wikipedia is the only knowledge–source used in the system. Equipping the system with more knowledge could improve the *TEP* and *RE* processes. It could be either domain-specific knowledge, for example special databases or any other structured source. Also the sources mentioned in 7.2.1 could be added to the system. Attaching and using more knowledge has already been taken care of in the system interfaces. However, the implementation of the interfaces still has to be done to bring the knowledge into the right format, which of course depends on the format of the source. Also the user interface would probably be affected. Nevertheless, this would be an interesting path to follow in order to improve the experience of exploratory search.

7.2.3   More Linguistic Modules

Although the results of the current modules are comparable to supervised or semi-supervised approaches, *MobEx* is dependent on redundancy of the data. To get rid of this strong dependency we need additional linguistic modules. An easy way of integration is to replace the current process of building the chunk pair distance model CPD_M for the *TEP* or the chunk triple distance model CTD_M for the *RE* part based on our *PMI* and *fuzzy match*.

7.2.4   Current and Future Use

In principle *MobEx* could be used by everyone right now. It is robust, fast, efficient, delivers excellent results and the user interface is accepted by people. However, one major feedback is that Microsoft finally changed its policy towards the usage of its search API. It has been free until August 2012. After that Micosoft has introduced a pay per use model, which seems to be practical for big companies only. Google stopped its service at all several years ago and there seems to be no other search service provider with an open and configurable API. Therefore we attached the only free search service for research, which is called BLEKKO (www.blekko.com). They provide an easy-to-use interface, but unfortunately the search results still are tailored to the USA, so using BLEKKO for other languages than English is not possible yet.

The core components could also be used by other applications. Especially the query disambiguation part is useful in any interactive search or information delivery systems. Whenever the data provides enough redundancy, the other components could be used too very efficiently. Also the knowledge sources that have been built in this thesis, like the index of Wikipedia Infoboxes can be of great use for example to support any kind of RE processes in other applications.

Appendix A

Evaluation of PCL(SIL)

This appendix describes our work on the *RE* task in detail in context of *MobEx* (see 4.2.1.2)

In order to check the quality of the PCL(SIL) method we compared it to state of the art machine learning algorithms. Despite the fastness and robustness, the quality of the classification needs to be checked. For this several standard data sets have been collected and applied to PCL(SIL):

- 20 Newsgroups: This dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. It has been spilt into a test and a training set (Table A.1)

- Reuters 21578: This is currently the most widely used test collection for text categorization research. The data was originally collected and labeled by Carnegie Group, Inc. and Reuters, Ltd. in the course of developing the CONSTRUE text categorization system (Table A.2 and A.3)

- Cade12: The documents in the Cade12 correspond to a subset of web pages extracted from the CADÊ Web Directory, which points to Brazilian web pages classified by human experts. This directory is available at Cade's Homepage[1], in Brazilian Portuguese.

---

[1]http://www.cade.com.br/

- WebKB: The documents in the WebKB are webpages collected by the World Wide Knowledge Base (Web-Kb) project of the CMU text learning group. These pages were collected from computer science departments of various universities in 1997, manually classified into seven different classes: student, faculty, staff, department, course, project, and other. For each class, the collection contains pages from four universities: Cornell, Texas, Washington, Wisconsin, and other miscellaneous pages collected from other universities. We discarded the classes staff, department and other as they were too university specific or contained not enough documents (Table A.5)

Class	# train docs	# test docs	Total # docs
alt.atheism	480	319	799
comp.graphics	584	389	973
comp.os.ms-windows.misc	572	394	966
comp.sys.ibm.pc.hardware	590	392	982
comp.sys.mac.hardware	578	385	963
comp.windows.x	593	392	985
misc.forsale	585	390	975
rec.autos	594	395	989
rec.motorcycles	598	398	996
rec.sport.baseball	597	397	994
rec.sport.hockey	600	399	999
sci.crypt	595	396	991
sci.electronics	591	393	984
sci.med	594	396	990
sci.space	593	394	987
soc.religion.christian	598	398	996
talk.politics.guns	545	364	909
talk.politics.mideast	564	376	940
talk.politics.misc	465	310	775
talk.religion.misc	377	251	628
Total	11293	7528	18821

Table A.1: 20 Newsgroups

Class	# train docs	# test docs	Total # docs
acq	1596	696	2292
alum	31	19	50
bop	22	9	31
carcass	6	5	11
cocoa	46	15	61
coffee	90	22	112
copper	31	13	44
cotton	15	9	24
cpi	54	17	71
cpu	3	1	4
crude	253	121	374
dlr	3	3	6
earn	2840	1083	3923
fuel	4	7	11
gas	10	8	18
gnp	58	15	73
gold	70	20	90
grain	41	10	51
heat	6	4	10
housing	15	2	17
income	7	4	11
instal debt	5	1	6
interest	190	81	271
ipi	33	11	44
iron steel	26	12	38
jet	2	1	3
jobs	37	12	49
lead	4	4	8
lei	11	3	14
livestock	13	5	18
lumber	7	4	11
meal feed	6	1	7
money fx	206	87	293
money supply	123	28	151
nat gas	24	12	36
nickel	3	1	4
orange	13	9	22
pet chem	13	6	19

Table A.2: Reuters21578-part1

Class	# train docs	# test docs	Total # docs
platinum	1	2	3
potato	2	3	5
reserves	37	12	49
retail	19	1	20
rubber	31	9	40
ship	108	36	144
strategic metal	9	6	15
sugar	97	25	122
tea	2	3	5
tin	17	10	27
trade	251	75	326
veg oil	19	11	30
wpi	14	9	23
zinc	8	5	13
Total	6532	2568	9100

Table A.3: Reuters21578-part2

Class	# train docs	# test docs	Total # docs
01–servicos	5627	2846	8473
02–sociedade	4935	2428	7363
03–lazer	3698	1892	5590
04–informatica	2983	1536	4519
05–saude	2118	1053	3171
06–educacao	1912	944	2856
07–internet	1585	796	2381
08–cultura	1494	643	2137
09–esportes	1277	630	1907
10–noticias	701	381	1082
11–ciencias	569	310	879
12–compras-online	423	202	625
Total	27322	13661	40983

Table A.4: Cade12

Class	# train docs	# test docs	Total # docs
project	336	168	504
course	620	310	930
faculty	750	374	1124
student	1097	544	164
Total	2803	1396	4199

Table A.5: WebKB

In (14) several state of the art SIL algortihms have been applied to different standard data sets. In table A.6 the properties of the four datasets are shown.

Dataset	Classes	Train Docs	Test Docs	Total Docs	Language
20 Newsgroups	20	11293	7528	18821	English
Reuters-21578	52	6532	2568	9100	English
Web KB	4	2803	1396	4199	English
Cade12	12	27322	13661	40983	Portuguese

Table A.6: Datasets

In order to evaluate the PCL(SIL) method it has been applied to the same sets and compared the results. Table A.7 shows the accuracy values obtained by each method. The "Dumb" classifier always predicts the most frequent class in the training set. It gives a baseline to classification and reflects the unbalance of the data.

Dataset	Dumb	kNN	SVM	PCL(SIL)
20-Newsgroups	0.0530	0.7593	0.8284	**0.8453**
Reuters-21578	0.4217	0.8322	**0.9377**	0.9202
Web KB	0.3897	0.7256	0.8582	**0.8591**
Cade	0.2083	0.5120	0.5284	**0.5344**

Table A.7: PCL(SIL) Evaluation and Comparison

The results show that our PCL(SIL) approach is comparable to the best classifiers[2], i.e. SVM for all datasets. In fact in three of four cases our approach outperforms the SVM. Only for the Reuters dataset we are slightly worse than the SVM.

---

[2]We used the RapidMiner open-source package (63)

Appendix B

Evaluation of PCL(MIL)

The PCL(MIL) algorithm has been applied to the example sets and compared with the results of (12).

In figure B.1 the two graphs, i.e. the ROC curve labeled with SSK-MIL and the ROC curve of PCL(MIL) are shown in one diagram. They are based on exactly the same data without using any biasing.

Especially in recall ranges between 0.0 and 0.8 our algorithm shows better results than the SVM approach using the subsequence kernel, in recall ranges between 0.4 and 0.8 the results are better or similar to the linguistically biased SVM approach.

For further impressions on how the PCL(MIL) algorithm performs in general we trained it on several sets of document snippets retrieved using the BING search engine. Each of these sets resembles certain binary relations.

The figures show that the perfomance heavily depends on the relation itself. For example the relation *marriage(person,person)* shows very good results whereas *causeOfDeath(person, cause)* performs relatively poor. As already mentioned in section 4.2.1.2 this effect correlates with the number of different ways a relation is expressed in the retrieved data.

Positive Traingsset		#Snippets
Arnold Schwarzenegger	* * * * * * *　Austria	375
Albert Einstein	* * * * * * *　Germany	622
Dirk Nowitzki	* * * * * * *　Germany	323
Detlef Schrempf	* * * * * * *　Germany	311
Brigitte Bardot	* * * * * * *　France	163
Negative Trainingset		
Berlusconi	* * * * * * *　Germany	1375
Franz Beckenbauer	* * * * * * *　Usa	622
Helmut Kohl	* * * * * * *　England	323
Positive Testset		
John Lennon	* * * * * * *　England	1375
Sophia Loren	* * * * * * *　Italy	622
Paul Anka	* * * * * * *　Canada	323
Negative Testset		
Michael Schumacher	* * * * * * *　Italy	1375
Madonna	* * * * * * *　England	622
Charlie Chaplin	* * * * * * *　Usa	323

Table B.1: Queries for relation: *origin(person, country)*

Positive Traingset			#Snippets
John Lennon	* * * * * * *	New York	399
Albert Einstein	* * * * * * *	Princeton	545
James Dean	* * * * * * *	Cholame	444
Charlie Chaplin	* * * * * * *	Vevey	501
George Harrison	* * * * * * *	Los Angeles	577
Negative Traingset			#Snippets
Willy Brand	* * * * * * *	Berlin	62
Maraget Thatcher	* * * * * * *	London	180
Helmut Kohl	* * * * * * *	Bonn	709
Positive Testset			#Snippets
Stan Laurel	* * * * * * *	Santa Monica	318
Rock Hudson	* * * * * * *	Beverly Hills	651
Frank Sinatra	* * * * * * *	Los Angeles	548
Negative Testset			#Snippets
Michael Schumacher	* * * * * * *	Italy	629
Madonna	* * * * * * *	England	479
Charlie Chaplin	* * * * * * *	Usa	519

Table B.2: Queries for relation: *placeOfDeath(person, city)*
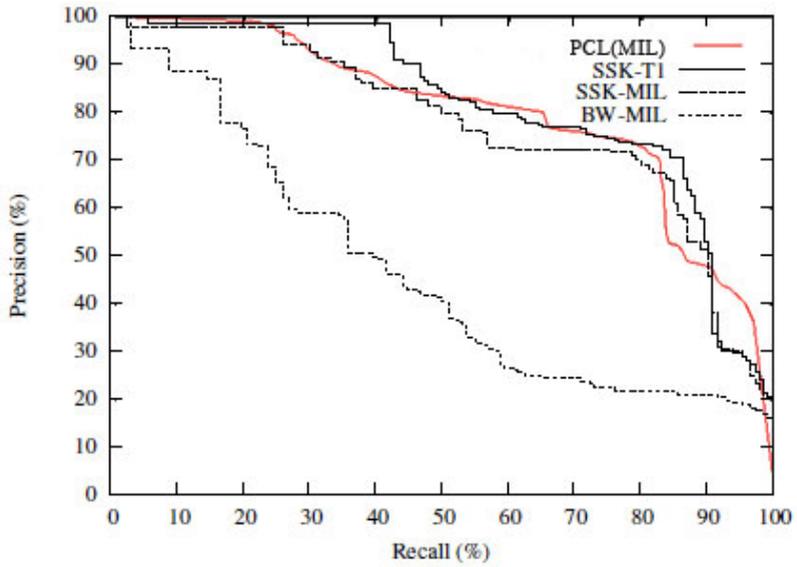
Figure B.1: Comparison of SVM unbiased and PCL (MIL)

ROC origin(person,country)
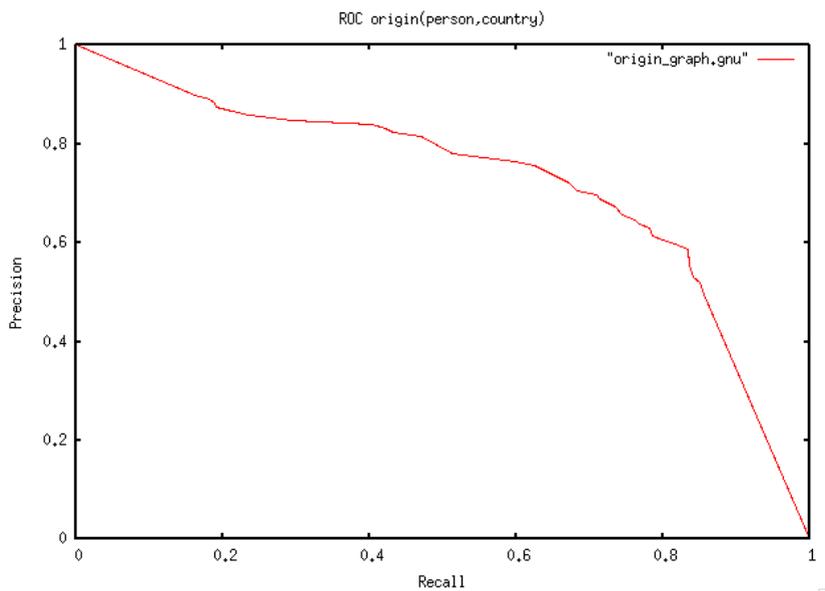
Figure B.2: Statistical ROC for origin(person,country)

Positive Traingset			#Snippets
Freddy Mercury	* * * * * * *	Aids	69
George Harrison	* * * * * * *	Cancer	537
James Dean	* * * * * * *	Accident	617
Kurt Cobaine	* * * * * * *	Suicide	89
Sharon Tate	* * * * * * *	Murder	517
Sid Vicious	* * * * * * *	Overdose	704
Negative Traingset			#Snippets
Willy Brand	* * * * * * *	Cancer	9
Maraget Thatcher	* * * * * * *	Cancer	60
Helmut Kohl	* * * * * * *	Cancer	500
Positive Testset			#Snippets
Elvis Presley	* * * * * * *	Heart Attack	656
Buddy Holly	* * * * * * *	Crash	556
Marilyn Monroe	* * * * * * *	Suicide	531
Negative Testset			#Snippets
Louis Armstrong	* * * * * * *	Murder	137
Graham Hill	* * * * * * *	Suicide	89

Table B.3: Queries for relation: *causeOfDeath(person, cause)*

Positive Traingset			#Snippets
Johannes Heesters	* * * * * * *	Simone Rethel	116
Nicolas Cage	* * * * * * *	Alice Kim	80
Nicole Kidman	* * * * * * *	Keith Urban	180
Tommy Lee Jones	* * * * * * *	Dawn Laurel	6
Negative Traingset			#Snippets
John Lennon	* * * * * * *	Paul McCartney	555
Stan Laurel	* * * * * * *	Oliver Hardy	470
Positive Testset			#Snippets
Clint Eastwood	* * * * * * *	Diana Ruiz	59
Eva Longoria	* * * * * * *	Tony Parker	147
Negative Testset			#Snippets
Jerry Lewis	* * * * * * *	Dean Martin	186
Kirk Douglas	* * * * * * *	Michael Douglas	166

Table B.4: Queries for relation: *marriage(person, person)*
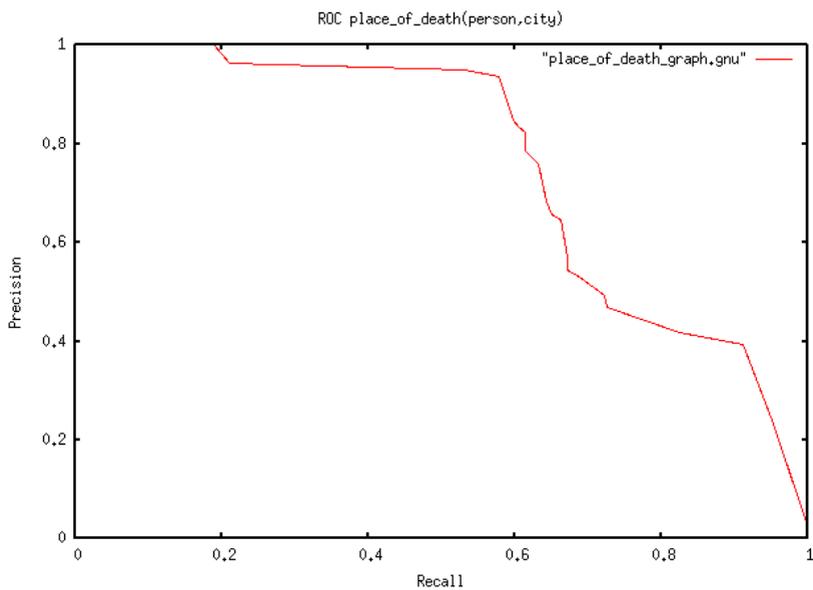
Figure B.3: Statistical ROC for placeOfDeath(person,city)
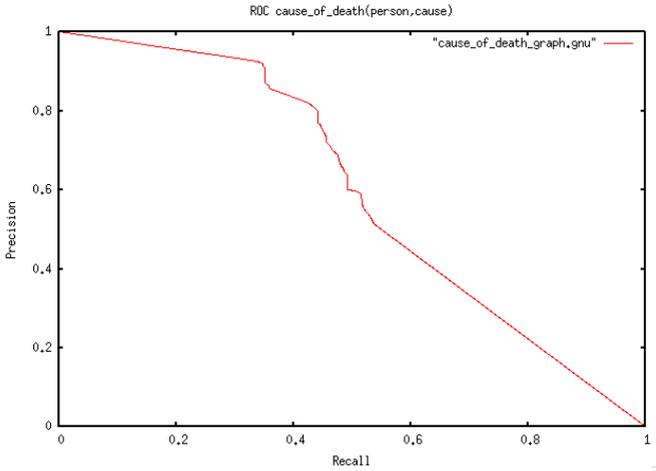
Figure B.4: Statistical ROC for causeOfDeath(person,cause)
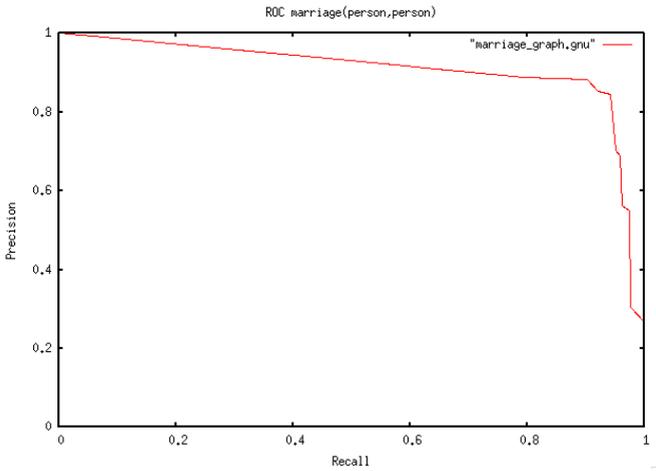
Figure B.5: Statistical ROC for marriage(person,person)

Appendix C

PCL(MIL) and linguistic processing

The combination of lingustic preprocessing and statistical machine learning has been proven to improve the overall accuracy for SIL problems ((67)). The question is: Will it also improve the results on the special task of extracting relations using multiple instance learning?

To answer this question we used domain independent shallow parsing modules and annotated the documents. The modules have been applied to the data of B.

The lingustic preprocessing consisted of the following parts:

- POS Tags: We used the POS Tagger by (36) and trained it with data of the Penn Treebank (61). The accuracy using 10-fold-cross validation is about 97%. With this information we filtered out words except nouns, verbs and adjectives.

- Stemming: The documents have been stemmed using the Porter Stemmer (47).

- Stemming + POS Tags and filtering out words except nouns, verbs and adjectives

- Dumb Parsing: The Dumb Parsing contains six steps for filtering snippets:

  1. Remove all tokens between the first occuring *NEs* and the last occuring verb group before the second NE:

2. Remove all tokens before the first occuring *NEs* except verb groups

3. If the snippet looks like ..... entity1 (... entity2) ..... return entity1 (.... entity2)

4. .... entity1 .... PREP entity2 $->$ ... entity1 Prep entity2 (if no verbgroup is between entity1 and entity2)

5. ... entity2's entity1.... $->$ entity2's entity1

6. ... VG ... entity2, entity1 .... $->$ VG entity2 entity1

Here are some examples showing the results of the steps above on real text snippets:

Original; *He's created this anarchic, you know, entity1-like, punk rock Joker - unlike any Joker ... Ledger died on January 22 of an accidental entity2 of prescription drugs.*

Result: *entity1-like died on january 22 of an accidental entity2*

Original: *entity1 part of God Save the Sex Pistols, featuring Nancy Spungen, the Swindle, My Way ... His death was ruled an accidental entity2. Immediately she heard the news, Sid's ...*

Result: *entity1 was ruled an accidental entity2*

Original: *In this autobiographical report Entity1 deals with his career as a body-builder and an American businessman. He was born in a small town near Graz, Entity2 and ...*

Result: *entity1 was born in a small town near graz , entity2*

As a baseline, again we tried linguistic preprocessing on the relation *company_acquisition(company, company)* and compared the results achieved (see figure C.1 and C.2). For this relation we see that the ROC curve has improved very much especially for the POS filtering. Several interesting things happen for the rest of the example relations.

Some results show improvements especially when putting more work into the preprocessing algorithms. Nevertheless, the overall gain is likewise poor, for some relations the results even drop dramatically (*placeOfDeath(person,place)*). As outlined in the chapter 3.2 web snippets are hard to process because they are not necessarily contiguous pieces of texts.

This is not only caused by the different style of the "snippet language", but also because NLP tools are usually trained on linguistically more well–formed sources like newspaper articles. However, because of the already mentioned problems with this approach for our system (see chapter 4.2.1.2) we did not go deeper into research in this direction.
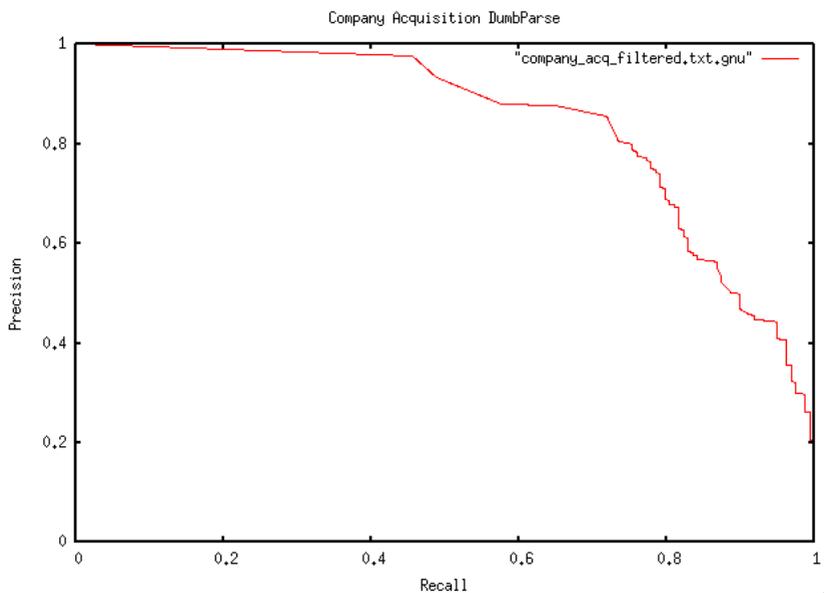

Figure C.1: Company acquisition using dumb parsing
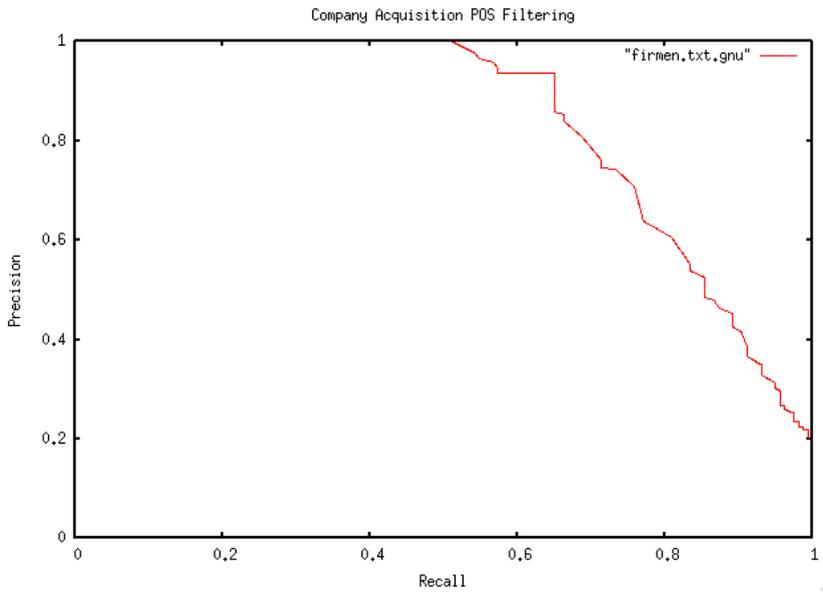
Figure C.2: Company acquisition using POS filtering
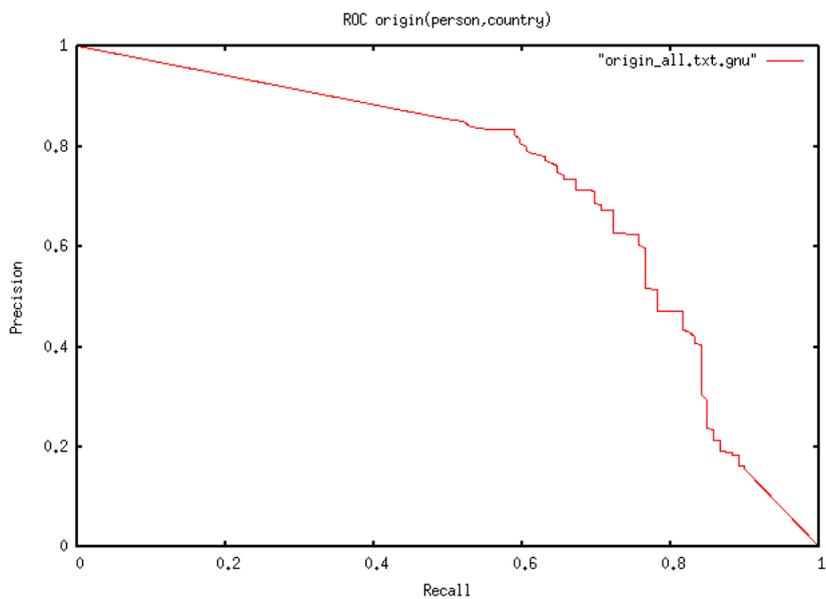
Figure C.3: Preprocessed ROC for origin(person,country)
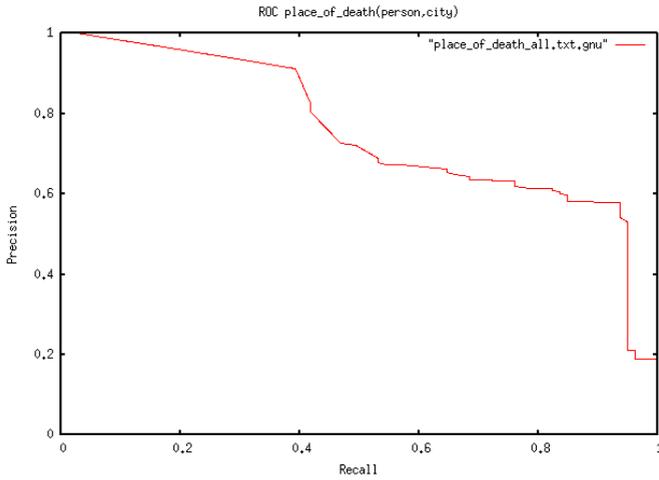
Figure C.4: Preprocessed ROC for placeOfDeath(person,city)
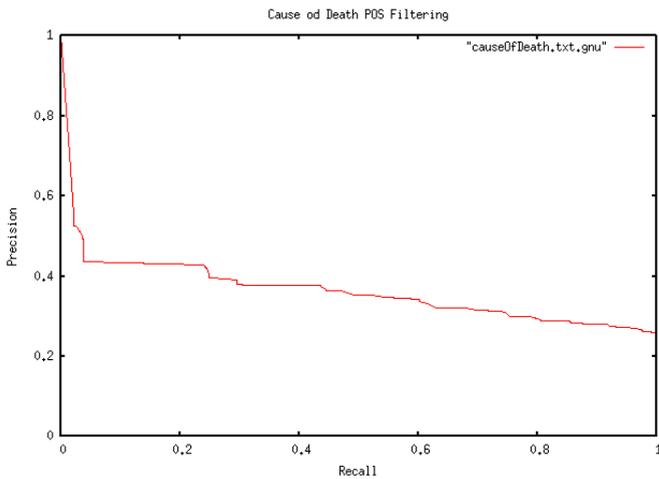
Figure C.5: Preprocessed ROC for causeOfDeath(person,cause)

Bibliography

[1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *In Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 85–94, 2000.

[2] Mehrnaz Sadat Akhavi, Mohammad Rahmati, and Nerssi Nasiri Amini. 3d visualization of hierarchical clustered web search results. In *Proceedings of the Computer Graphics, Imaging and Visualisation*, CGIV '07, pages 441–446, Washington, DC, USA, 2007. IEEE Computer Society.

[3] E. Alfonseca and S. Manandhar. An unsupervised method for general named entity recognition and automated concept discovery. In *Proceedings of the 1st International Conference on General Word-Net*, India, 2002.

[4] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge, MA, 2003.

[5] Marco Baroni and Stefan Evert. Statistical methods for corpus exploitation. In *In A. Lüdeling and M. Kytö (eds.), Corpus Linguistics. An International Handbook, Mouton de Gruyter, Berlin*, 2008.

[6] M. Becker, W. Drozdzynski, H. Krieger, J. Piskorski, U. Schäfer, and F. Xu. Sprout – shallow processing with typed feature structures and unification. In *Proceedings of the International Conference on NLP (ICON 2002)*, 2002.

[7] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Soren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. In *Web Semantics: Science, Services and Agents on the World Wide Web 7 (3): 154-65*, 2009.

[8] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In Gregory F. Cooper and Serafin Moral, editors, *UAI*, pages 43–52. Morgan Kaufmann, 1998.

[9] Sergey Brin. Extracting patterns and relations from the world wide web. In Paolo Atzeni, Alberto O. Mendelzon, and Giansalvatore Mecca, editors, *WebDB*, volume 1590 of *Lecture Notes in Computer Science*, pages 172–183. Springer, 1998.

[10] Jake D. Brutlag, Hilary Hutchinson, and Maria Stone. User preference and search engine latency. In *Google: JSM Proceedings, Qualtiy and Productivity Research Section*, 2008.

[11] R. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL*, volume 6, pages 9–16, 2006.

[12] Razvan C. Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In *Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS)*, Vancouver, BC, December 2005.

[13] Razvan C. Bunescu and Raymond J. Mooney. Learning to extract relations from the web using minimal supervision. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL*. The Association for Computational Linguistics, 2007.

[14] Ana Cardoso-Cachopo and Arlindo Oliveira. Combining lsi with other classifiers to improve accuracy of single-label text categorization. In *Proceedings of EWLSATEL 2007 — First European Workshop on Latent Semantic Analysis in Technology Enhanced Learning, Heerlen The Netherlands, March 29-30, 2007*, 2007.

[15] Zhixiang Chen and Bin Fu. On the complexity of rocchio's similarity-based relevance feedback algorithm. *JASIST*, 58(10):1392–1400, 2007.

[16] Paul Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschütter. Using odp metadata to personalize search. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 178–185, New York, NY, USA, 2005. ACM.

[17] Yaacov Choueka. Looking for needles in a haystack or locating interesting collocational expressions in large textual databases. In Christian Fluhr and Donald E. Walker, editors, *RIAO*, pages 609–624. CID, 1988.

[18] K. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[19] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL*, volume 2007, pages 708–716, 2007.

[20] D. Davidov, A. Rappoport, and M. Koppel. Fully unsupervised discovery of concept-specific relationships by web mining. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic, 2007.

[21] Emilio Di Giacomo, Walter Didimo, Luca Grilli, and Giuseppe Liotta. Graph visualization techniques for web clustering engines. *IEEE Transactions on Visualization and Computer Graphics*, 13:294–304, March 2007.

[22] Shipra Dingare, Malvina Nissim, Jenny Finkel, Claire Grover, and Christopher D. Manning. A system for identifying named entities in biomedical text: How results from two evaluations reflect on both the system and the evaluations. In *Comparative and Functional Genomics 6:pp 77-85*, 2004.

[23] Doug Downey, Stefan Schoenmackers, and Oren Etzioni. Sparse information extraction: Unsupervised language models to the rescue. In *ACL*. The Association for Computer Linguistics, 2007.

[24] Kathrin Eichler, Holmer Hemsen, Markus Löckelt, Günter Neumann, and Norbert Reithinger. Interactive dynamic information extraction. In Andreas Dengel, Karsten Berns, Thomas M. Breuel, Frank Bomarius, and Thomas Roth-Berghofer, editors, *KI*, volume

5243 of *Lecture Notes in Computer Science*, pages 54–61. Springer, 2008.

[25] Kathrin Eichler, Holmer Hemsen, and Günter Neumann. Unsupervised relation extraction from web documents. In *LREC*. European Language Resources Association, 2008.

[26] Oren Etzioni. Machine reading of web text. In Derek H. Sleeman and Ken Barker, editors, *K-CAP*, pages 1–4. ACM, 2007.

[27] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

[28] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134, 2005.

[29] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545. ACL, 2011.

[30] Alejandro Figueroa and Günter Neumann. Language independent answer prediction from the web. In *In proceedings of the 5th Fin-TAL, Finland*, 2006.

[31] Alejandro Figueroa, Günter Neumann, and John Atkinson. Searching for definitional answers on the web using surface patterns. *Computer*, 42(4):68–76, 2009.

[32] James R. Foulds and Eibe Frank. A review of multi-instance learning assumptions. *Knowledge Eng. Review*, 25(1):1–25, 2010.

[33] Susan Gauch, Jason Chaffee, and Alexander Pretschner. Ontology-based personalized search and browsing. *Web Intelligence and Agent Systems*, 1(3-4):219–234, 2003.

[34] Eugenie Giesbrecht and Stefan Evert. Part-of-speech tagging - a solved task? an evaluation of pos taggers for the web as corpus. In *In proceedings of the 5th Web as Corpus Workshop, San Sebastian, Spain*, 2009.

[35] Daniel Gillick. Sentence boundary detection and the problem with the u.s. In *HLT-NAACL (Short Papers)*, pages 241–244. The Association for Computational Linguistics, 2009.

[36] Jesus Gimenez and Lluis Marquez. Svmtool: A general pos tagger generator based on support vector machines. In *LREC*. European Language Resources Association, 2004.

[37] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proc. 25th Internat. Conf. on Very Large Data Bases*, pages 518–529. Morgan Kaufmann Publishers Inc., 1999.

[38] Mark A. Greenwood and Mark Stevenson. Improving semi-supervised acquisition of relation extraction patterns. In *Proc. of the Information Extraction Beyond The Document Workshop (COLING/ACL 2006)*, pages 29–35, Sydneyand Australia, 2006.

[39] Ralph Grishman and Beth Sundheim. Message understanding conference-6: a brief history. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*, COLING '96, pages 466–471, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[40] Xianpei Han and Jun Zhao. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 215–224, New York, NY, USA, 2009. ACM.

[41] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discovering relations among named entities from large corpora. In Donia Scott, Walter Daelemans, and Marilyn A. Walker, editors, *ACL*, pages 415–422. ACL, 2004.

[42] Erik Hatcher, Otis Gospodnetic, and Mike McCandless. *Lucene in Action*. Manning, 2nd revised edition. edition, 8 2010.

[43] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 517–526, New York, NY, USA, 2002. ACM.

[44] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *In Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[45] Orland Hoeber and Xue Dong Yang. Interactive web information retrieval using wordbars. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '06, pages 875–882, Washington, DC, USA, 2006. IEEE Computer Society.

[46] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[47] David A. Hull. Stemming algorithms: A case study for detailed evaluation. *JASIS*, 47(1):70–84, 1996.

[48] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 271–279, New York, NY, USA, 2003. ACM.

[49] John S. Justeson and Slava M. Katz. Technical terminology some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27, 1995.

[50] Mika Käki. Findex: search result categories help users when document ranking fails. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 131–140, New York, NY, USA, 2005. ACM.

[51] Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. The sketch engine. In *Proceedings of EURALEX*, pages 105–116, 2004.

[52] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145. Morgan Kaufmann, 1995.

[53] Thomas Landauer, Danielle S. McNamara, Simon Dennis, and Walter Kintsch. *Handbook of latent semantic analysis*. Lawrence Erlbaum Associates, Mahwah N.J., 2007.

[54] Anton Leuski and James Allan. Lighthouse: Showing the way to relevant information. In Jock D. Mackinlay, Steven F. Roth, and Daniel A. Keim, editors, *INFOVIS*, pages 125–129. IEEE Computer Society, 2000.

[55] V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.

[56] Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[57] Dekang Lin. Automatic identification of non-compositional phrases. In Robert Dale and Kenneth Ward Church, editors, *ACL*. ACL, 1999.

[58] Zhengzhong Liu and Qin Lu. High performance clustering for web person name disambiguation using topic capturing. *Ratio*, 2011.

[59] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[60] Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49:41–46, April 2006.

[61] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[62] Ryan T. McDonald, Fernando C. N. Pereira, Seth Kulick, R. Scott Winters, Yang Jin, and Peter S. White. Simple algorithms for complex relation extraction with applications to biomedical ie. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer, editors, *ACL*. The Association for Computer Linguistics, 2005.

[63] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, August 2006. ACM.

[64] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *J. Log. Program.*, 19/20:629–679, 1994.

[65] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[66] David Nadeau, Peter D. Turney, and Stan Matwin. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In Luc Lamontagne and Mario Marchand, editors, *Canadian Conference on AI*, volume 4013 of *Lecture Notes in Computer Science*, pages 266–277. Springer, 2006.

[67] Günter Neumann and Sven Schmeier. Shallow natural language technology and text mining. *KI*, 16(2):23–26, 2002.

[68] Günter Neumann and Sven Schmeier. A mobile touchable application for online topic graph extraction and exploration of web content. In *Proceedings of the ACL-HLT 2011 System Demonstrations*, pages 20–25, Portland, Oregon, June 2011. Association for Computational Linguistics.

[69] Günter Neumann and Sven Schmeier. Exploratory search on the mobile web. In Joaquim Filipe and Ana L. N. Fred, editors, *ICAART (1)*, pages 82–91. SciTePress, 2012.

[70] Günter Neumann and Sven Schmeier. Guided exploratory search on the mobile web. In Ana L. N. Fred, Joaquim Filipe, Ana L. N. Fred, and Joaquim Filipe, editors, *KDIR*, pages 65–74. SciTePress, 2012.

[71] Günter Neumann and Sven Schmeier. *Interactive Topic Graph Extraction and Exploration of Web Content*, pages 1–24. Theory and Applications of Natural Language Processing. Springer, 6 2012.

[72] Tien Nguyen and Jun Zhang. A novel visualization model for web search results. *IEEE Transactions on Visualization and Computer Graphics*, 12:981–988, September 2006.

[73] Brendan O'Connor, Michel Krieger, and David Ahn. Tweetmotif: Exploratory search and topic summarization for twitter. In William W. Cohen and Samuel Gosling, editors, *ICWSM*. The AAAI Press, 2010.

[74] Stanislaw Osinski, Jerzy Stefanowski, and Dawid Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent Information Systems*, pages 359–368, 2004.

[75] Stanislaw Osinski and Dawid Weiss. Carrot2: Design of a flexible and efficient web information retrieval framework. In Piotr S. Szczepaniak, Janusz Kacprzyk, and Adam Niewiadomski, editors, *AWIC*, volume 3528 of *Lecture Notes in Computer Science*, pages 439–444. Springer, 2005.

[76] Stanislaw Osinski and Dawid Weiss. Carrot2: Making sense of the haystack. *ERCIM News*, 2008(74), 2008.

[77] Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *AAAI*, pages 1400–1405. AAAI Press, 2006.

[78] Feng Qiu and Junghoo Cho. Automatic identification of user interest for personalized search. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *WWW*, pages 727–736. ACM, 2006.

[79] Lisa F. Rau. Extracting company names from text. In *In Proc. Conference on Artificial Intelligence Applications of IEEE.*, pages 29–32, 1991.

[80] Binyamin Rosenfeld and Ronen Feldman. Ures : an unsupervised web relation extraction system. In *ACL*. The Association for Computer Linguistics, 2006.

[81] D. Roth and W. Yih. Relational learning via propositional algorithms an information extraction case study. In *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001. Seattle.

[82] Mark Sanderson. Ambiguous queries: test collections need more sense. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 499–506, New York, NY, USA, 2008. ACM.

[83] Inessa Seifert, Kathrin Eichler, Holmer Hemsen, Sven Schmeier, Michael Kruppa, Norbert Reithinger, and Günter Neumann. Dilia - a digital library assistant - a new approach to information discovery through information extraction and visualization. In Kecheng Liu, editor, *KMIS*, pages 180–185. INSTICC Press, 2009.

[84] Satoshi Sekine. On-demand information extraction. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *ACL*. The Association for Computer Linguistics, 2006.

[85] Violeta Seretan and Eric Wehrli. Accurate collocation extraction using a multilingual parser. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *ACL*. The Association for Computer Linguistics, 2006.

[86] Violeta Seretan and Eric Wehrli. Multilingual collocation extraction with a syntactic parser. *Language Resources and Evaluation*, 43(1):71–85, 2009.

[87] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[88] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 824–831, New York, NY, USA, 2005. ACM.

[89] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the NAACL Conference on Human Language Technology (HLT)*, New York, 2006.

[90] Yusuke Shinyama and Satoshi Sekine. Named entity discovery using comparable news articles. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04. Association for Computational Linguistics, 2004.

[91] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Pearson Addison-Wesley, Upper Saddle River, NJ, 5. edition, 2009.

[92] F. Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1):143–177, 1993.

[93] A. Sping, D. Wolfram, M.B.J. Jansen, and T. Saracevic. Searching the web: The public and their queries. *Journal of the American Society for Information Science and Technology*, pages 226–334, 2001.

[94] Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. An improved extraction pattern representation model for automatic ie pattern acquisition. In Erhard W. Hinrichs and Dan Roth, editors, *ACL*, pages 224–231. ACL, 2003.

[95] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 675–684, New York, NY, USA, 2004. ACM.

[96] Jian-Tao Sun, Hua-Jun Zeng, Huan Liu, Yuchang Lu, and Zheng Chen. Cubesvd: a novel approach to personalized web search. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 382–390, New York, NY, USA, 2005. ACM.

[97] M. Timonen, P. Silvonen, and M. Kasari. Classification of short documents to categorize consumer opinions. In *Advanced Data Mining and Applications - 7th International Conference, ADMA 2011*, 2011.

[98] P.D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the 12th European Conference on Machine Learning*, pages 491–502. Springer-Verlag, 2001.

[99] Peter D. Turney. Measuring semantic similarity by latent relational analysis. *CoRR*, abs/cs/0508053, 2005.

[100] Peter D. Turney. Expressing implicit semantic relations without supervision. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 313–320, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[101] Ryen W. White and Resa A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan & Claypool Publishers, [San Rafael, Calif.], 2009.

[102] Fei-Yu Xu. *Bootstrapping relation extraction from semantic seeds*. PhD thesis, Saarland University, 2008. http://d-nb.info/987755218.

[103] Roman Yangarber and Ralph Grishman. Machine learning of extraction patterns from un-annotated corpora. In *Proceedings of the 14th European Conference on Artificial Intelligence: ECAI-2000 Workshop on Machine Learning for Information Extraction (2000) Berlin, Germany*, 2000.

[104] Alexander Yates. *Information Extraction from the Web: Techniques and Applications.* PhD thesis, 2007.

[105] Oren Zamir and Oren Etzioni. Grouper: a dynamic clustering interface to web search results. In *Proceedings of the eighth international conference on World Wide Web*, WWW '99, pages 1361–1374, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[106] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106, 2003.

The goal of this thesis is to provide a general framework (MobEx) for exploratory search especially on mobile devices. The central part is the design, implementation, and evaluation of several core modules for on-demand unsupervised information extraction well suited for exploratory search on mobile devices and creating the MobEx framework. These core processing elements, combined with a multitouchable user interface specially designed for two families of mobile devices, i.e. smartphones and tablets, have been finally implemented in a research prototype. The initial information request, in form of a query topic description, is issued online by a user to the system. The system then retrieves web snippets by using standard search engines. These snippets are passed through a chain of NLP components which perform an on-demand or ad-hoc interactive Query Disambiguation, Named Entity Recognition, and Relation Extraction task. By on-demand or ad-hoc we mean the components are capable to perform their operations on an unrestricted open domain within special time constraints. The result of the whole process is a topic graph containing the detected associated topics as nodes and the extracted relationships as labelled edges between the nodes. The Topic Graph is presented to the user in different ways depending on the size of the device she is using. Various evaluations have been conducted that help us to understand the potentials and limitations of the framework and the prototype.