Professional Search in Pharmaceutical Research

Alex Kohn

München 2009

Professional Search in Pharmaceutical Research

Alex Kohn

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität
München

vorgelegt von
Alex Kohn

München, den 24.11.2009

Erstgutachter:	Prof. Dr. François Bry (Ludwig-Maximilians-Universität München)
Zweitgutachter:	Prof. Dr. Steffen Staab (Universität Koblenz-Landau)
Tag der mündlichen Prüfung:	19.01.2010

Abstract

In the mid 90s, visiting libraries – as means of retrieving the latest literature – was still a common necessity among professionals. Nowadays, professionals simply access information by 'googling'. Indeed, the name of the Web search engine market leader "Google" became a synonym for searching and retrieving information. Despite the increased popularity of search as a method for retrieving relevant information, at the workplace search engines still do not deliver satisfying results to professionals.

Search engines for instance ignore that the relevance of answers (the satisfaction of a searcher's needs) depends not only on the query (the information request) and the document corpus, but also on the working context (the user's personal needs, education, etc.). In effect, an answer which might be appropriate to one user might not be appropriate to the other user, even though the query and the document corpus are the same for both. Personalization services addressing the context become therefore more and more popular and are an active field of research.

This is only one of several challenges encountered in 'professional search': How can the *working context* of the searcher be incorporated in the ranking process; how can unstructured free-text documents be enriched with *semantic information* so that the information need can be expressed precisely at query time; how and to which extent can *a company's knowledge* be exploited for search purposes; how should data from distributed sources be accessed from into *one-single-entry-point*.

This thesis is devoted to 'professional search', i.e. search at the workplace, especially in industrial research and development. We contribute by compiling and developing several approaches for facing the challenges mentioned above. The approaches are implemented into the prototype YASA (Your Adaptive Search Agent) which provides meta-search, adaptive ranking of search results, guided navigation, and which uses domain knowledge to drive the search processes. YASA is deployed in the pharmaceutical research department of Roche in Penzberg – a major pharmaceutical company – in which the applied methods were empirically evaluated.

Being confronted with mostly unstructured free-text documents and having barely explicit metadata at hand, we faced a serious challenge. Incorporating semantics (i.e. formal knowledge representation) into the search process can only be as good as the underlying data. Nonetheless, we are able to demonstrate that this issue can be largely compensated by incorporating automatic metadata extraction techniques. The metadata we were able to extract automatically was not perfectly accurate, nor did the ontology we applied contain considerably "rich semantics". Nonetheless, our results show that already the little semantics incorporated into the search process, suffices to achieve a significant improvement in search and retrieval.

We thus contribute to the research field of context-based search by incorporating the working context into the search process – an area which so far has not yet been well studied.

Zusammenfassung

Die seit den 90er Jahren vorherrschende Informationsflut als auch das Aufkommen neuer Technologien haben die Prozesse des Informationszugriffes auf nie dagewesene Art und Weise geprägt. Das Ergebnis dieses Wandels ist, daß Menschen heutzutage nach Informationen ‚googeln' anstatt Bibliotheken zu durchstöbern. Tatsächlich ist der Name des derzeitigen Internet-Suchmaschinen-Marktführers Google zu einem Synonym für die Suche nach Informationen geworden. Dieses Phänomen betrifft insbesondere auch Experten, für die Suche nach Informationen ein Teil des alltäglichen Geschäftes ist. Folglich sind Suchmaschinen nicht nur im Web die erste Wahl um Informationen zu finden sondern auch im Intranet von Firmen.

Obwohl die Verwendung von Suchmaschinen bei Experten – insbesondere bei Fachkräften in Unternehmen – sehr populär geworden ist, liefern Suchmaschinen im Intranet immer noch nicht zufriedenstellende Ergebnisse.

Eine mögliche Ursache unter anderen ist, daß Suchmaschinen häufig den Kontext des Suchenden (persönliche Bedürfnisse, Hintergrundwissen, usw.) ignorieren. Tatsächlich ist aber die Relevanz eines Suchergebnisses, nicht nur von der eigentlichen Suchanfrage und der Dokumentsammlung abhängig, sondern auch vom Arbeitskontext des Suchenden. Folglich kann eine Antwort – bei gleichbleibender Suchanfrage und identischem Korpus – für den einen Benutzer relevant sein und für den anderen Benutzer nicht. Die Einbeziehung des Kontexts bei der Suche ist ein aktives Forschungsfeld und wird zunehmend auch in Personalisierungsdiensten führender Internet-Suchmaschinen berücksichtigt.

Kontext-basierte Suche ist nur eine von vielen Herausforderungen im Umfeld von spezialisierten Suchmaschinen: Wie kann der *Arbeitskontext des Suchenden* in die Ermittlung der Rangfolge der Ergebnisdokumente einbezogen werden; Wie können vorhandene Daten mit *semantischen Informationen* bereichert werden, so daß die Frage präzise formuliert werden kann; Wie und zu welchem ausmaß kann das *Vorwissen eines Unternehmens* dazu genutzt werden die Suche zu verbessern; Wie sollen *verteilte Daten* in einem Suchportal zusammengefaßt werden.

Die vorliegende Dissertation befaßt sich dem Thema „Expertensuche", d.h. Suche am Arbeitsplatz, insbesondere in der Forschung und Entwicklung. Ein Beitrag dieser Arbeit liegt in der Zusammenstellung und Entwicklung von Ansätzen, mit denen den zuvor genannten Herausforderungen begegnet werden kann. Die Ansätze werden in dem Prototyp YASA (Your Adaptive Search Agent) implementiert, welcher Meta-Suche, adaptive Sortierung von Suchergebnissen und unterstütztes Navigieren ermöglicht. Zahlreiche Prozesse profitieren dabei von domänen-spezifischem Wissen. YASA wird in der pharmazeutischen Forschungsabteilung von Roche in Penzberg (ein größeres Pharma Unternehmen) produktiv genutzt. Letzteres bietet ein ideales Umfeld für die empirische Untersuchung der angewandten Prinzipien.

Die überwiegende Speicherung der Daten in Form unstrukturierter Textdokumente und das Fehlen expliziter Metadaten, stellten eine ernste Herausforderung dar. Die Einbindung von Semantik (traditionell als formale Wissensrepräsentation verstanden) kann nämlich nur so gut sein wie die zugrundeliegenden Daten. Nichtsdestotrotz sind wir in der Lage dieses Problem durch Einbindung automatischer Metadaten-Extraktionsmethoden weitgehend zu umgehen. Die Metadaten, welche wir extrahieren konnten, waren weder perfekt noch war die daraus resultierende und von uns verwendete Ontologie semantisch betrachtet besonders reich. Unsere Ergebnisse zeigen aber, daß bereits ein bißchen Semantik die Informationsbeschaffung deutlich erleichtert.

Der Beitrag der Arbeit liegt also auf dem Gebiet der kontext-basierten Suche, d.h. der Einbeziehung des Arbeitskontexts in den Suchprozeß – ein Gebiet, welches bis jetzt noch nicht gut erforscht wurde.

Acknowledgements

"It is with words as with sunbeams. The more they are condensed, the deeper they burn."

Robert Southey (1774 – 1843)

I would like to thank my scientific mentor and advisor Prof. François Bry, who – always optimistic and positive – helped me discover my research interests as well as to find and shape my ideas. I appreciated the discussions with him, his feedback, and his friendly attitude throughout my dissertation. I am also grateful to Prof. Steffen Staab for his willingness to scientific cooperation. The experience and knowledge he provided were particularly enlightening and helpful for my research.

Next, I thank my supervising tutor Dr. Alexander Manta at Roche and the company itself for offering me the scientific and financial opportunity to pursue my doctoral thesis. In countless discussions, Alexander guided me in times of despair, doubt, and idea hunting. His open mindedness gave me a lot of flexibility and freedom during my research. Special thanks to my colleague Dr. Stefan Klostermann who provided valuable insights about research at Roche and guidance during my first year. Thanks to my colleagues from In Silico Sciences who gave me advice and support in all questions related to bioinformatics and statistics. Thanks to the technical staff of Scientific Research Informatics for their support with servers and hardware upgrades. Special thanks to all colleagues from Pharma Research who participated in the evaluation of YASA. Finally, a great and warm thanks to Tobias Högel, who wrote his diploma thesis under my supervision, and to Florian Stadler and Marc Gössling, whom I partially supervised during their internship at Roche. Their contribution and dedication helped me tremendously.

Thanks to my current as well as past fellow students in the PMS department at the University of Munich, Paula-Lavinia Pătrânjan, Sacha Berger, Edgar Stoffel, Tim Furche, Benedikt Linse, Michael Eckert, Stephan Leutenmayr, Alexander Pohl, Christoph Wieser, Jakub Kotowski, Klara Weiand, and Olga Poppe who made the time at the university always a pleasure. Special thanks to Dr. Norbert Eisinger who gave me valuable feedback about my work. Thanks also to our secretary Ingeborg von Troschke, who helped me a lot with administrative tasks.

Most importantly I would like to thank my friends and my family for their love and care. Special thanks to my fiancée who absorbed so much of my stress, gave me energy and who made me take my mind of things, even though she is quite busy pursuing her master scholar.

Last, I would like to thank everyone who pushed me to finish this thesis.

Contents

I Prelude

Chapter 1 Introduction 3
1.1 Motivation 4
1.2 Hypotheses 4
1.3 Contributions 5
1.4 Structure of the thesis 6

II Background

Chapter 2 Search for information 13
2.1 Information retrieval process overview 14
2.2 Traditional retrieval models 16
2.2.1 Boolean model 17
2.2.2 Vector space model 19
2.2.3 Term weighting 21
2.2.4 Discussion 24
2.3 Text processing 25
2.3.1 Tokenization 25
2.3.2 Stopword removal 26
2.3.3 Stemming and lemmatization 26
2.4 Search in the World Wide Web 27
2.4.1 PageRank in a nutshell 28
2.4.2 HITS in a nutshell 30
2.4.3 Web search engines 31
2.4.4 Discussion 33
2.5 Search in an intranet environment 35
2.5.1 Differences between intranet search and Web search 35
2.5.2 Open issues in intranet search 37
2.5.3 Search in structured sources 39
2.5.4 Enterprise search engines 40
2.6 Evidence for document relevance 44
2.6.1 Content evidence 45
2.6.2 Context evidence 46
2.6.3 Time evidence 46
2.6.4 Hyperlink evidence 47
2.6.5 URL evidence 47
2.6.6 Feedback evidence 48
2.7 Precision and recall 48

Chapter 3 Adaptation in Information Retrieval 51
3.1 User modeling 51

3.1.1 User model types 52
3.2 Personalized search 54
3.2.1 Personalization process types 55
3.2.2 Methods for personalized search 56
3.2.3 Personalized search based on the search history 58
3.2.4 Adaptation of search results based on the search history 61
3.2.5 Other applications of personalized search 61
3.3 Recommender systems 62
3.4 Algorithms for recommender systems 64
3.4.1 Memory-based algorithms 65
3.4.2 Model-based algorithms 67
3.5 Discussion 67

Chapter 4 Semantic Technologies 71
4.1 Semantic Web 71
4.1.1 Resource Description Framework (RDF) 73
4.1.2 RDF Schema (RDFS) 75
4.1.3 Web Ontology Language (OWL) 77
4.2 F-Logic 78
4.2.1 The is-a hierarchy 79
4.2.2 The object base 79
4.2.3 Rules and queries 80
4.2.4 F-Logic vs. OWL-DL 81
4.3 Semantic technologies in Information Retrieval 82
4.3.1 State of the art 82
4.3.2 Discussion 84
4.4 Semantic technologies in life sciences 85
4.4.1 State of the art 85
4.4.2 Discussion 86

III Core

Chapter 5 Characteristics of professional search in pharmaceutical research 89
5.1 Evaluation of the initial situation in the department investigated 89
5.1.1 Intranet web 90
5.1.2 File shares 91
5.1.3 Databases and applications 93
5.1.4 Search engine usage on the PRPZ-WebSite – A log file analysis 94
5.1.5 Empirical studies of information acquisition 95
5.2 Quality characteristics of a professional search tool 96
5.2.1 One single entry point 96
5.2.2 Role-specific ranking 96
5.2.3 Guided navigation 97
5.2.4 Exploit existing knowledge 98
5.2.5 Professional search beyond research and development 98

Chapter 6 An ontology-based information retrieval approach for professional search 101
6.1 Beyond traditional search for information 101
6.1.1 Key approaches 102
6.1.2 Targeted sources 104
6.2 Ontologies 105
6.2.1 Classification ontology 105
6.2.2 Annotation ontology 106
6.2.3 Adaptation ontology 108
6.3 Assigning metadata to “unstructured” text documents 108
6.3.1 Considered document classes 109
6.3.2 Text representation 110
6.3.3 Named Entity Recognition 111
6.3.4 Classification using a knowledge-engineering approach 112
6.3.5 Classification rules 113
6.3.6 Classification using Machine Learning 116
6.3.7 Discussion 118
6.4 Search results adaptation 119
6.4.1 Knowledge-based adaptation 119
6.4.2 Advanced user roles 122
6.4.3 Log-based adaptation 123
6.4.4 Extending knowledge-based adaptation with implicit feedback 124
6.5 Weighting document relevance 124
6.5.1 Similarity function 125
6.5.2 The context’s influence on the ranking of results 126
6.6 Discussion and related work 129

Chapter 7 The professional search agent prototype YASA 131
7.1 Implementation decisions 131
7.1.1 Search engine: Lucene 132
7.1.2 Semantic web middle-ware: OntoBroker 132
7.1.3 Text analysis platform: UIMA 133
7.1.4 Machine learning platform: WEKA 133
7.2 Architecture of YASA 134
7.2.1 Data adapter 135
7.2.2 Crawler 137
7.2.3 Annotator 137
7.2.4 Indexer and searcher 137
7.2.5 Graphical user interface 137
7.2.6 Service layer 139
7.3 Discussion 139

Chapter 8 Evaluation 141
8.1 Usage, access, query, and session statistics 141
8.1.1 Changes in the search engine usage on the PRPZ-WebSite – A log file analysis 142
8.1.2 Access of sources within YASA 144
8.1.3 Statistics concerning individual queries 146

8.1.4 Statistics concerning query duplicates 147
8.1.5 Statistics concerning query sessions 148
8.1.6 Statistics concerning click statistics 149
8.1.7 Discussion 149
8.2 Text categorization performance 150
8.2.1 Training and test set 150
8.2.2 Optimization of the ML text classifier 151
8.2.3 Knowledge engineering vs. machine learning 152
8.2.4 Discussion 154
8.3 Retrieval performance evaluation using click-through data 155
8.3.1 Design 156
8.3.2 Baseline ranking vs. baseline ranking with feedback 157
8.3.3 Baseline ranking vs. baseline ranking with context 158
8.3.4 Discussion 159
8.4 Controlled experiments 161
8.4.1 Evaluation process and methodology 161
8.4.2 Test persons 163
8.4.3 Designing the tasks 164
8.4.4 Test system 166
8.4.5 Conducting the evaluation 167
8.4.6 Results of the observation phase 167
8.4.7 Results of the feedback phase 172
8.4.8 Results interpretation 175
8.5 Discussion 176

IV Conclusion

Chapter 9 Summary and future work 181
9.1 Summary 181
9.2 Lessons learned 183
9.3 Future work 184
9.4 Outlook 186

Bibliography 189

List of Figures 201

List of Tables 204

Abbreviations 206

Publications of the author related to this thesis 207

Curriculum Vitae 208

Part I

Prelude

Chapter 1

Introduction

"An expert is a person who has made all the mistakes that can be made in a very narrow field."

Niels Bohr (1885 – 1962)

1.1 Motivation....4
1.2 Hypotheses....4
1.3 Contributions....5
1.4 Structure of the thesis....6

Once upon a time scientists were experts in their field. They knew not only the "hot questions" but also the scientists involved and the various approaches investigated. More important, they were well informed of novel research results. Gone are these favorable times! Hot issues and active research teams emerge with high pace and being informed early enough might be essential for success. Furthermore, no one can any longer keep an eye on the research publications, patents, and other information that might be relevant to one's research. As a consequence, scientists often feel – and in fact they sometimes are – rather unaware of areas that are of prime importance to their research. High diversity, considerable amounts of information and extremely fast communication are key characteristics of today's research – especially in medical biology.

The profiling and retrieval of the most relevant information for any scientist, based on similarities between roles, information needs and reading behavior, is a way to cope with these aspects of today's research. Systems providing such services are made possible by emerging *Semantic Web* techniques as well as established *Information Retrieval*, *Text Categorization*, and *Adaptation* methods.

This thesis investigates different approaches to satisfy the information needs of scientists in Pharmaceutical Research, based on the concrete needs of a research department at Roche in Penzberg, a research centre of a major pharmaceutical company. The approaches are implemented in an "Intelligent Information Portal" which functions as a platform for conducting several empirical studies.

1.1 Motivation

Search is an increasingly important method for information access and retrieval in large companies. Having experienced the possibilities of public search tools such as *Google* or *Yahoo!,* employees now expect a similar performance in context of their professional search activities.

However, public search significantly differs from 'professional search'. Relevance for instance is rarely the same in public and 'professional search'. Consider for example a hematologist searching for *"polycythemia vera",* a blood disorder, at *Google* (on the 4th of Mai 2009 at *www.google.de*). Due to *Google's* virtue to present most popular pages on any subject, the first five returned results link to encyclopedia like web pages which describe the blood disorder in a very general manner. Although these pages might be relevant to the average searcher, they are probably not what a pharmaceutical researcher is looking for. Even if *Google* first returned the pages relevant to pharmaceutical researchers, not all these pages would be relevant to every pharmaceutical researcher. Indeed, research is a highly specialized activity: different researchers have different interests. One might be interested in drugs for curing the blood disorder *"polycythemia vera";* another one might be interested in the causes of the disorder; a third one in the costs of the disorder for society in order to analyze the market possibilities of new drugs, etc. Thus, the *"one size fits all"* approach of public search engines is not appropriate for 'professional search'.

Another reason why public search engines are rarely appropriate for 'professional search' is that they do not access data behind a company's firewall, that is, the data that best reflects on-going professional activities. Further, public search engines are not focused on expert knowledge, the field where professional users are active. In case of pharmaceutical research, this expert knowledge is mostly expressed in databases, public life science ontologies, libraries, etc. It is worth noting that specialized search services of public search engines, such as *Google Scholar*, are important efforts to fulfill the need of 'professional search' for expert sources. In addition, search engines and information services targeted at specific areas are available, such as the life science literature search service *PubMed*.

1.2 Hypotheses

The previous section briefly outlined that public search is only to some extent – if at all – appropriate to professionals. The aim of this thesis is to determine which quality characteristics are most adequate for a professional search tool. In order to identify these (i.e. investigated approaches), we first discuss the following working hypotheses, on which the research described in this thesis is based:

1. Professional searchers prefer to access all sources via one single entry point
2. Professional searchers require a role-specific ranking of results
3. Guided navigation is relevant for professional search
4. A company's knowledge is instrumental for professional search

The *first hypothesis* is deduced from the fact that much of the success of public search tools such as Google, is because they offer a convenient way of searching and

accessing all kind of data from one single entry point (i.e. one search box). We assume that similarly, professional searchers would also prefer to have a search tool which gives access to large parts of the professional information sources. Therefore, we investigate in this thesis whether the approach of a meta-search engine (Chapter 2) – giving access to public domain data as well as in-house data – is preferred by professional users.

The *second hypothesis* results from the observation that the "one size fits all" paradigm applied by today's search engines is in contrast to research nowadays, which requires a high degree of specialization. As a consequence, we postulate that professional searchers require a role-specific ranking of results. Even though various adaptation techniques (Chapter 3) could be applied to achieve a role-specific ranking of results, we restrict our investigations to knowledge-based and log-based approaches. Knowledge-based methods are considered because they offer a convenient way of encoding static a priori knowledge (cf. fourth hypothesis) into the ranking process. Log-based methods on the other hand cover the dynamic parts as they adjust the adaptation based on past search activities. Whether the individual approaches or their combination improve search performance is one target of the investigations conducted in this thesis (Chapter 8.3).

The *third hypothesis* is given because guided navigation (faceted navigation) [Yee et al. 2003] enables professionals to "search as they think". Arguably, being able to browse information along multiple dimensions instead of a fixed taxonomy enables a user to find information faster as he can use his preferred way of structuring information. Facets have also a second effect: they provide a brief overview of the results' characteristics so that professionals are supported in case they conduct queries in unfamiliar topics. In order to provide facets, existing documents must be enriched with annotations (i.e. metadata). We investigate whether a knowledge-engineering or a machine learning approach are suitable for this task (Chapter 8.2). We then evaluate whether the automatically generated facets improve the information retrieval process of a user (Chapter 8.4).

The *fourth hypothesis* characterizes a key difference between public search and professional search: In 'professional search' a priori knowledge about the searcher is usually available whereas in a public search environment this is generally not the case. Professional searchers are part of a larger organization in which they have specific roles and tasks. These roles and tasks reflect their professional context and could thus be used as a priori knowledge so that information retrieval is improved (Chapter 6). We investigate the benefits of using a company's a priori knowledge in the context of hypothesis two (knowledge-based adaptation; cf. 6.4.1) and three (guided navigation; cf. 6.3.4).

1.3 Contributions

The contributions of this thesis are twofold. On the one hand, we present characteristics of professional search in a scientific community, devise a conceptual framework, and the architecture of a professional search tool. The characteristics being presented are derived from the working hypotheses and lead to a number of

applied principles for designing professional search tools. On the other hand, we provide an implementation of the principles in a prototype called YASA (Your Adaptive Search Agent) and we conduct an empirical evaluation of the applied principles in a real world setting using YASA.

YASA provides meta-search (cf. first hypothesis), adaptive ranking of search results (cf. second hypothesis), guided navigation (cf. third hypothesis), and it uses domain knowledge to drive several processes (cf. fourth hypothesis). The wiring of the methods is done in an intelligent and unique manner, achieving a novel approach in the context of professional search. The prototype is deployed in the pharmaceutical research department of Roche in Penzberg, with several hundred users.

The prototype (Chapter 7) can be considered an ideal platform for conducting experiments in the area of professional search. In this thesis we verify which of the principles (one single entry point, role-specific ranking of results, guided navigation, exploitation of contextual knowledge) are most instrumental for professional search. The availability of the prototype to several hundred employees enables us to collect sufficient empirical data in order to draw statistically significant conclusions to the given hypotheses.

Our main findings revolve around incorporating semantics into the search process so that context-based search is enabled – an area, which so far has not yet been well studied in research. Indeed, research in adaptation mostly focused on specific types of content as well as personalization services based on a user's search history. The incorporation of the working context as conducted in this thesis (Chapter 6) has not yet been well studied in research.

A further issue we address is the fact that our work targets unstructured free-text documents. Hence, the ability to extend search with semantics is limited by the quality of the symbolic knowledge representation, which itself is dependent on the quality of the indexed documents. This is a key issue because free-text documents do not contain explicit semantics.

We show that this limitation can be largely compensated by automatic metadata extraction techniques. Further, we show that even in light of the "sparse semantics" we are able to extract, the benefits for search are considerable. (cf. Chapter 8).

1.4 Structure of the thesis

The thesis is structured as follows. We begin with the background part (Chapter 2, 3, and 4), which introduces the basics of information retrieval, adaptation and semantic technologies and presents the main research directions in the area. We continue with the core part, which begins with the analysis of the peculiarities of professional search in research & development and identify topics and techniques to be investigated (Chapter 5). Following that, we introduce the concept of a professional search tool (Chapter 6) as well as the technical design decisions and architecture of the developed prototype (Chapter 7). Finally, we present the results of the empirical study conducted with the prototype and their consequences for 'professional search'

(Chapter 8). In the last part, we conclude the work (Chapter 9). A more detailed overview of the chapters is given next.

Part II: Background

Chapter 2: Search for information
We start with the introduction of the basics of search engines. Here, special focus is put on the information retrieval aspects. Then, we move on to search in the Web and its impacts on traditional information retrieval. Finally, we discuss search in intranets and its differences to search in the Web.

The chapter is fundamental for this thesis as it describes the state of the art in public search as well as in professional search. Further it points out current limitations and issues in context of professional search. Last, it provides the necessary background knowledge for the concepts introduced in Chapter 6.

Chapter 3: Adaptation in Information Retrieval
Adaptive techniques seem to be a promising approach for achieving acceptable precision and recall levels even with exponentially growing information volumes. The chapter introduces the fundamentals of adaptive systems. In particular, the user modeling, i.e. construction of user profiles and involved issues are discussed. Further, an overview of adaptive applications is given.

The chapter not only provides the foundation for understanding the role-specific ranking approach introduced in Chapter 6 but also goes beyond and discusses recommender systems. Even though these are not considered in context of this thesis, we believe that discussing them is necessary in order to provide a complete picture of adaptive systems.

Chapter 4: Semantic Technologies
We begin with the presentation of the Semantic Web vision and standards. Then, we introduce the ontology language F-Logic which is used in this thesis for capturing domain knowledge. Following that, the usage of semantic technologies in Information Retrieval systems is discussed. Finally, we examine the role of ontologies and rules for the health care & life science sector.

This chapter is of particular importance because ontologies and rules (written in F-Logic) are an integral part of the overall concept introduced in Chapter 6. More precisely: a priori knowledge of the professional domain is encoded in a semantic knowledge base; role-specific ranking of search results and the automatic annotation of documents is based on ontologies and rules.

Part III: Core

Chapter 5: Characteristics of professional search in pharmaceutical research
The chapter begins with an evaluation of the initial situation in a research department at Roche. The focus is on existing information sources and the way they are used. Difficulties and shortcomings are outlined and discussed. Based on this analysis, the working hypotheses of this thesis are justified and several quality characteristics a professional search tool should offer are suggested.

Chapter 6: An ontology-based information retrieval approach for professional search
This chapter introduces a concept for an ontology-based information retrieval portal. Each corner stone of the concept is described in detail and the technical feasibility is discussed.

In particular we describe the meta-search approach; the role-specific ranking of results based on knowledge-bases and logs; and guided navigation based on automatic annotations of text documents. In order to automatically enrich text documents with annotations we use ontologies and rules, Machine Learning, and Natural Language Processing. While the basic principles of knowledge bases are described in Chapter 4, the techniques of Machine Learning and Natural Language Processing are not addressed in the background part of this thesis. Rather we point to reference work in this area. The reason for skipping the discussion of the two topics is that these are merely auxiliary technologies.

Chapter 7: The professional search agent prototype YASA
Design decisions and the architecture of the prototype named "*Your Adaptive Search Agent*" (YASA) are outlined. YASA implements the concepts introduced in Chapter 6.

Chapter 8: Evaluation
The empirical results obtained from evaluating the professional search prototype YASA in a R&D context are presented. In particular, we evaluate: the usage of YASA compared to other search tools in the investigated department; we investigate query sessions of YASA and interpret them; we examine which annotation method works best, i.e. knowledge-engineering or machine learning; we examine whether our knowledge-based and log-based role-specific ranking methods outperform the baseline ranking; we conduct a user study in which various aspects of the prototype YASA are examined.

The chapter is the core contribution of this thesis as it shows which methods are suitable for a professional search tool in R&D and which are not.

Part IV: Conclusion

Chapter 9: Summary and future work

The last chapter gives a summary and conclusion of the conducted work. Further, we discuss the consequences of the obtained results and the applicability of the presented concepts in other environments. Finally, areas of future work are outlined.

Part II

Background

Chapter 2

Search for information

"As a general rule the most successful man in life is the man who has the best information."

Benjamin Disraeli (1804 – 1881)

2.1	Information retrieval process overview	14
2.2	Traditional retrieval models	16
2.3	Text processing	25
2.4	Search in the World Wide Web	27
2.5	Search in an intranet environment	35
2.6	Evidence for document relevance	44
2.7	Precision and recall	48

The "Sputnik shock" in the 1950s and the subsequent funding by U.S. government was a key enabler of early research in information retrieval. At that time, information retrieval was based on mechanized literature searching systems. The first computers enabled free text indexing to spread in the 1960s. In that time period, Gerald Salton set the foundations of modern search technology by introducing the Vector Space Model [Salton et al. 1975]. The 1970s are characterized by a large-scale adoption of computer typesetting which provided useful material for information retrieval systems. In the same decade, C. J. Van Rijsbergen proposed the probabilistic retrieval model [Van Rijsbergen 1979]. In the 1980s the digital information available grew further and in 1989, Tim Berners-Lee set the foundations for today's World Wide Web. The invention of the Web revolutionized the world as it enabled anyone to publish any content. Driven by the people's need to find information in the steadily growing Web, many researchers and companies started to create Internet search engines: Archie (1990; the first search tool for the Internet), I, Robot (1993; the first Web Wanderer), Excite (1993), EINet Galaxy (1994), Yahoo (1994), Lycos (1994), Infoseek (1995), Alta Vista (1995), Inktomi (1996), AskJeeves (1997), Northern Light (1997), Google (1997), and MSN Search (1998). The given list contains only the well known ones. In fact many more have been developed and some are still in use today.

This chapter introduces basic principles of today's modern information retrieval. Key aspects of professional search in the Web as well as in intranets are described.

We begin this chapter by giving an overview of the basic steps in the information retrieval process. In the subsequent chapters we discuss several aspects of Information Retrieval (IR) which are relevant for this thesis. In particular, we introduce the vector space model as this is the retrieval model which is applied by our concept. Then, we discuss text processing, i.e. how to represent text. Text processing is not only important for IR but also for text classification (cf. 6.3.2 and 8.2.2). Following that, we switch the topic to search in the Web. On the one hand, we introduce the ranking algorithms PageRank and HITS, which turned the Web search engine market upside down. On the other hand, we list several general as well as specialized search engines of the Web and discuss their strength and weaknesses with respect to professional search. Having introduced search in the Web, we then continue with search in an intranet environment. This chapter is of particular interest for this thesis as it outlines the typical problems and the current state of the art in intranet search. Following that, we talk about evidence for document relevance. Here, various options for improving the ranking of results in the Web as well as in intranets are shown. We briefly discuss which of the options are applicable in an intranet environment and which not. Finally, we give an introduction to the widely used performance measures precision and recall. These are also applied in various evaluations conducted in this thesis.

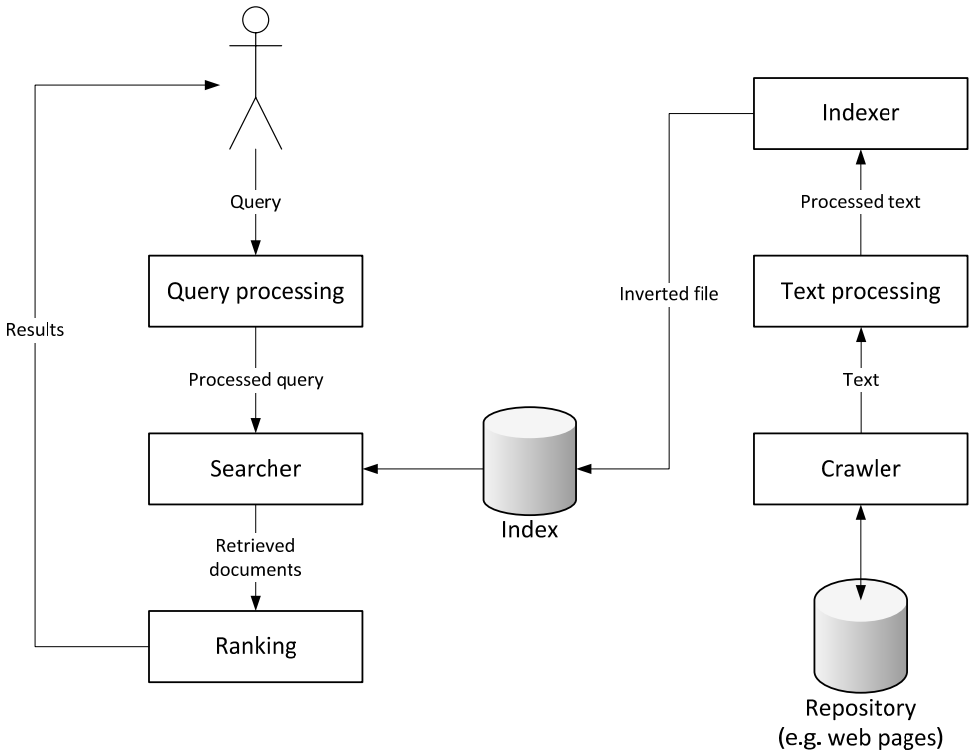
2.1 Information retrieval process overview

An IR system consists of several interconnected modules (Fig. 2-1) which enable two basic processes: Building an index and querying the index. The *index*, an integral part of the IR system, contains the searchable "features" (typically the words of text documents) and enables fast query answering.

Building a search index starts with a *crawler* (also known as robot, bot, or spider) [Heydon & Najork 1999]. Search engines use a crawler program in order to find new documents in a data repository. The crawler contains thus the algorithms to "crawl" (scan) specific data repositories. In case of the World Wide Web, discovery of new documents is accomplished by starting from a set of seed URLs and then recursively following all hyperlinks from the seed pages. The crawling process of file share repositories is quite similar to Web crawling. The crawler also starts from a set of seed pages or entry points. However, since no hyperlinks exist in file shares, the crawler simply uses the operating system's directory listing command in order to find new documents. Thus, it recursively scans the directory tree using a breadth-first-search or depth-first-search approach. Other types of repositories are crawled in a similar fashion. The crawler implementation needs to know the search scope, which is highly dependent on the targeted IR application. In one case, search in the full-text content does suffice. In another case, both, search in full-text and metadata is relevant (e.g. in Web search the body's content as well as the meta-tags' content of HTML pages). Yet in other scenarios, only metadata is considered. Literature retrieval tools e.g. might be restricted to search in metadata (author, title, journal, year of publication, and abstract), in order to protect the publisher's rights on the full-text content.

The output of the crawler is delegated to a *text processing* engine (Chapter 2.3) where several text manipulation tasks are conducted: tokenization (splitting sentences into words), removal of stop words (frequent words like “the”, “or”, “and” …), stemming (reducing words to their root form, e.g. “eating” becomes “eat”), and synonym injection (e.g. adding “research paper” to ”research article”) are just a few examples. The aim of the text processing pipeline is to identify index terms that best describe the content of a document and help to discriminate it from others. An index term is a “content-bearing key […] which needs not be single words but may be multi-word units” [Jones 1999]. For instance, the text “Acetyl-CoA” might be processed to the single index term “Acetyl-CoA” or to the index terms “Acetyl” and “CoA”, depending on what is relevant for the application field. The important point is that the same text processing pipeline must be applied to a user’s query. Otherwise discrepancies between indexed content and queries could lead to anomalies in the search results.

Fig. 2-1: Information retrieval process.



In the next step, the processed text is transferred to the *indexer* which builds an inverted file. The inverted file is an index structure where each indexed term is linked to the document in which it occurs, just like a book index where each term points to page numbers. The benefit of using an inverted file is that documents matching a query terms can be found efficiently.

Once the index is constructed, the IR system can be queried by its users. Analogous to the text processing step, the query is first analyzed by a text processing engine in exactly the same way, before it is transformed into an internal representation. The processed query is then transmitted to the searcher.

The *searcher* processes the query and retrieves all relevant documents from the index. Here, the investment of building the index pays off because thousands of queries can be answered within milliseconds.

In the final step, the retrieved documents are passed to the *ranking* module. This module plays a crucial role as its task is to order the results by relevance. The documents matching best the user's information need to be located at the top of the result list.

In the next chapter, parts of the information retrieval process are described in detail, i.e. the *query processing*, the *searcher*, and the *ranking* of results. Afterwards, the invention of the World Wide Web and its impacts on information retrieval as well as on the scientific community and their search behavior are discussed. Following that, the focus is set to search in intranet environments and the difficulties encountered there are outlined. Finally, state of the art methods for improving ranking of results are listed and discussed.

2.2 Traditional retrieval models

Formally, an information retrieval model is a quadruple $(D, Q, F, R(q_i,d_j))$ [Baeza-Yates & Ribeiro-Neto 1999], where

- D is a representation of documents,
- Q is a representation of queries (user information needs),
- F is a modeling framework for D, Q, and the relationships among them,
- $R(q_i,d_j)$ is a ranking function which defines an ordering among the documents for the given query.

In case the Boolean model (Chapter 2.2.1) is used, D is a set of words (indexing terms) occurring in the documents. Q is a Boolean expression of index terms composed with Boolean operators. F is the set algebra (in case of the Vector Space Model shown in Chapter 2.2.2 it would be the vector space and its operators). R predicts a document as relevant to a query if the document can be satisfied, i.e. the document matches the query. The components of the IR model can be derived in a similar fashion for other IR models.

In advance to the introduction of the retrieval models, we define two notions which are used in the subsequent chapters: "index term weight" and "index term weight vector". The definitions revolve around weighted terms – a key principle for measuring relevance in IR, which is usually incorporated in the retrieval models. For instance, an index term occurring in every document of the indexed corpus is not useful as it can't discriminate well between different documents. If, however, an index term occurs only in a small subset of documents, it is very useful for discriminating documents, and thus for making queries.

Definition 2-1: Index Term Weight

Let $D=\{d_1,\ldots,d_N\}$ be the set of all indexed documents. Further, let $K=\{k_1,\ldots,k_T\}$ be the set of all distinct index terms. Then, a term weight $w_{i,j}$ is a non-negative number reflecting the relevance of index term k_i for document d_j.

Index terms appearing in a document d_j, receive a positive weight $w_{i,j}>0$. The higher the number, the more relevant is the index term k_j for document d_j. In case an index term does not occur in a document, its weight is set to $w_{i,j}=0$. The scale of term weights, and if they are continuous or discrete depends on the model used for calculating them. In one case they might be binary (e.g. Boolean model), in another case they might be continuous and normalized to the range of [0..1] (e.g. vector space model), and so forth. Most methods for calculating the term weights are based on the "term frequency – inverse document frequency" (tf-idf) measure, which is introduced in chapter 2.2.2.

The index term weight as defined previously, represents the weight of an index term for a specific document. The index term weight vector, which is defined next, groups the index term weights of one document into a vector.

Definition 2-2: Index Term Weight Vector (Document Vector)
Let K be defined as previously, namely the set of all distinct index terms. Then, each document d_j is represented by a t-dimensional vector $d_j=(w_{1,j},\ldots,w_{T,j})$ where $w_{i,j}$ represents the weight of the index term k_i in the document d_j.

Modern IR has its roots in the 70's and so it is no surprise that nowadays dozens of different retrieval models and hundred of variations exists. The literature describes amongst others the Boolean model, the vector space model [Salton et al. 1975], the extended Boolean model [Salton et al. 1983], the probabilistic model [Robertson & Jones 1976; Robertson & Walker 1994], the fuzzy set model [Fox et al. 1992], the generalized vector space model [Wong et al. 1985], the latent semantic indexing model [Furnas et al. 1988], the neural network model [Wilkinson & Hingston 1991], the inference network model [Turtle & Croft 1989], and many variations of the basic approaches. Due to this large amount of different approaches we will put focus only on the Boolean model and on the vector space model. Although simple, these models perform well and are among the most popular in the field of IR. In addition, hybrid approaches which attempt to combine the advantages of the vector space model and the Boolean model are discussed.

2.2.1 Boolean model

Boolean algebra, formulated by George Boole in 1847 [Boolean algebra 2008], is the calculus of the binary truth-values 0 and 1, or F and T (False and True). Boolean algebra differs from ordinary numeric algebra because it applies a two element set of values which are not seen as numeric but as symbolic quantities. The Boolean algebra has three basic operations defined, namely conjunction ($\wedge$, AND), disjunction ($\vee$, OR) and negation ($\neg$, NOT). These logical connectives can be used to construct propositions. The truth-value of the resulting proposition is dependent on the truth-values of the components and on the connectives employed.

The Boolean model is not only based on Boolean algebra, but also on set theory. This is possible, because the basic Boolean operators ($\wedge$, $\vee$, and $\neg$) are isomorphic to the basic set operations (intersection, union and complement). The atomic terms

connected by the Boolean operators can be interpreted as representing propositions that are true or false, or as representing sets that do or do not contain certain items.

In the Boolean model we would thus express a query as a connective of terms, e.g. "antibody $\wedge$ cancer". Each query term specifies a set of matching documents (documents containing the query term). In the example, the set matching "antibody" would be intersected with the set of documents matching "cancer", yielding the set of documents containing both terms, "antibody" and "cancer", which is seen as the answer to the query.

Next, we define the index term weight for the Boolean model. The definition maps the binary truth values of the Boolean algebra to the index term weights.

Definition 2-3: Index Term Weight of Boolean model
Let d_j be a document and k_i an index term. Then, the binary index term weights, i.e. $w_{i,j} \in \{0,1\}$, of the Boolean model are defined as:

$$w_{i,j} = \begin{cases} 1 & \text{if } k_i \in d_j \\ 0 & \text{otherwise} \end{cases}$$

The similarity function of the Boolean model is also binary. If $\mathrm{sim}(d_j,q)=1$ then the document is considered relevant, i.e. it fulfills the query's conditions (denoted as: the document satisfies the query). Otherwise, i.e. $\mathrm{sim}(d_j,q)=0$, the document is considered irrelevant.

Definition 2-4: Similarity function of the Boolean model

$$\mathrm{sim}(d_j,q) = \begin{cases} 1 & \text{if } d_j \text{ satisfies } q \\ 0 & \text{otherwise} \end{cases}$$

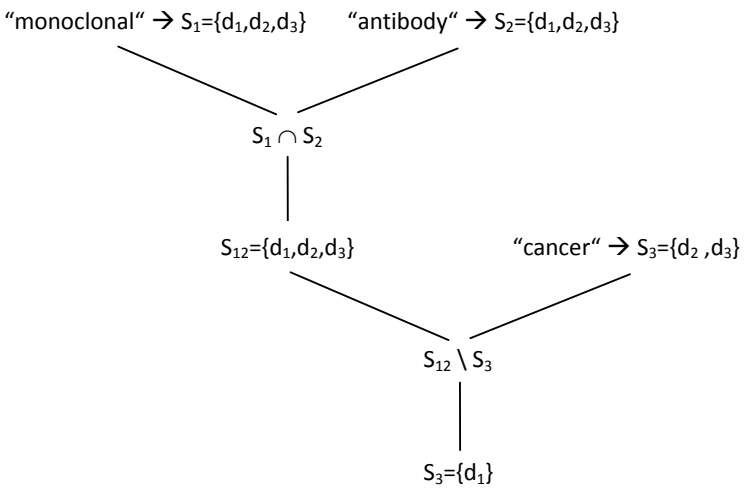
In order to illustrate the Boolean model, we define a sample corpus in Table 2-1.

Table 2-1: Sample corpus consisting of three different documents. The document index terms are the result of stop-word removal and Porter's stemming algorithm.

Document	Text content	Index terms
d_1	Rare side effects of Monoclonal Antibody therapy: Heart problems and skin problems.	rare, side, effect, monoclon, antibodi, therapi, heart, problem, skin, problem
d_2	Monoclonal Antibody drugs are used for the treatment of breast cancer, colon cancer, and lung cancer.	monoclon, antibodi, drug, us, treatment, breast, cancer, colon, cancer, lung, cancer
d_3	A monoclonal antibody is a laboratory-produced molecule that attaches to specific defects in a cancer cell.	monoclon, antibodi, laboratori, produc, molecul, attach, specif, defect, cancer, cell

Assume that the Boolean query is q="monoclonal $\wedge$ antibody $\wedge \neg$cancer". First, for each query term the matching documents are calculated. Then, the document sets are merged using the given connectives, yielding the final answer ser. In the example, the result is $\{d_1\}$. The query processing is illustrated in Fig. 2-2.

Fig. 2-2: Query processing.



Query processing in the Boolean model is usually optimized in order to improve runtime performance [Sormunen 2000]. Optimization means to determine the optimal query processing order. Algorithms for query optimization consider factors like set size or occurrences of a query term within a document. It is thus similar to the optimization of SQL queries in database systems.

Though the Boolean model is simple and easy to use, it provides neither partial matching nor a ranking of results. Partial matching allows a document to match a query even though not all query terms are covered, i.e. present in the document. In the introduced document corpus for instance, d_2 does only partially match the query "cancer AND cell" and thus, would not be in the answer set if the Boolean model is used. Another restriction of the Boolean model is that all terms are considered equally important, i.e. a Boolean operator has much more influence than a critical word. Nevertheless, the Boolean model was for three decades the primary retrieval model in commercial tools and many professional searchers are still relying on the Boolean model. Indeed, the ability to provide exact matches is popular among librarians. Because ranking by relevance is not possible, other criteria for ranking results are used, like sorting by alphabet, by categories, by chronology or by date. The latter approach is used e.g. by *PubMed*, where the newest publications are listed first.

The next model to be introduced compensates for the major drawbacks of the Boolean model. It provides a relevance ranking and partial matching.

2.2.2 Vector space model

The vector space model uses vectors to represent documents and queries [Baeza-Yates & Ribeiro-Neto 1999; Salton et al. 1975]. Each document d_j consists of a T-dimensional vector, where each entry contains the weight $w_{i,j}$ of the corresponding index term k_i (Definition 2-2). The model creates a space in which the document vectors are represented (Fig. 2-3). The dimension of the space is determined by the fixed collection of documents d_j and index terms k_i, i.e. it consists of T rows (the

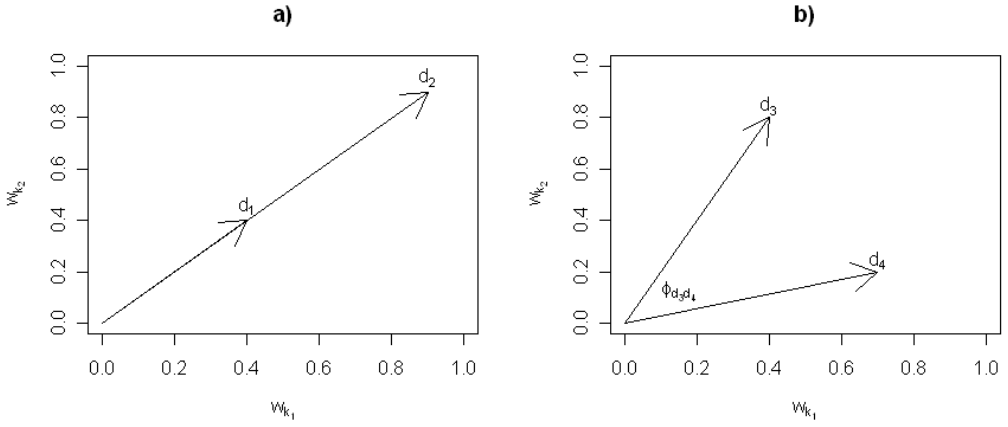
collection of all index terms) and N columns (the collection of all documents). Because the documents and terms are represented as vectors, linear algebra can be used for calculation purposes.

Fig. 2-3: A vector space matrix consisting of
document vectors (vertical) and term vectors (horizontal).

	$\mathbf{d_1}$	$\mathbf{d_2}$	…	$\mathbf{d_N}$
$\mathbf{k_1}$	w_{11}	w_{12}	…	w_{1N}
$\mathbf{k_2}$	w_{21}	w_{22}	…	w_{2N}
…	…	…	…	…
$\mathbf{k_T}$	w_{T1}	w_{T2}	…	w_{TN}

Consider for instance Fig. 2-4, which illustrates two 2-dimensional vector spaces and assume that the index term weights reflect the number of times a term occurs in a document. The vector spaces a) and b) consist of the same base vectors, created by the index terms k_1 and k_2. The displayed documents d_1,…,d_4 contain a different distribution of the index terms, i.e. their document vectors differ. In picture a) the document vectors d_1 and d_2 have different lengths and point in the same direction, while in picture b) the vectors d_3 and d_4 point in different directions. The vectors displayed in case a) could result if d_1 contains the terms k_1 and k_2 only once, and if d_2 contains each term twice. In case of b) the relative proportion of the index term weights is different between vector d_3 and d_4. For instance, vector d_3 might contain term k_1 once and term k_2 twice, while vector d_4 contains term k_1 twice and term k_2 once.

Fig. 2-4: Illustration of two 2-dimensional vector spaces.



Given the vectors of two documents, a similarity between the vectors can be computed, which reflects the degree of similarity in the corresponding terms and term weights. The used similarity measure is usually the inner product (scalar product) of the two vectors or a function of the angle (e.g. cosine) between the corresponding vector pairs. In case of picture a) the angle between the two vectors is zero degrees and thus they would be considered identical even though one document is a multiple of the other with respect to index term occurrence. In picture b) the relative proportion of the index term weights is different and thus the angle is non-zero.

The vector similarity is the normalized scalar product between a query vector and a document vector. Therefore, the similarity value varies in [0, 1]. A value of $\mathrm{sim}(d,q)=1$ corresponds to maximal similarity, i.e. $d=q$, while a similarity value of 0 means that d and q share no terms. Results of a query are ranked by decreasing similarity values, i.e. relevancy.

Definition 2-5: Vector similarity
Let q be a query having the vector $q=(w_{1,q},\ldots,w_{t,q})$. Further, let $d_j=(w_{1,j},\ldots,w_{t,j})$ be the vector of a document. Then, the similarity between the two vectors is computed by the cosine of the angle:

$$\mathrm{sim}(d_j,q)=\cos(\theta)=\frac{d_j \bullet q}{|d_j|\times|q|}=\frac{\sum_{i=1}^{t} w_{i,j}\times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2}\times\sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

Having defined the similarity function the next important step is to introduce a better weighting schema than the binary weighting. The following section discusses term weighting in detail.

2.2.3 Term weighting

Term weights should be continuous and they should weight a term the higher the more important it is for a document. Defining term importance has been elaborately described in literature. A good overview can be found in [Zobel & Moffat 1998]. Three main factors are usually incorporated into the final term weighting formula: Term frequency, document frequency, and document length. Term frequency incorporates the idea that a term describes a document well if it occurs often within the document's text. Document frequency on the other hand says that a term is important if it is not too common across the document collection. Normalization by the document length compensates for the tendency to score longer documents higher. Next, we briefly discuss approaches for calculating term frequency, document frequency, and document length normalization (Table 2-2).

A natural method for calculating *term frequency* (tf) is to simply count the occurrences of an index term t in a given document d. The more often a term occurs in a document, the more relevant the term is for the document. The downside of this approach is that high-frequency terms do significantly bias relevance.

A common approach to deal with this issue is *normalization*: Normalization can be done directly in the term frequency component (cf. Table 2-2 "augmented", "log average") or on the whole document vector (cf. Table 2-2 "cosine", "pivoted unique", "byte size") after the weights have been calculated.

In case of "augmented" normalization, the term frequency is normalized by the most frequent term to a range between 0.5 and 1. Because this definition is based solely on one term, namely the maximal term, it might bias the weights of all other terms of the document vector. In particular, if two documents d_1 and d_2 are identical except that d_1's and d_2's maximum frequent term occurs more often in d_1 due to

some stylistic reasons, then the term frequencies of d_1 would be lower compared to d_2.

Alternatively, the influence of high-frequency terms can be reduced by taking the "logarithm" of the term's occurrence within a document. The logarithm dampens the effect of high frequency terms, yielding a similar effect as normalization.

Another option is the "log average" definition, which normalizes the logarithm of the term frequency by the logarithm of the average term frequency. In effect, a term frequency of 1 would reflect average importance, a frequency above 1 would reflect that the term is quite important for the document and an average below 1 would reflect that the term is not so important for the document. The benefit of this approach over the augmented method is that normalization is not dependent on one term but on all terms. Hence, outliers (in sense of term frequency) do not impact weighting so extremely.

In case of the "cosine" normalization, the weights are divided by the Euclidean length of the document vector. The distortion of weights by terms having an unusually high frequency is thus reduced. In addition, the effect of a high frequency term may be further reduced by combining term frequency with document frequency (cf. below). A disadvantage of the cosine normalization is that it tends to favor shorter documents. Consider for instance a document d_1 which covers topic A. Further, consider a document d_2 which covers topic A and several other topics – d_2 thus covers all the terms from d_1 and several other terms related to the other topics. In effect, the weights of the terms relevant to topic A will be dragged down in d_2 by the weights of the other topics' terms. Hence, a query asking for topic A will be more likely to receive d_1 than d_2.

The issue of cosine normalization was investigated more closely in [Singhal et al. 1996]. The authors plotted the probability of retrieval and the probability of relevance vs. the document length. Their results supported the initial assumption that the "probability of retrieval" is higher than the "probability of relevance" for shorter documents. Based on their findings they introduced the "pivoted unique" normalization which compensates for this error by introducing a correction factor which adjusts the original normalization.

Last, normalization could be based on the number of characters in the document, or we could introduce a bias towards a specific document length.

The *document frequency* within a collection could be ignored so that the original term frequency is not modified. However, using the document frequency is often advantageous in terms of retrieval precision (Chapter 2.7).

The inverse document frequency (idf) is not about relevancy but about the discriminatory power of a term in context of a document collection. Terms which are frequently used in a document collection have a weak discriminatory power, while terms which are rarely mentioned in a collection have a strong discriminative power.

Interestingly, the discriminative power ignores the relevancy of a term, i.e. the term frequency. The idf is defined as the logarithm of the inverse fraction of documents containing a given term. Taking the logarithm of the idf and thus rescaling it is a matter of taste. Nonetheless, idf is often logarithmized because logs are additive and thus, agree well with document scoring functions which tend to be additive. A discussion of theoretical arguments for idf can be found in [Robertson 2004].

Given the codes from Table 2-2 one can define weighting schemas for document vectors and query vectors using a triple code notation. For instance, a common weighting schema yielding high retrieval effectiveness is "lnc-ltc" [Lee 1995]. The schema specifies that the document's vector weights should be calculated using the logarithm term frequency "l", no document frequency "n", and that cosine normalization "c" should be applied. The weights of the query vector are calculated almost the same except that the document frequency "t" is considered, i.e. term frequencies are multiplied with the inverse document frequency. Notice that the factor "t" is usually not calculated for the document vectors but for the query vectors. Indeed, for the cosine similarity measure it does not matter if the idf is calculated for the document vectors or for the query vectors. Hence, omitting the idf calculation for the document vectors is computationally beneficial as we do not have to re-calculate all term weights of all document vectors in case the document index is modified (e.g. by adding a new document to the index).

Table 2-2: Overview of term weighting components. Let N be the number of documents in the collection, and let T be the number of terms in the collection. Further, let t be a term, let d be a document, let w_i be the weight of term i, and let $occ_{t,d}$ refer to the number of times term t occurs in document d. Further, let df_t be the number of documents in which term t occurs. Then, the term weighting components can be defined as follows.

Term frequency	Document frequency	Normalization
n (natural) $occ_{t,d}$	n (no) 1	n (none) 1
l (logarithm) $1+\log(occ_{t,d})$	t (idf) $\log\frac{N}{df_t}$	c (cosine) $\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_T^2}}$
a (augmented) $0.5+\frac{0.5\times occ_{t,d}}{\max_t(occ_{t,d})}$	p (probability idf) $\max\left\{0,\log\frac{N-df_t}{df_t}\right\}$	u (pivoted unique) $\frac{1}{(\text{slope}\times\#\text{of unique terms})+(1-\text{slope})\times\text{pivot}}$
b (Boolean) $\begin{cases}1 & occ_{t,d}>0\\0 & \text{otherwise}\end{cases}$		b (byte size) $\frac{1}{\text{CharLength}^{\alpha}},\ \alpha<1$
L (log average) $\frac{1+\log(occ_{t,d})}{1+\log(avg_{t\in d}(occ_{t,d}))}$		

The subsequent development of weighting schemas yielded two widely used and effective weighting schemas [Singhal 2001]. The first, *Okapi BM-25*, was developed by Robertson and his colleagues [Robertson et al. 2000]. It is based on the probabilistic retrieval model published in the 1970s [Robertson & Jones 1976]. The second is called *pivoted normalization weighting* [Singhal et al. 1999]. A discussion of these state of the art weighting schemas is omitted here, because they are not considered in this thesis. Instead we use the "lnc-ltc" weighting schema. The reason is two-fold. First, the "lnc-ltc" schema is supported by the search engine our work is based on (Chapter 7). Second, and more importantly the "lnc-ltc" schema also gives a reasonable retrieval performance.

Measures based on the tf-idf schema are extraordinarily robust, so that even much more complex models have difficulties to beat it. Therefore, the majority of IR systems are based on the tf-idf measure. Notice however, that the tf-idf phrase is often used in literature to refer to any weighting schema which incorporates the term frequency and the document frequency, regardless of the applied term weighting method, which could be a very simple one like "ntn" or a state of the art weighting schema like BM-25. Nonetheless, the choice of the weighting schema strongly depends on the considered document collection. What works well in one case might not be appropriate for another.

2.2.4 Discussion

The vector space model is superior to the Boolean model as it allows for partial matching and a ranking of results. Best of all, the computational expensive calculations of the term weights can be done offline for the document vectors so that fast ad hoc retrieval is possible.

Nonetheless, the vector space model suffers by two issues: Polysemy and synonymy. In case of polysemy the vector space model can not discriminate between different meanings of the same word, i.e. $\mathrm{sim}_{\mathrm{true}}(d,q) < \mathrm{sim}_{\cos}(d,q)$. In case of synonymy, the vector space model is unable to associate terms, i.e. $\mathrm{sim}_{\mathrm{true}}(d,q) > \mathrm{sim}_{\cos}(d,q)$. The fundamental reason for this is the vector space model's assumption of term independence.

In order to mitigate the problem of synonymy and polysemy the Latent Semantic Indexing (LSI) approach could be applied [Furnas et al. 1988]. In LSI the documents are mapped to a low-dimensional space in which the semantic associations between the terms are reflected. Computation of similarity is conducted in just the same manner as in the Vector Space Model. The LSI yields a higher recall than the VSM and a slightly better precision. However, the LSI is computationally expensive for large collections. Further, it is not required for documents which are well described by a few terms. Rather, it is effective in case documents on the desired topic contain only a subset of these terms. Hence, the benefits of applying LSI depend on the considered document collection.

An increased recall could also be achieved with the vector space model, if query expansion would be applied. Query expansion means that query terms are expanded with their synonyms, e.g. “car” becomes “car automobile”. The disadvantage of this approach is that a thesaurus with synonyms is required and that expanded queries tend to be rather long, potentially increasing processing time.

Regarding the usage of Boolean queries vs. so called “natural language queries” (vector space model), a study conducted by [Turtle 1994] found that professional searchers prefer Boolean queries. The reason is that in the examined use case, experts want to precisely state their question. In the same evaluation [Turtle 1994] also found that despite the usage of Boolean queries by experts, natural language queries perform better on average. Only for special queries does the Boolean model outperform natural language queries.

The Boolean model is often combined with the previously introduced vector space model to get the advantages of both approaches: a filtering of results based on Boolean logic and a relevance ranking based on the vector space model. One of the earliest approaches for achieving this goal is the Extended Boolean model which was proposed in [Salton et al. 1983]. The framework provided by Salton et al. interprets Boolean operations in terms of algebraic distances. Even though the extended Boolean model has been introduced more than 20 years ago it is barely used [Baeza-Yates & Ribeiro-Neto 1999]. Instead, search engines combine the models serially: Results are first filtered by Boolean logic and then ranked by the vector space model. This serial approach will also be used in context of this thesis.

2.3 Text processing

In the previous section, the basic principles of document retrieval have been outlined: Documents and queries are represented as vectors with a tf-idf weighting schema and their correspondence is calculated using the vector similarity measure. However, simply using every word of a document as an index term is not a good approach because not all words are equally significant. Terms with a low discriminative power (idf value near 0), i.e. terms which occur with high frequency in any document of the collection, like the words “the”, “or”, “and”, “a”, etc. are not useful index terms. Therefore, a document is processed by one or more analyzers before being indexed. An analyzer is a combination of several text operations like tokenization, stopword removal, and stemming.

Next, the frequently used tokenization, stopword removal, and stemming operations are explained in more detail.

2.3.1 Tokenization

Tokenization is the process of splitting the character stream of a text document into separate tokens. A trivial approach of doing this would be to split the character stream at every space character. However, considering only space characters wouldn’t yield optimal results. Punctuation marks, quotation marks, exclamation marks, quote signs, hyphens, and many other characters must be considered when processing the character stream. According to [Fox 1992], special attention has to be

given to punctuation characters, hyphens, digits and letter case. A punctuation mark might indicate the end of a sentence or it might be an integral part of the word. As an example consider the word “EC3.4.21.69” which refers to a serine protease enzyme. Removing the punctuation marks would put the digits out of context. On the one hand, a query for “EC3.4.21.69” will still return the document as the query is processed by the same tokenizer. On the other hand, a query for the number “21” will result in false positive hits. A common approach for dealing with this issue is to add special rules to the tokenizer. In the example the “EC”-numbers would be treated differently, i.e. they are not tokenized at dot characters. In just the same manner hyphens (e.g. in department names like “PH-RT”) and digits (e.g. in time “10a.m.”) might be an integral part of a word, leading to similar contextual problems if tokenized. The last point mentioned by Fox is the letter case, e.g. “General Motors” vs. “general motors”. Ignoring it might result in the loss of the word’s true semantics. In the example we would loose the knowledge that “General Motors” refers to a company. Nevertheless this problem is in general ignored, i.e. the character stream is either made lower case or upper case.

2.3.2 Stopword removal

In this process, words which have a high frequency across the document corpus are removed, i.e. they are not considered as index terms. High-frequency words like “a”, “the”, and “is” are not good discriminators as they usually occur in almost all documents. Another benefit gained by removing stopwords is the size reduction of the index structure by 40% or more. While removing stopwords has its clear benefits it can as a side effect also reduce recall (i.e. the amount of returned documents considered relevant; cf. Chapter 2.7). Deciding upon which words to include in the stopword list is thus a crucial task. Many different stopword lists exist and the inclusion or exclusion of stopwords is often dependent on the targeted corpus – indeed, in a corpus about logic words like “and”, “or”, and “not” would be considered relevant. A list of general stopwords for the English language can be found e.g. in [Fox 1989].

2.3.3 Stemming and lemmatization

Stemming is the process of removing prefixes and/or suffixes from a word. Consider for instance the words “connect”, “connected”, “connecting”, “connection”, and “connections”. These words have a similar meaning and can thus be conflated into a single term by removing the suffixes “-ed”, “-ing”, “-ion” and “ions”, yielding the stem “connect”. Stemming can thus reduce complexity by reducing the number of indexed terms and hence the size of the index structure. Another advantage is that relevant documents can be found regardless of the used query word variation (like singular, plural, past tense, ...).

Despite its advantages, stemming can also raise new problems. There are cases, where words with a distinct meaning are conflated, i.e. they have the same stem. As an example consider the words “wand” and “wander”, which obviously have different meanings. They are conflated together, receiving “wand” as a stem. Another example are the words “new” (adjective) and “news” (announcement), which are conflated to “new”. Between these two extremes, of similarity and

dissimilarity, there is a continuum of cases where one can argue in favor or against conflating.

The different stemming algorithms [Smirnov 2008] being described in literature focus mostly on suffix removal because most word variations are introduced through suffixes. The most popular suffix removal algorithm is the one developed by [Porter 1980]. It is simple, fast, elegant, and it yields a similar performance as more complex algorithms. An example of the Porter stemmer is given in Table 2-1.

Lemmatization is closely related to stemming. While stemming uses an algorithmic approach based on heuristics, lemmatization is based on vocabularies and morphological analysis of words. Lemmatization returns only the base of a word form as given in the dictionary, namely the lemma. For instance, lemmatizing the word “saw” yields either “see” or “saw” depending on whether the used token was a verb or a noun. In contrast, the heuristics used in stemming algorithms might conflate the word to “s”. Therefore, lemmatization provides a higher quality in terms of retaining a word’s semantic. The improvement comes at the cost of higher implementation efforts as well as a slower runtime of the algorithm.

The usage of a stemming algorithm is not obligatory. In fact due to the reduced costs of storage space and due to the disputes about the benefits of stemming for IR [Frakes 1992] many search engines ignore stemming completely. Of course, if morphological word variations are still to be matched other methods must be applied such as query expansion. Query expansion simply means to add additional terms to the query. Each word is expanded by its variants, achieving a similar effect as stemming. The query “connect” for instance, is expanded to “connect OR connected OR connecting OR connection OR connections”, so that all variants are covered. However, this approach can become expensive in terms of computation time when long queries are processed.

2.4 Search in the World Wide Web

In 1989 Tim Berners-Lee revolutionized information storage, access, and retrieval by inventing the World Wide Web (WWW) [Gillies & Cailliau 2000]. The invention of the WWW had strong influence on the field of information retrieval. IR which was until then the playground of a few specialists suddenly became an ever growing need of millions of users. Everyone was seeking information on the WWW. However, the size of the Web made search a tedious task. Traditional IR techniques (Boolean model, vector space model, and probabilistic model) which performed well on smaller data sets were not capable of delivering a good performance for the WWW’s information jungle. Indeed, in the mid 1990s search in the Web was like looking for a needle in a haystack. In response to this, many people started to collect and bookmark good websites. The bookmarks were sorted by topic and published in the WWW, so that other people could benefit from “topic of interest” catalogues containing high quality links.

In 1998 IR on the Web was revolutionized by the incorporation of link analysis [Langville & Meyer 2006]. The central idea is that a link is viewed as a

recommendation to the page linked to. Hence, link analysis algorithms calculate relevancy, intuitively "popularity", for the pages of the Web. The higher the popularity, the more important a page is. For example, if a page P_A points to a page P_B, then P_B is considered more relevant than if no page would link to it. Further, the relevancy of the link depends on the importance of P_A. If P_A has a low relevancy, i.e. only few pages point to P_A, then the link from P_A to P_B is less relevant. If on the other hand, many pages link to P_A, then the link from P_A to P_B is more relevant.

The first algorithms to use link analysis were PageRank [Brin & Page 1998; Page et al. 1998] and HITS [Kleinberg 1999]. Analyzing the hyperlink structure of the Web improved search precision dramatically. Nowadays any web search engine uses link analysis as a piece of its ranking formula. However, spammers soon realized how PageRank works and how it can be manipulated. As a result the linkage structure of the web changed and it is assumed that link analysis does no longer play the major role it once did in search engines like Google [Najork et al. 2007].

Because link analysis had a strong impact on IR in the Web, we briefly introduce the most popular algorithms next. We start with PageRank, continue with HITS and then give a short comparison between them.

2.4.1 PageRank in a nutshell

Let P_i be a web page. Then, let B_{P_i} be the set of pages linking to P_i. Let $|P_j|$ be the number of outlinks from page P_j. Then, the simplified PageRank is defined as follows.

Definition 2-6: Simplified PageRank

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|}$$

The PageRank of P_i is the sum of the PageRanks of all pages linking to P_i. Because the PageRank values of P_j linking to P_i are unknown, Brin and Page applied an iterative procedure to calculate the PageRank values.

Definition 2-7: Recursive PageRank calculation

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|}$$

$r_k(P_i)$ is the PageRank at iteration k and $r_{k+1}(P_i)$ the PageRank of P_i at step $k+1$. Initially all PageRank values are set to $r_0(P_i)=1/n$, where n is the total number of pages. The iteration is conducted until the PageRank values converge.

Using a matrix notation, the PageRank values of all pages can be stored in a single vector. In order to do so, an $n \times n$ matrix H and a $1 \times n$ row vector π is introduced.

Definition 2-8: Hyperlink matrix H

$$H_{i,j} = \begin{cases} \frac{1}{|P_i|} & \text{if there is a link from } P_i \text{ to } P_j \\ 0 & \text{otherwise} \end{cases}$$

The matrix H is like an adjacency matrix except that its non-zero values are probabilities of following a link to P_j from P_i. Using the matrix H and the row vector $\pi^{(k)T}$ – which is the PageRank vector at the k-th iteration – the simplified PageRank can be written in matrix notation as follows.

Definition 2-9: Simplified PageRank in matrix representation

$$\pi^{(k+1)T} = \pi^{(k)T} H$$

The simplified PageRank definition has two drawbacks. First, convergence is not guaranteed. Second, pages can accumulate a higher rank score at each iteration step (rank sink) leaving no score for the others. In advance to the introduction of the final PageRank algorithm which fixes these issues, the random surfer model is introduced. The random surfer model illustrates the adjustments to the simplified PageRank.

The simplified PageRank corresponds to a surfer who randomly walks on the graph of the Web: The surfer clicks on successive links at random. Doing so he could get stuck in dangling pages (pages with no outlinks) or he could get stuck in a loop of web pages. However, in reality a surfer will neither get stuck in dangling pages nor in a loop of web pages. Instead, he would randomly jump to another page to continue his surfing. Mathematically, this random jump is modeled by replacing the 0^T rows of H with $(1/n)e^T$, where n represents the total number of pages and e^T is the row vector of all 1s. Hence, if a random surfer enters a dangling page he can jump to any page at random.

Definition 2-10: Stochastic adjustment

$$S = H + a\left(\frac{1}{n}e^T\right), \text{where}$$

$$a_i = \begin{cases} 1 & P_i \text{ is a dangling node} \\ 0 & \text{otherwise} \end{cases}$$

In order to guarantee convergence, Page & Brin made a second adjustment yielding the so called Google matrix which is stochastic and primitive (irreducible and aperiodic).

Definition 2-11: Google matrix

$$G = \alpha S + (1-\alpha)\frac{1}{n}ee^T$$

The parameter alpha is a scalar between 0 and 1. It models the probability of the random surfer to follow the hyperlinks vs. making a random jump to another page.

The higher its value, the more fluctuates the PageRank for even small changes. Further, the higher the value the more iterations are needed until convergence. Therefore, Page and Brin suggested setting the parameter alpha to 0.85 which gives a good balance between efficiency and effectiveness. Given the matrix G, the final PageRank formula can be written as follows.

Definition 2-12: PageRank

$$\pi^{(k+1)T} = \pi^{(k)T} G$$

2.4.2 HITS in a nutshell

The previous section briefly outlined how PageRank uses in-links to calculate the relevancy of a webpage. HITS uses like PageRank hyperlinks to calculate popularity scores. The crucial difference to PageRank is that HITS produces two popularity scores for every page [Langville & Meyer 2006]: an authority score and a hub score. An authority is a page with many in-links and a hub is a page with many out-links. Further, authorities and hubs can be given the adjective "good" if the following circular statement holds: A good authority is a page to which good hubs point and a page is a good hub if it points to lots of pages that are good authorities. Another important difference between PageRank and HITS is the computation of the page scores. Computation of PageRank scores is query-independent and can be done offline for a large hyperlink graph. In contrast, HITS scores are query-dependent. Therefore, HITS is computed online for a query-dependent subset of the hyperlink graph. In this respect, the HITS algorithm is computationally more expensive than PageRank.

We continue with a brief description of the HITS algorithm. Let x_i be the authority score and let y_i be the hub score. Further, let E be the set of all directed edges in the Web graph and let $e_{i,j}$ represent the directed edge from node i to node j. Similar to PageRank, the scores are computed iteratively.

Definition 2-13: Authority score and hub score [Kleinberg 1999]

$$x_i^{(k)} = \sum_{j:e_{j,i} \in E} y_j^{(k-1)} \text{ and } y_i^{(k)} = \sum_{j:e_{i,j} \in E} x_j^{(k-1)} \text{ for k=1,2,3,...}$$

The computation of the scores involves two steps. First, a neighborhood graph N is built which is related to the query. Then, the authority and hub scores of each page in N are calculated. The neighborhood graph N contains all Web pages which match the query. In addition, N is expanded by nodes which pointed to from N or pointing into N. It is assumed that this expansion includes related pages and that it resolves the synonym problem [Langville & Meyer 2006]. For instance, if the query term is "car", pages in N might point to pages outside N, which contain the term "automobile". The expansion is usually limited to a maximum number in order to reduce complexity. Once the neighborhood graph is created, the authority and hub scores are computed until convergence.

Because the definition of HITS is stronger than PageRank (HITS considers two scores rather than just one PageRank score), one would expect HITS to deliver a better ranking performance than PageRank. However, a recent study by [Najork et al. 2007] shows that HITS does not perform considerably better than PageRank.

2.4.3 Web search engines

In the Web two types of search engines can be found: General, all purpose search engines (e.g. Google or Yahoo!) covering a broad spectrum of content and specialized search engines (such as Scirus, Bionity, Entrez or ExPASy) which restrict to a small domain of interest. While the general search engines are known by the majority of users, the specialized search tools are only known by domain experts. Considering the fact that today's all purpose search engines like Google or Yahoo! are indexing billions of web pages the question arises why a single all purpose search engine does not suffice?

First of all, the indexes of two distinct search engines are not identical [Bailey et al. 2007; Spink et al. 2006], regardless if "all purpose" search engines targeting the Web or "specialized" search engines targeting a specific domain are compared. Therefore, inquiries which aim to retrieve all relevant data must use more than one search engine in order to achieve the coverage. Despite their smaller index size, specialized search engines usually provide a higher coverage of domain specific information than an "all purpose" search engine. The common pitfall of all purpose search engines is their ignorance of the Deep Web (database driven portals). Hence, huge amounts of domain relevant data are invisible to them. Targeting only domain relevant information has another benefit for specialized search engines because noise (irrelevant information) is filtered. This is particularly useful for disambiguating homonyms. For instance, scientists looking for "REM" are not interested in the rock group but in web pages about sleep behavior. Wrapping it up, specialized search engines are expected to give results of a higher quality [Steele 2001] compared to all purpose search engines.

Nowadays, thousands of specialized search engines (products, yellow pages, literature, patents, news, social networks, etc.) exist. The variety is so high that next, we mostly restrict on popular tools from the health care & life sciences (HCLS) domain.

The publisher Elsevier hosts the search engine *Scirus* (www.scirus.com) which has its focus on scientific information. It covers about 480 million documents and Web pages from a variety of scientific domains like pharmacology, chemistry, medicine, astronomy, mathematics, physics, and so forth. Scirus, which uses the FAST technology (which is now owned by Microsoft; cf. Chapter 2.5.4) to drive its search engine, offers a variety of options to refine search. Most notable is the fact, that Scirus has a wide range of subject areas (SciTopics; www.scitopics.com) which can be used to specify the search domain. Further, search can be refined by specifying a particular author, journal, article, date range, information type (abstracts, articles, books, conferences, patents, etc.), and file format. Last but not least, query keywords can be combined with Boolean operators.

Bionity (www.bionity.com) is an information portal of the Chemie.DE company. The focus of Bionity is set to biotechnology, biosequences, pharmaceuticals, lab equipment and institutions. The information from the different areas is organized similar to the Yellow Pages or to product catalogues. In addition to these, Bionity offers also news articles. All accessible data can be browsed using their pre-defined thesaurus. Alternatively, a user can access the data by using their search tool. Several search refinement options are available: The information type (all, antibodies, products, companies, news …) can be specified; combination of query words can be switched between conjunction and disjunction; query words matching can be exact or partial; and the search target can be switched between full-text, title, and company name.

The *Entrez* cross-database search tool (http://www.ncbi.nlm.nih.gov/Entrez) is one of the most important search engines in the HCLS sector. It is a federator which gives access to many NCBI (National Center for Biotechnology Information) databases such as PubMed (a free digital archive of biomedical and life sciences journal literature), PubChem (a free database of chemical structures of small organic molecules), Entrez Protein, and Entrez Nucleotide. Advanced search options include Boolean operators and the possibility to restrict search to particular database fields (e.g. medical subject headings). The first result page provides a summary of the hits, i.e. each database is listed together with the total number of hits matching the query. Results can be viewed by selecting a database of interest. The search interface for each database is similar. Another noteworthy feature is that past searches are stored. Past searches can be combined to form new ones. Further, notifications can be transmitted via e-mail if updates for saved queries occur.

The *ExPASy* (Expert Protein Analysis System) proteomics server (www.expasy.ch) is a specialized search engine for protein sequences and protein structures. It covers a variety of databases like Swiss-Prot, PROSITE, SWISS-2DPAGE, etc. as well as other cross-referenced databases such as EMBL, OMIM, Medline, etc. The databases are full-text indexed and can be searched by keywords using Boolean operators and wildcards. In addition, search can be restricted to a specific database. More sophisticated refinement options for search are often available in the specialized database page. Despite the fact, that all databases are accessible from one search box, they don't share a common layout. Hence, the visualization varies from simple flat files to full-fledged designs.

Google Book Search is specialized on finding books online – similarly to Amazon. Currently, the Google Books index contains about 7 million books, which are full-text searchable. Depending on the local copyright laws, the books are displayed as full previews or only as short result summaries. Advanced search enables search by authors, journals, title / subject, and ISBN / ISSN. Further, results can be restricted by date and language. Google Books is thus a good alternative to Amazon.

Google Scholar is a multidisciplinary, broad meta-search engine for scientific literature. It covers peer-reviewed papers, theses, books, abstracts and articles. Their

index covers publications from the Web as well as from the Deep Web. In order to access the latter, Google Scholar cooperates with several journals and digital repositories [Jacso 2008]. For instance, they index about 7 million documents from Elsevier – still a small portion considering that Elsevier hosts 480 million documents. Another partially integrated database is PubMed Central. Here, about a quarter of the articles are indexed. Recently, Google Books was also integrated into Google Scholar. The total size of Google Scholar's index is unknown because Google does not provide any information about it. Regarding the search features, Google Scholar offers an advanced search tab which enables search by authors and publisher. Further, results can be restricted by date and by a few broad topics of interests. Despite the advantages, one should not forget that Google Scholar is still in beta, having several bugs and limitations [Jacso 2008]. A very annoying bug is that Boolean search does not work correctly in Google Scholar. For instance, a search for "information" yields 61,900,000 results and a search for "retrieval" returns 1,450,000 hits (submitted on the 21st April 2009). According to the Boolean model, a search for "information OR retrieval" should return the union of the individual results. However, the total number of results does not nearly match the union. In fact, only 7,200,000 hits are returned for "information OR retrieval" – a serious error in context of professional search. Similar effects occur when restricting the date range. Google Scholar also has problems with the classification of author names. Often, weird names are displayed as authors. For instance searching for "Arabidopsis" returns "A. Initiative" as a key author, even though "A. Initiative" is part of a title and not an author's name. A further limitation is that result sets often contain citation records with almost no information about the content. Further, many duplicates occur in the result list because Google Scholar is not able to detect if two citations are the same. Another limitation is that the ranking of results can't be adjusted (e.g. by date, citation, etc.). Last but not least, the first result items often link to journals while the open access versions of the same paper are ranked lower.

The last paragraphs have only touched the tip of the search engine iceberg. In just the same manner, workers of companies are confronted with a large variety of intranet search engines. Here, focused search engines are even more important than in the Web, because employees – especially in R&D – have a high degree of specialization and thus very focused interests. An overview of the usage of specialized search engines in a R&D department of Roche is given in chapter 5.1.

2.4.4 Discussion

The invention of the WWW as well as the development of the search engines which operate on the Web had a strong influence on the way people access information. In the past it was common to visit libraries and pore over books. Often, the required literature was unavailable and had to be ordered, arriving at the library with a delay of several days. Nowadays, most information is available online and retrieval is a matter of a few mouse clicks. In addition, the variety of topics and the number of accessible information sources is extremely large with.

Modern ranking algorithms, like PageRank and HITS, enabled state-of-the-art search engines to achieve a significant improvement in retrieval performance compared to

previous methods. In fact, the average user uses the word "google" as a synonym for "searching the Web for information". This holds also for professionals to some extent because Google offers many convenient services for them: Search for transportation, search for accommodations, search for patents, search in Wikipedia, and search in selected literature databases. Nevertheless, due to the size and heterogeneity of the professional domains, specialized search engines are used by many professional searchers (cf. previous section). Because the indexes do not overlap well, a complete coverage implies several different search engines. Nonetheless, surveys show that professionals tend to use only a few specialized search engines [Miller 2002; Mühlbacher 2008]. In fact, they are often unaware about a search engine's indexed content and do often assume all purpose search engines would cover all relevant content (e.g. Google indexes all Web content).

Professional searchers might thus benefit from a "portal" which knows about their domain of expertise. Such a portal could automatically decide to which sources a query is transmitted and it might even integrate the results into a unifying interface. Further, a ranking of results adjusted to the professionals' context would be convenient. That means, not only ranking by text similarity, date, and citation is interesting for professionals but also by context. Ideally the topics matching the professional's context would be selected automatically.

A problem which primarily affects "all purpose" but barely specialized search engines is spam [Henzinger et al. 2002], i.e. the malicious manipulation of the page ranking. In the 90s, spamming was a piece of cake because traditional IR techniques were not able to cope with spam – they were tuned for neat document collections. It sufficed to create pages with manipulated text content in order to boost their relevance. While the manipulation was often invisible to the user (because the chosen text color matched the background color), the indexers still processed the fake text content. The fake text was placed in such a manner that manipulated pages yielded a high similarity score for many queries. As a result it was common that a lot of spam sites appeared in the first result pages of a query. Then, link analysis algorithms like PageRank and HITS appeared. Soon, spammers realized how these algorithms work and they adjusted their techniques. Now they use sophisticated link farms ("artificial" links between pages) in order to increase their authority score resp. hub score values. The development of anti-spam methods and new spam techniques is like a cat-and-mouse game. For professional search however, we assume that spam is not such a big issue. First, spammers try to target popular queries and not exotic technical terms. Second, professionals often use specialized search engines which are barely affected by spam. Third, professionals search also in intranets, which are in general not affected by spam (cf. next section).

People expect another revolution from the Semantic Web (Chapter 4), which aims at making the Web content "machine understandable" so that complex reasoning is enabled. Even though the idea and Semantic Web standards exist now for over 8 years, the vision of a large scale Semantic Web has not yet become reality. Maybe this is due to the fact that the semantic information has to be provided manually and

in a common meta-model. Despite this, there are already some successful projects using semantic technology, especially in small domains.

2.5 Search in an intranet environment

Huge amounts of data are not exposed to the Internet but are kept behind an organization's firewall, accessible only by its members. Data within such a private network (intranet) can encode business processes, administrative tasks, domain expertise, etc. and is often of special value to the organization. Therefore, being able to find information in private networks is essential as a study conducted by the market analyst IDC shows. Their white paper called "The high costs of not finding information" [Feldman & Sherman 2004] states that information workers spend 15% to 35% of their work time on searching for information. Further, 40% of corporate users complain that they cannot find the information they need to do their work. Arguably the costs caused by time wasted for searching information, reworking information, and missed opportunities can get enormous for a company. Consequently, a company can greatly benefit if its workers can easily discover existing data in their intranet. This applies in particular to large corporations where huge amounts of data are available.

This sub-chapter is dedicated to search in intranet environments – the area our work is focused on, i.e. professional search at the workplace. In this respect, the following sections are crucial to the main part of this thesis. In detail, we will discuss characteristics of intranets, issues of search in intranets, and solutions of leading search engine companies.

2.5.1 Differences between intranet search and Web search

An intranet environment is different in many aspects from the Internet. Even though both are using the TCP/IP protocol for communications, the structure in which data is stored differs a lot. The Internet has the World Wide Web as the main structure for storing and interlinking information. As a consequence, users can access all data on the Web using a single tool, namely their web browser. Hypertext exists in many intranet landscapes. However, hypertext in intranet environments is much less prevalent than in the Internet. In fact hypertext is just a small piece of the landscape, sharing it with file share protocols, database systems, etc. An intranet environment is a collection of all kind of electronic information within a private network. Therefore, search in an intranet environment differs from search in the Internet and the term enterprise search is used instead of web search.

Enterprise search is defined by [Hawking 2004] as search over all electronic text content of an organization, including search of the organization's external websites, search of internal websites, search of other electronic text held by the organization in the form of e-mail, database records, documents on file shares and the like.

The differences between search in an enterprise environment and search in the Internet have been analyzed in [Fagin et al. 2003]. Fagin et al. mention six main differences which affect search: The qualitative linkage structure, the macro level

structure, spam, the search context, the answer set size, and the search engine friendliness. Next, each point is briefly outlined.

Difference 1: Qualitative linkage structure
Comparing solely the hypertext structure of an intranet to the structure of the WWW will reveal significant differences [Fagin et al. 2003; Xue et al. 2003]. While in both cases a collection of interlinked documents is present, the way pages are linked differs.

In the Internet, the linkage-structure is generated democratically. According to [Surowiecki 2004], the "collective solution", i.e. the average guess of a group, outperforms the individual guesses in approximately 98% of cases. Hence, the paradigm of the popularity vote exploited by link-analysis algorithms like PageRank can be applied.

In intranets however, this paradigm fails. Here, links between documents reflect something fundamentally different. First, they are often generated by a few employees with special privileges. Second, documents are created to deliver information and not to draw as much attention as possible. Third, links do not reflect a document's relevancy. Indeed, in a company the manifestation of the linkage structure often corresponds to the organization's departmental structure. Therefore, a link pointing to a document can not be considered as a recommendation.

Consequently, search approaches which work well for the Internet might not deliver the expected ranking performance for the hypertext part of an intranet.

Difference 2: Macro level structure
In [Broder et al. 2000] the macroscopic structure of the Web is described by four pieces. At the heart, the Internet consists of a giant strongly connected component (SCC). Every page in the SCC can reach any other page of the SCC along directed links. The second and third pieces are called IN and OUT. IN contains all pages that can reach the SCC, but no page from the SCC can reach IN pages. OUT contains all pages that cannot reach the SCC, but SCC pages can reach OUT. The last piece is TENDRILS. These are pages which cannot reach the SCC and cannot be reached by the SCC.

In [Fagin et al. 2003], a study of IBMs macro level intranet structure was conducted. While the intranet contains the same pieces (SCC, IN, and OUT) as the Web, the distribution of the sizes differs. Most notably, the SCC of IBMs intranet is much smaller than the SCC of the Web. IBM's intranet had 10% of its pages in the SCC while the Web has 30% of its pages in the SCC. Consequently, if we assume that the hyperlink structure of other intranets has similar differences, we can conclude that PageRank does not work very well in intranets.

Difference 3: Spam
Spam is a well known problem of today's Web. This phenomenon does not exist in an intranet, i.e. intranets can be regarded as spam-free [Fagin et al. 2003]. Workers encounter a high social pressure in delivering good results, i.e. in showing colleagues

their professional capabilities at work. By doing so, they can speed up their career. The incentives are thus high not to create spam in a private network. As a consequence several ranking heuristics (cf. Chapter 2.6; anchor-text, URL depth, etc.) which can't be applied in the Internet are applicable in intranets. As an example, consider the hyperlink's anchor-text heuristic stating that "a page p is more relevant for a query q if the words in q are part of a hyperlink that points to page p". This is a dangerous criterion in the Internet because spammers can easily misuse this to get their pages ranked higher. In intranets however this danger is not an issue.

Difference 4: Search context
Another difference between search in the Internet and search in a corporate environment concerns the user's context or points of view. In the Internet the only contextual information which is usually available is the location of the searcher and maybe his preferences derived from past actions. In intranets however, the available contextual information is much richer. Departments, projects, and group memberships are just a few examples. Therefore, considering the user's context at query time has a much bigger relevance in intranets (Chapter 5).

Difference 5: Answer set size
Queries in intranets have often only a small set of correct answers. At first glance this seems to make search in intranets easier than in the Web as the number of pages to be ranked is considerably smaller. However, the drawback is the inability to identify the correct answer page. Having only a few or a single answer page means that the searcher has to use notions and terms that match the vocabulary of the document. If he uses synonyms or more general / specific search terms he might not find the desired document. In the Web this problem is mitigated due to the informational equivalence of many different answer pages.

Difference 6: Search engine friendliness
The heterogeneity of platforms and formats makes intranets less search-engine friendly [Fagin et al. 2003]. Database driven portals for example generate multiple views based on the database content. Often, the views contain temporarily generated hyperlinks to other layout styles, or to actions such as deletion, insertion, and modification. In case a crawler would follow these kinds of links it could lead to unexpected behavior such as modification of database entries.

Search engine friendliness is also an issue in the Internet. Otherwise, there would not be so much active research in the area of Semantic Technologies (Chapter 4) which aims at making data machine-interpretable. At least, in the Internet it suffices to support a few formats so that large parts of the content are covered. This is in contrast to intranets. In intranets, many different search engine friendly interfaces – targeting many different formats – are required in order to achieve an acceptable coverage of the content.

2.5.2 Open issues in intranet search

In this section, we outline open issues encountered in intranet search. In [Hawking 2004] several open problems are mentioned such as defining a test collection,

effective ranking, search in e-mail and media files, and exploiting the search context. Next, we briefly describe the most relevant ones.

Issue 1: Defining an enterprise search test collection
Having an appropriate test collection for performance evaluations is without any doubt very important and can be considered a must have for the investigation of the other problems. An enterprise test suite should contain a realistic mix of different data types like websites, file shares, xml, etc. Creating such a test collection for different enterprises, i.e. different expert environments, is a difficult task.

Issue 2: Effective ranking over heterogeneous enterprise collections
Results from distinct sources (web pages, e-mails, database entries, file shares, desktop, etc.) are ideally combined into a single list using an appropriate ranking formula. However, due to the heterogeneity of the file types (structured entries vs. unstructured entries or linked entries vs. non linked entries), the used languages, and the targeted domain, this is a difficult task. Indeed, company employees could benefit from a search tool which includes "all information sources", i.e. any internal and external sources a user can access. However, building such a portal becomes a difficult task due to the ranking issue.

Issue 3: Estimating relevancy for documents which are not part of a web
Many sources (e.g. file shares, e-mails and databases) are missing key evidence factors for document relevancy (Chapter 2.6) which were successfully applied in Web search. For instance, link analysis or anchor text evidence can't be applied. Therefore, finding other evidence as a support to text similarity is an open and important problem.

Issue 4: Exploiting search context within enterprise searches
Search could be greatly improved if the search context was known. Restaurant search e.g. can benefit from location based context. Similarly, other queries could be expanded with context if rich user profiles were supplied. The main issue here is how to extract context information and how to aggregate the various contextual aspects. In this thesis we will analyze how context can be extracted and incorporated into the search process (Chapter 6).

Issue 5: Providing effective search over continuous media
Media collections ranging from audio & video to time-series data (e.g. mass spectra) become more and more important in enterprises. It is assumed that in future a lot of collections will become interlinked and accessible by means of a web browser. At least by then, the ability to search media collections will become very important. Hence, search techniques for continuous media must be developed.

The previous list is of course not complete and many other problems exist. Another issue which is particularly important in context of companies is security. In a corporate environment strong security settings are applied to data repositories like file shares, databases, applications, or web portals. The problems caused by security settings are twofold. On the one hand, a search engine must natively support the

security settings of the indexed data in order to be deployed. On the other hand, a user won't find the information he is looking for if the security access rules forbid it. Therefore, security is to some extent comparable with the Deep Web problem of search on the Web.

As mentioned in the second issue, enterprise collections are heterogeneous, i.e. search in databases or xml is also relevant. The next section describes briefly how search in such structured sources differs from search in unstructured sources.

2.5.3 Search in structured sources

So far, this thesis used the term "search" as a notion for search in unstructured sources, i.e. search in free-form natural language text records. As a matter of fact, we focus our work particularly on unstructured records because the need of finding information in these is particularly large (Chapter 5). Nonetheless, structured sources are a very important source of knowledge within corporate environments. Indeed, professional search is also about querying structured sources (databases), which often hold huge quantities of relevant data.

In structured sources information is stored according to a given schema. In case of relational databases [Codd 1970] the "reality" is modeled in tables. Tables have a defined set of columns with a defined data type and they can have relations to other tables. Databases have thus precise "semantics" while text documents are often incomplete or ambiguous. For instance, databases are using common data types like person name (a string value) or telephone number (a numeric value). In a free-form natural language text however, there is no structural information which explicitly tells us that a text snippet represents a person's name or a telephone number.

A key difference between structured sources and unstructured sources is the way information or data is accessed. In case of databases the user's "information need" has to be formulated precisely, i.e. which tables and columns should be queried, how should the data be joined, etc. The answers produced are therefore exact matches to the query, and they lack a ranking by relevance. This and other differences between Data Retrieval and Information Retrieval are listed in Table 2-3.

Table 2-3: Data Retrieval vs. Information Retrieval after [Van Rijsbergen 1979]

	Data Retrieval	**Information Retrieval**
Matching	Exact match	Partial match, best match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic
Classification	Monothetic	Polythetic
Query language	Artificial	Natural
Query specification	Complete	Incomplete
Items wanted	Matching	Relevant
Error response	Sensitive	Insensitive

Despite the fact that data retrieval enables complex queries, an IR style for accessing structured sources is often desirable. The latter enables fast querying without the need to know the database's schema. An issue of this approach is how to display a summary of a matched data entry. The value of a single table entry has in general no

meaning. It is the context, i.e. the data type and the relationships to other entities which give meaning to the single entry. As an example consider a project database having the two tables projects(name, id) and milestones(project_id, description, date). If someone searches for “project kick-off” he might find a hit in the “description” column of the milestones table. Nevertheless, without the context, he neither knows which project is meant nor the date of the kick-off. Therefore, an answer must contain all relevant relationships in order to be useful for a searcher. Particularly with regard to enterprise applications, like customer relationship management which embed large quantities of business logic, search results must be displayed in context.

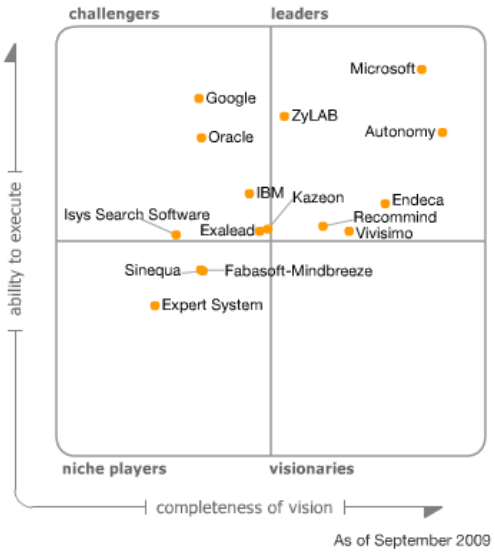
2.5.4 Enterprise search engines

There is a broad spectrum of enterprise search engines available on the market, varying from low end solutions to high end, i.e. industrial strength, solutions. In this section we will focus on the latter: major vendors offering high end products, which go beyond simple search. Their capabilities include amongst others federated search, text categorization and clustering, fact and entity extraction, taxonomy creation and management, information presentation, and personalization services.

Evaluating and comparing the search products on the market is a difficult task. First, scientific investigations of the vendors’ search solutions are barely available. Second, we can not afford (due to limited resources) to evaluate each product. Therefore, we use the product sheets available on the vendors’ website and a report conducted by the Gartner group [Andrews 2009] as the primary source of information.

Gartner analyzed vendors offering mid to high-end search tools. They created a so called “Magic Quadrant for Information Access Technology” (Fig. 2-5) into which the vendors have been classified using a custom evaluation criteria [Andrews 2009]. The quadrants are: leaders, challengers, niche players, and visionaries.

Fig. 2-5: Gartner’s magic quadrant of information access



Leaders are characterized as having a high architectural flexibility, providing developers with the tools to tune relevancy settings and customize their solutions, and they have enough depth to serve as platform vendors such that their software can be used to solve most information access problems.

The *challengers'* quadrant is characterized by vendors who have enough resources to penetrate the information access technology market effectively. In contrast to the leaders quadrant however, they lack the vision to address all information access opportunities. Further, enterprise search is not their core source of revenue. Nevertheless, Gartner says that any of the challengers could emerge as leaders within 24 months if they invest the necessary resources.

Visionaries are characterized by having insightful approaches and flexible architectures. However they lack financial resources to provide future strength. No vendor is listed in this quadrant for 2009.

Niche players have specialized on a certain issue and lack the breadth to satisfy a wide variety of projects.

We are not going to describe each vendor of the "Magic Quadrant". Rather we restrict to the search solutions offered by Microsoft (which acquired FAST in 2008), IBM, Autonomy (which acquired Verity in 2005), Endeca, Vivisimo, Google, Fabasoft/Mindbreeze, and Recommind. The focus is thus set on the leader's quadrant. In addition, the open source search engine Lucene will be discussed, because it is a highly customizable search tool for mid to high-end solutions.

Next, we will give a comparison of the strength and weakness of the mentioned search solutions.

Microsoft is a global software company having presence in most countries. Its FAST Search product can give recommendations (content-based and collaborative-based filtering; cf. Chapter 3.2) about unknown content to users. Hence, they expose the "long tail" [Anderson 2006] (less popular) content to the user. Success of those recommendations is stored for future reference. Personalization can establish multiple profiles for individuals based on time of day or location. Technically, they demonstrated the ability to handle extremely large data sets and heavy traffic, including installations with "spikes" of over 1,000 queries per second and multi-terabyte corpora. FAST can index a broad variety of content sources.

IBM has a broad spectrum of products for information access including a free product developed with Yahoo, a basic enterprise edition, an edition with significantly advanced capabilities for deployment in vertical sectors, and an offering to fuse search and business intelligence (IBM Content Analyzer). IBM has sales and support presence in many world regions and a particular facility with non-textual multimedia and content analytics. They are also investing efforts into "social search", i.e. determining relevance by considering user interactions and contributions. IBM

does not exploit users' search history or explicit status in an organization as means of determining the relevancy of results.

Autonomy has the ability to index a broad spectrum of content. They are also capable of providing large scale installations exceeding a petabyte of data. Further, they also provide deep user profiling, including users' search histories, explicit roles, behavior in day-to-day authoring, and interactions with external and internal information sources. The downsides of using Autonomy are their high prices compared to other vendors and the lack of support.

Endeca's connector family gives it a particularly broad capability to index from a variety of content sources. These include manufacturing applications to which few, if any, other vendors connect. Additionally, Endeca has a unique search federation connector strategy that extends its reach through enterprises in an innovative fashion. Endeca records user searches, navigational behavior and the intersection of the two, providing a unique perspective on users' past choices and unusually rich user profiles. This information can be combined with explicit profile data from enterprise-established permissions and privilege lists. Endeca has a particular facility with action-oriented content analytics, such as analysis models covering a broad spectrum of user behaviors and textual analytics (including concept extraction). Its recommendation and document summarization engines are claimed to be especially effective.

Vivisimo puts its clustering capabilities into foreground. Clustering enables search results to be put into topical categories. A significant number of index connectors allows for a variety of content sources. Vivisimo is also stressing their capabilities in "social search". They have a simple pricing model based on corpus volume or document count which makes it easy to predict costs. Compared with the other Leaders, Vivisimo's financial resources are fairly limited.

Google Search Appliance has a strong brand due to its popularity in Web search. In enterprise search they offer support for over 200 file formats and they are able to handle large scale installations. Another capability is the adjustment of the document part relevance. One could e.g. specify whether the abstract is considered more relevant than the rest of the document. In the latest version they have also added support for role-based personalization, in which administrators can adjust the ranking of results for certain user groups. However, they have no investment in exploiting users' historical behavior. While Google Search Appliance offers almost no adjustment to the local environment, its biggest strength is that it is a ready to use, out-of-the-box search engine. Further, customers benefit from a simple pricing model.

Fabasoft/Mindbreeze offer strong content analytics and federation capabilities with connectors for a variety of content sources. The system supports different relevancy models which are applied to specific types of queries. Further, the system offers "guided navigation", a term shaped by Endeca. Guided navigation means that search results are grouped e.g. by time, type, origin, etc. Hence, an overview of the result

set is generated by the system. They have a simple pricing model based on user count. It provides a predictable initial cost base. Fabasoft/Mindbreeze offers only limited user profiling.

Recommind is unusually good at using statistical analysis to extract meaning from documents. They have various categorization features where documents can be categorized into taxonomies. They are also able to generate a taxonomy from existing content.

Lucene, an open source project of the Apache foundation, is not listed in Gartner's magic quadrant because they are no commercial entity. Lucene at its core includes an API for index and search functionality. There is a large user community which actively and continuously develops Lucene, extensions, and other open source products on top of it (e.g. Solr). Due to its popularity many developers know the project and have specialized on customizing and implementing Lucene-based solutions in companies. In contrast to proprietary search solutions, a company using Lucene is not dependent of a vendor and its price policy. That is why Lucene is a cost-effective option for companies which want to build their custom made search solution from scratch. It is expected that in future more companies will build their search tools on top of Lucene.

This section has introduced several high-end search engines for the enterprise. According to the vendors' product brochures, a rich repertoire of functionality is offered by their respective search engines so that finding information should be a piece of cake. Interestingly, studies like IDC's "The High Cost of Not Finding Information" as well as several in-house studies conducted at Roche (Chapter 5) disagree. In fact, they point out that search engines still do not deliver satisfying results to professionals.

Maybe this is because existing tools do only partially cover the working hypotheses defined in Chapter 1.2. Hence, essential quality characteristics of a professional search tool might be implemented insufficiently or not at all.

The first hypothesis ("Professional searchers prefer to access all sources via one single entry point") is considered by most industrial strength search tools and usually implemented using a federated search approach. The difference is in the degree of pre-defined connectors.

The second hypothesis ("Professional searchers require a role-specific ranking of results") on the other hand is considered only by few vendors. Personalization services based on some primitive form of user profiling is offered for instance by Google Search Appliance and Endeca. Others such as Autonomy claim to offer a much richer and deeper profiling.

The third hypothesis ("Guided navigation is relevant for professional search") is also only considered by some vendors. Interestingly, the approach by which guided navigation is implemented differs. Some rely on clustering (e.g. Vivisimo) which

usually delivers ad-hoc categories, others rely on text categorization, and yet others rely on pre-defined metadata (e.g. FAST by Microsoft).

The fourth and last hypothesis ("A company's knowledge is instrumental for professional search") is usually ignored by commercial solutions.

Wrapping it up, there is not one vendor which covers all aspects. Most products fail due to a support for role-specific ranking of search results, information extraction (for guided navigation), and most importantly exploitation of a priori knowledge. However, the biggest issue is that even though some aspects are covered, no information is provided about their effectiveness. A key question which thus remains unanswered is "which principles improve the effectiveness of finding information and which do not".

Answering this question and determining which principles work is a key contribution of this thesis. Because the commercial entities lack customization to the specific needs of a professional search environment, we decided to conduct our research based upon Lucene, which offers so much flexibility and extensibility (Chapter 7.1.1).

2.6 Evidence for document relevance

The relevance of a document can be based on various sources of evidence. So far, we already introduced the well-established tf-idf metric, which is based on text similarity, and additionally metrics based on link analysis such as PageRank. This section completes the previous metrics as it gives a brief introduction to several other sources of evidence.

Scoring metrics, i.e. evidence for document relevancy, can be classified into two groups: *query dependent* evidence and *query independent* evidence. Query dependent evidence means that the relevancy of documents depends on the query. The tf-idf metric e.g. falls into this category. Query independent evidence is a static value which is associated with a document. Having such a static value associated with each document, enables to calculate a document's relevancy without providing a query. The PageRank value e.g. is a query independent metric. Similarly, a metric which scores recent documents higher than older ones, is a query independent metric, because it relies on the document's last modification date to calculate relevancy – a static value which is query independent.

In theory, any two scoring schemas could be combined. In practice however, this is not always the case. Combining for instance two contradictory metrics, like one scoring new documents high with one scoring old documents high, will result in no useful ranking.

In [Craswell et al. 2005] three methods for combining query independent evidence with a query dependent baseline are discriminated: *rank-based*, *language modeling prior*, and *relevance score adjustment*. In the first approach, the baseline score and the query independent scores are transformed into two rankings. The merged score is computed based solely on the items' position, i.e. on the order of the items. A

benefit of this approach is that power law scores can be combined with linear scores without introducing a potential bias towards a few pages having a very large score and the majority of pages having no score. This method could be e.g. applied to combine the PageRank's power law distributed scores with the tf-idf scores. In the second approach, a prior is calculated for each item which is then combined with the language modeling probability by multiplication. The last approach uses a linear combination of the baseline score and the static score to calculate the ranking of results:

$$\text{Score} = a_0 \times \text{SimBaseline} + \sum_{i=1}^{n} a_i \times \text{SimMetric}_i \text{, where } \sum_j a_j = 1 .$$

The a_i factors control the influence of each metric on the final score. In our case evidence is combined using the relevance score adjustment method (Chapter 6.5).

Next, we give an overview of query (in)dependent evidence factors (Table 2-4). The list is based on the factors published in [Fagin et al. 2003; Westerveld et al. 2002] and displays the most common evidence factors. While there are many sources of evidence, the difficult task is to choose amongst these and combine them in a proper way. Next, we discuss each type of evidence and its applicability in intranets.

Table 2-4: Query (in)dependent evidence factors.
+: applicable, -: not applicable, o: applicable if the institution's privacy policy allows the usage of personal data

Source	Evidence	Applicable in intranets
Content	Full text	+
	Meta fields	+
	Structural information	+
	Anchor text	+
Context	IP address	o
	Cookies	o
	Session	o
	Location	o
	User	o
	Security	o
Time	Date of creation	+
	Last modification	+
	Last access	+
Link information	PageRank	-
	InDegree	-
	Citations	-
URL information	URL length	+
	URL depth	+
Implicit feedback	Click-through data	+
	Time	+
Explicit feedback	Rating	o
	Tagging	o
	Semantics	o
	Preferences	o

2.6.1 Content evidence

Content evidence covers any information which can be gathered from the indexed object, like full text, meta fields (e.g. author, subject, etc.), structural information

(e.g. headings, abstracts, etc.), and anchor text. The full-text content of objects is a de-facto standard source for relevance ranking, used by almost all search engines.

Data provided in meta fields is often considered to be of higher quality than data in full text content. If for instance the query word "NASA" is found once in the "title" meta field of a document d_1 and the same word is also found once in the full text content of a document d_2, then d_1 is considered more relevant than d_2.

Similar to meta fields, text within certain structural parts of documents is considered more relevant. A document's headings for instance, give a good summary of the chapter's content. Therefore, if a word is matched in both, a heading and in a regular paragraph, the match in the heading should be considered more relevant. However, this principle can only be applied if the pre-defined headings are used. In case custom formats (like a 14 point italic font) are applied, an automatic detection of headings becomes very difficult. In the worst case, relevance could thus only be adjusted based on coarse structural information such as font size and font emphasis (bold, italic, and underlined).

Links occurring in a document could also be described as part of structural evidence. However, we decided to separate links from regular structural text properties as they are only common within HTML documents. The label of a link, in the following referred to as anchor text, gives valuable information about the target's content and should thus be considered as evidence when ranking results [Craswell et al. 2001; Eiron & McCurley 2003; Fagin et al. 2003; Fujii 2008; Westerveld et al. 2002]. In other words: An anchor's text is like a tag, giving a good summary of the target's content. In the Internet this assumption might be dangerous due to spammers. Though, this is not the case for intranets because these are considered spam free. Nonetheless, the applicability of such methods is low in intranets due to the low availability of links within regular office documents.

2.6.2 Context evidence

Context evidence such as IP address, cookies, session, location and user id are often used by adaptive systems to adjust the ranking of results. Common examples encountered in Web search engines are so called location based services. People who search at Google for "Greek restaurants" for instance are shown restaurants from their area. The location is automatically deduced by mapping the client's IP address to the geological position. Arguably, this assumption makes sense in most of the cases.

In contrast to the Web, the usage of context evidence in a company depends highly on their privacy policy. The problem is that most systems relying on this kind of evidence need to some extent user data in order to work (Chapter 3).

2.6.3 Time evidence

Timestamps are an important part of many information objects. Typically, the relation between a timestamp and an information objects reflects an event which

occurred at the specified time. Common events are creation, modification, and deletion of objects.

News, blogs, e-mails, discussion groups, and document management systems are just a few areas, where timestamps give us information about the time an item was created, modified or deleted. Depending on the domain, time has an important role when searching for information. In case of news, scientific literature and the like, the majority of people are mostly interested in the latest articles. For instance, a searcher might only be interested in the latest news about "NASA". This knowledge is often incorporated in a search engine's ranking heuristic, i.e. the text similarity score is combined with a recency score. While this assumption might be valid for the majority, it ignores people who are looking for older news. In the running example, a searcher might not be interested in the current missions of NASA but in the first space missions.

A better approach would be to consider the distribution of relevant documents over time (such as the date a news article was first published). According to [Li & Croft 2003] there are three types of queries. The first query type has a uniform distribution of relevant documents over time (e.g. elections in the U.S.). The second query type favors very recent documents (e.g. the current oil crisis) and the third query type has most relevant documents within a specific period in the past (e.g. World War II). Based on these query types, the authors of [Li & Croft 2003] developed a query language model and conducted an evaluation versus TREC[1] ad-hoc title queries. Their method outperforms the baseline models. In particular, it outperforms the linear combination (text similarity + recency) method used by most commercial search engines. However, a major drawback of this approach remains: in order to be useful in large scale, queries must be automatically categorized into time-based queries.

2.6.4 Hyperlink evidence

Link information became one of the most important sources of evidence for ranking results in the WWW. Algorithms such as PageRank and HITS revolutionized the search experience (Chapter 2.4.1). As we discussed in Chapter 2.5.1, this kind of evidence is not very useful in the hypertext part of intranets due to structural differences. Further, other file formats (PDF, Word, Excel, etc.) usually lack a comparable linkage structure. Instead, citation links are often encountered such as bibliography references. In order to extract and interpret such bibliography links we would need to apply text analysis. Because there are barely explicit URLs in non-HTML documents and because extracting citations with Natural Language Processing (NLP) techniques is a difficult task, we do not consider this kind of evidence in this thesis.

2.6.5 URL evidence

A URL contains valuable query-independent information for the ranking of search results [Fagin et al. 2003]. The length of a URL could indicate the authority of a page.

[1] Text REtrieval Conference (TREC); Publishes test collections for various tasks; http://trec.nist.gov/

If two pages have similar content, then the page with the shorter URL has probably a higher authority than the other. Similar, the depth of an URL (the nr. of slash "/" characters) could be related to the authority. Pages located at the top can be considered more general than pages located in deeper hierarchies.

2.6.6 Feedback evidence

Implicit feedback describes evidence which can be gathered through observation of a user's interaction with the search engine. The extracted evidence can be query dependent as well as query independent. A naïve heuristic would be to score a document the higher the more often it was accessed. A more "clever" version would be to regard the accessed documents in context. One option would be to use the query as the context [Joachims 2002; Radlinski & Joachims 2005; Xue et al. 2004]. If a document is often accessed in context of a query, then it is more relevant than others, but only in this context. Another option would be to use the user's context (interest, preferences, location, etc.). If a document is often clicked by searchers with a common context, then this document is more relevant than others for their context.

Closely related to implicit feedback is explicit user feedback. In contrast to the previous approach, explicit feedback is usually of a higher quality but not as abundant as implicit data. Explicit information can be e.g. demographic information, personal interests, research area, project team memberships, rating of items, etc. The data can be used in just the same manner as is done with implicit feedback, i.e. either as query dependent or query independent evidence.

Both types of feedback evidence are introduced in detail in Chapter 3.

2.7 Precision and recall

Precision and recall are widely used performance measures which are also applied in this thesis to determine the performance of various components. Most important for us is their usage in information retrieval and in text categorization. In the former, the measures are used to evaluate the retrieval performance of search engines. In the latter, the measures are used to evaluate the classification performance. A definition of precision and recall is given next.

Definition 2-14: Precision

Let R be the set of relevant documents and let A be the answer set of a query. Precision is defined as the percentage of relevant answers among all answers.

$$\text{Precision} = \frac{|R \cap A|}{|A|}$$

The definition of precision was given using two auxiliary sets R and A, which were defined so that precision in information retrieval is explained. Nonetheless, the formula stays exactly the same if we consider precision in text categorization. We merely replace R by the set of documents which belong to the positive class and A by the set of documents which were labeled to belong to the positive class. Hence, in

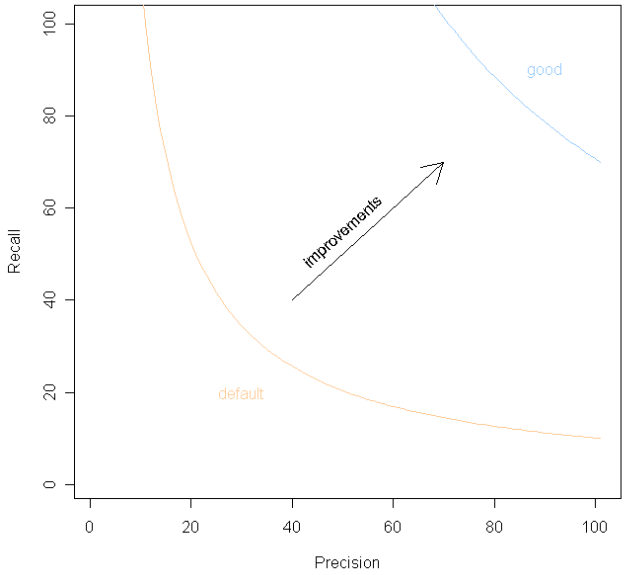
text categorization precision is the number of correctly classified items divided by the number of all items labeled as belonging to the positive class.

Definition 2-15: Recall
Let R be the set of relevant documents and let A be the answer set of a query. Recall is the fraction of relevant documents which have been retrieved.

$$\text{Recall} = \frac{|R \cap A|}{|R|}$$

Empirical studies show that precision and recall are often in a relationship: If the precision increases, the recall declines and vice versa (Fig. 2-6). In terms of information retrieval this means that if we have a retrieval function which is tuned towards high recall (for instance by incorporating synonyms and taxonomies) the precision will be low due to many false positives hits.

Fig. 2-6: Precision – Recall trade off graph



While the previously described precision-recall trade off always holds, the slope of the curve varies from system to system. The difference is the strength of the relationship between the two. In the default case, the area in which both, precision and recall have high values is small (red curve in Fig. 2-6). In a good case, the area is higher (blue curve in Fig. 2-6).

In IR, reference collections (e.g. the TREC collection) are often employed to determine the performance of retrieval algorithms. A reference collection consists of a set of text documents and a set of narrative information needs. A group of specialists mark the relevant documents for each information need.

In order to conduct the evaluation, the information needs are converted into queries. Then, the retrieval algorithms are executed on each query and the answers are compared with the expected answers. The performance is usually expressed based on 11 standard recall levels, i.e. the precision is calculated for the recall levels

of 0%, 10%, …, 100%. Often, the precision-recall curves are averaged over multiple queries to yield a representative result.

The usage of precision and recall is straightforward for small collections where all documents are known. In the Web however, the maximum recall of a query can't be determined because the total amount of correct answers is not known. Precision and recall are therefore problematic measures for the Web.

Chapter 3

Adaptation in Information Retrieval

"The only true wisdom consists in knowing that you know nothing.
And in knowing that you know nothing, that makes you the smartest of all."
Socrates (469 BC – 399 BC)

3.1	User modeling	51
3.2	Personalized search	54
3.3	Recommender systems	62
3.4	Algorithms for recommender systems	64
3.5	Discussion	67

The steady growth of accessible information in the Web as well as in corporate environments confronts users with new challenges. Adaptive techniques seem to be a promising solution for coping with the prevailing information overload. Essentially adaptive techniques conduct the difficult task of providing only the relevant information for a user's specific interests. This chapter gives an overview of fundamental concepts in the area of adaptive systems. We begin with user modeling, continue with personalization of search results, and then address recommender systems. Finally, we discuss several aspects of applying adaptive systems in corporate environments.

3.1 User modeling

In literature, the term user modeling is used for describing the investigation of user models. User models are used to represent a user's characteristics. Personalized, adaptive systems infer from the model a user's preferences, so that actions, services, user interfaces, etc. are adjusted accordingly. Software systems which adapt to the user's information needs are called "adaptive systems". These are clearly distinguished from "adaptable systems", in which a user can adapt the system manually to his needs [Jameson 2003]. Adaptive systems try to automatically detect the user's needs and to self-adjust to an individual. We follow the definition of user adaptive systems given by [Thompson et al. 2004].

Definition 3-1: Personalized, user adaptive system [Thompson et al. 2004]
Personalized, user adaptive systems obtain preferences through interactions with users, keep summaries of these preferences in a user model, and utilize this model to adapt themselves to generate customized information or behavior.

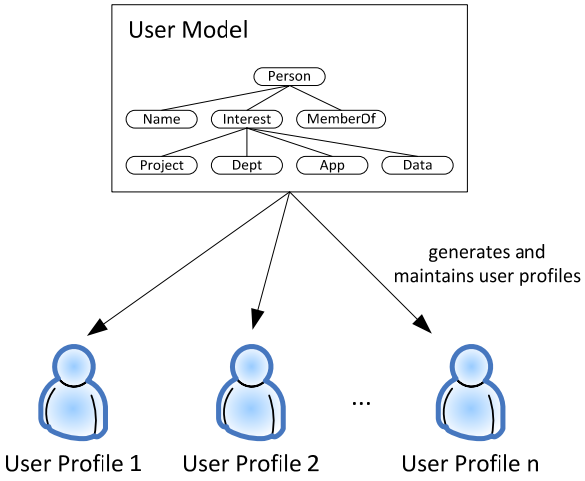
User-adaptive systems rely on user models and user profiles for customizing the system to the individual. Next, we give a definition of our understanding of user models and user profiles.

Definition 3-2: User model
A user model describes generic characteristics of users, relevant to their interaction with the system. It is an abstract representation of user properties and preferences which is used to adjust a system's actions and services accordingly. A user model also generates and maintains user profile instances.

Definition 3-3: User Profile
A user profile is an instantiation of a user model for an individual user. It contains the personal properties and preferences of the individual.

Fig. 3-1 shows a user model and its relations to the user profiles. In the user model, a hierarchical structure is depicted, which represents the captured information content (i.e. properties and preferences). Currently, no standard format for user models exists. Therefore user models' content and format is highly application-dependent, leading to a low re-usability across systems. The user model applied in context of this thesis will be based on ontologies (cf. Chapter 4 and Chapter 6). The ontology provides a semantic structure for storing information about individuals and it enables to conduct logical inferences based on rules, i.e. deciding which actions the adaptive system should take.

Fig. 3-1: Relationships between a user model and user profiles



The following chapter gives an overview of user model types. Then, applications of user modeling in the context of IR are discussed.

3.1.1 User model types

[Thompson et al. 2004] distinguish five types of user models: individual user models or stereotypical user models, information content of models, explicit or implicit user model data acquisition, handcrafted or learned models, and short-term or long-term models. Next, we briefly outline each point.

Difference 1: Individual user models vs. stereotypical user models
So far, when we talked about user models, we implicitly referred to individual user models, i.e. a model, generating user profiles for individuals. However, these are not the only type of models. The second type is called "stereotype user model" [Rich 1979]. In contrast to the "individual user model", a stereotype model targets user-roles, like "biologist" or "computer scientist". A stereotype user profile is thus shared between several individuals. For instance, all workers of a company which have an education in biology might share the stereotype "biologist". Stereotype models are thus an abstract form of user models. Adaptive systems employing stereotype user models are trying to adjust the system to the interests of a social community sharing a stereotype. Inherent to the design, stereotype user models provide a higher anonymity than individual user models but lack the high precision provided by individual models. Therefore a good balance between generality and precision must be found when stereotype user modeling is applied.

Difference 2: Information content of user models
The information represented in a user model depends on the application. In principle any kind of information, from simple to complex objects can be stored. A user model can store for instance personal information (name, location, bank account, education, social network contacts, …), user behavior (clicked items, viewed items, bought items, time, …), device information (display, connection, …), preferred settings (file format, screen resolution, …), device information (display, connection, …), goal, current task, information need, required information depth, previously gained experience, previously gained knowledge, etc.

Difference 3: Explicit vs. implicit data acquisition
As indicated in the introduction, a key issue in adaptive systems is the retrieval of user preferences. We can distinguish two basic techniques: explicit requests and implicit measures. Explicit methods require the user to actively give information about himself to the system, e.g. by answering online-surveys, by rating items of interest, etc. Unfortunately people often feel it as a burden to explicitly state their preferences [Avery & Zeckhauser 1997]. Therefore, implicit methods are often employed instead. Implicit methods try to infer a user's preferences by observation, e.g. by analyzing which objects a user visits, how long he investigates the objects, how his navigation patterns are and so forth. The analysis of implicit data is often conducted by means of Web usage mining, i.e. the extraction of patterns from Web logs. An overview of Web mining methods for personalization is given in [Eirinaki & Vazirgiannis 2003]. The decision of using either explicit or implicit data for user profiling also influences a user model's fluctuation. Explicit data acquisition is good for static models, while implicit data acquisition is good for dynamic models. The statement is justified by the necessity to keep dynamic models up-to-date. Arguably, constantly asking a user about his preferences is an impractical approach.

Difference 4: Handcrafted vs. learned models
User models can either be created automatically or manually. An automatic creation of user models can be conducted by means of Machine Learning (ML) [Mitchell

1997]. ML uses any user specific (implicit or explicit) data like explicit item ratings or implicit click-through data in order to create a user profile for the individual. The benefit of this approach is the ability to adjust over time: user profiles are updated based on newly collected data. However, the automatic approach suffers from the cold-start problem, i.e. the adaptive system does not immediately work for new users because the ML algorithm has not enough data to create a user profile. Manually created user models do not have this drawback. Though, the manual work required by this approach is a drawback. The creation of user profiles, which is done by domain experts, takes time as well as the adjustment of user profiles due to changes in interests.

Difference 5: Durability of user models
The amount of time a user profile is valid can vary from system to system. In one case, a profile might be only used for a single session while in another case a profile might be used for a lifetime. The first is referred to as short-term models and the second as long-term models. E-commerce (online stores) usually applies long-term models. Amazon for instance, stores the history of ordered products for the lifetime of a user's account. An important issue in long-term models is the information half-life. If a model does not adjust properly over time, it will not be able to match the user's true information needs. As a consequence, Amazon might give bad recommendations for customers who change their purchase behavior. Just imagine a student, who buys many books about linear algebra to prepare for an exam. Later, he might still receive recommendations for math books even though he is not interested anymore. Short-term models don't have this drawback. However, they are ignorant to past events, possibly missing important aspects of user behavior.

In context of this thesis, we prefer stereotype user models over individual user models due to the inherent privacy issues (Chapter 3.5). Data acquisition is done by means of explicit methods. The main data sources are administrative databases, which contain user preferences such as involved projects, applications or research areas. Such data is not only available at Roche but in any other major company. Regarding the decision between handcrafting a user model and learning, we decided to handcraft the user model. The benefit is that the cold-start problem is circumvented, effectively providing more incentives for the usage of the prototype. Hence, our user model is a short-term model as past events are not considered.

3.2 Personalized search

The prelude part of this thesis started amongst others with a discussion of the implications of information overload on a user's search experience. A key conclusion was that the one-size-fits-all approach applied by most search engines is not suitable for professional search.

In this section we will discuss promising techniques of personalized search which try to overcome the limitations of the one-size-fits-all approach. Personalized search aims at adapting the results of a search engine to the information needs of the individual, based on some user model and some context of their activity. This is in

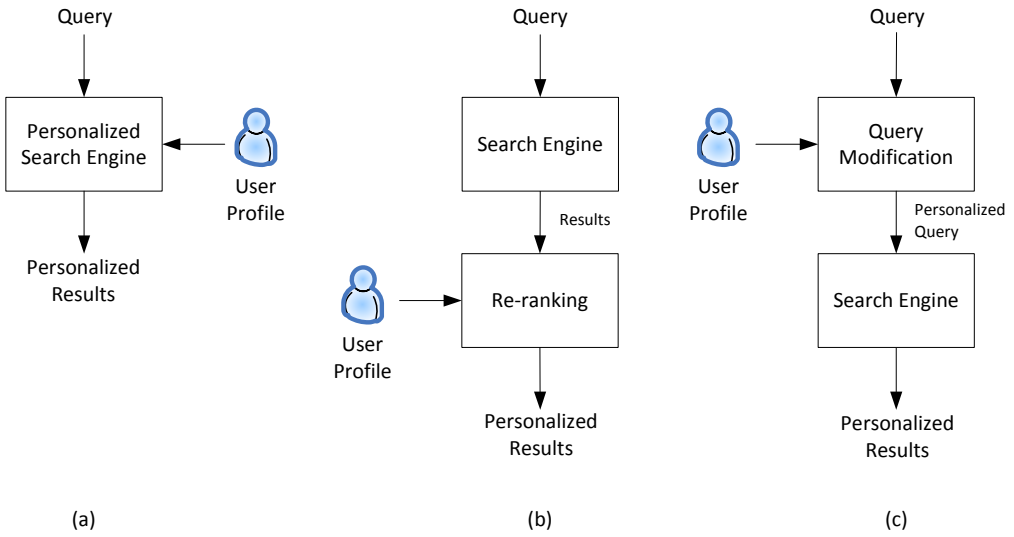
contrast to traditional search engines which return the same result list regardless of who submitted the query.

Next, we introduce personalized search approaches – an area of active research which recently also gained interest among search engine vendors.

3.2.1 Personalization process types

Personalization can affect the search process in three distinct phases [Micarelli et al. 2007; Pitkow et al. 2002]: a) personalization as part of the retrieval process (native personalized search engine), b) personalization based on re-ranking, and c) personalization based on query modification. In each case, user modeling is the major component needed in order to provide personalized results (cf. Fig. 3-2).

Fig. 3-2: Types of personalization processes. The user profile can occur during the retrieval process (a), a-posteriori in a re-ranking step (b), or a priori in a query modification step (c).



Native personalized search engines (Fig. 3-2 (a)) do directly integrate the personalization components into the ranking process. As such they are the most likely type to provide quick query response as well as a good personalization performance. However, the extra computational costs caused by personalization may limit its usage in scenarios where high-traffic spikes are common and delivering answers within a second is a must.

Personalization based on re-ranking (Fig. 3-2 (b)) is in its first steps similar to common search tools, except that the returned results are not delivered to the user but to an adaptation module sitting between the original search engine and the user. The adaptation module re-ranks the results according to the user's profile. The personalized results are then shown to the user. A key issue of this approach is speed, which usually is considerably slower compared to the other two personalization processes. Two common approaches to mitigate the speed issue, is a) to limit re-ranking to the top-k results and b) to consider only the result snippets of the search results instead of the whole full text content. Personalization based on re-ranking can be implemented server-side or client-side. Usually a client-side

implementation is preferred because it allows richer user profiles. Indeed, having client-side software allows capturing additional metrics beyond simple click-through data, such as accessed URLs, time spent, movement of scroll bars, etc. Therefore personalization accuracy is expected to deliver good results.

Personalization based on query modification (Fig. 3-2 (c)) is the last variant. The basic idea is to take the original query and modify it in such a way that a personalized query is obtained which reflects well the user's information needs. For instance if a user is looking at a series of pages about cars and then searches for "jaguar", the adaptive system may expand the query with the terms "automobile" or "car" to exclude false positive hits ("jaguar", the animal). Search is then conducted with the personalized query. The retrieval procedure is thus the same as in the case of a non-personalized scenario. In effect, retrieval speed is not affected by personalization. The downside of this approach is that ranking can only be affected by means of query modification, i.e. it is limited by the expressiveness of the search engine's query language. Fine-granular personalization of search results might thus not be possible.

In our concept we are going to apply personalization based on query modification (Chapter 6). Using a native personalized search engine was not considered, because we decided to use Lucene, which does not offer such services. Further, making Lucene's internal ranking process personalized is a much higher effort than using either a re-ranking or a query-modification approach. Initially, we even used a re-ranking approach. However, the considerable drop in retrieval speed was not acceptable so that in the end, we decided to use a query-modification approach. Because Lucene's query language is quite powerful (it offers term boosting, document boosting, etc.) we are able to achieve similar effects as in the case of re-ranking but with de facto no performance drawbacks.

3.2.2 Methods for personalized search

The previous section gave an overview of personalized search and it outlined that the user profile is an integral part of personalized search. Learning and keeping the user profile updated requires feedback data which can be either implicit or explicit. As a matter of fact, most applications in the area of personalized search are based on implicit feedback data because people are usually reluctant in providing explicit information [Avery & Zeckhauser 1997].

In [Micarelli et al. 2007] six types of personalized search approaches are distinguished: Personalization based on 1) the current context, 2) the search history, 3) rich user models, 4) collaborative approaches, 5) result clustering, and 6) on hypertextual data. In addition to the six types distinguished in [Micarelli et al. 2007], we define a seventh type denoted as "personalization based on the working context". We briefly outline each type next.

The idea of the first type, "personalization based on the current context" is to infer a user's context based on his current activity. In particular, implicit data such as currently viewed documents, viewed emails, and open web pages are taken into

account in order to predict a user's information needs. It requires thus various log data from the desktop which is gathered by monitoring the user's activities. Hence, client side software has to be installed in order to monitor the activities. A system implementing this approach is e.g. Watson [Budzik et al. 2001]. Because distributing and maintaining client side software in a corporate environment is related with much higher efforts than in the case of server side software, we do not further investigate such methods in the context of this thesis.

The second type, "personalization based on search histories" is based on the exploitation of the user's search history – a rich source of implicit data, consisting of information such as past queries, browsed pages, and selected results. Search history data can be easily recorded on the server side without the need to install any client side software. The convenience of this data source resulted in many publications investigating this type of personalization. Because of its popularity we are going to discuss personalization based on search histories in more detail in the following section.

The third type, "personalization based on rich user models" makes use of implicit and explicit feedback. Basically, the user supplies additional explicit feedback to the system by voting on the relevance of retrieved results. The additional data allows a richer representation of the user's information needs. Research in this area made often use of semantic networks to store concepts of a domain and the associations between them [Asnicar & Tasso 1997; Gentili et al. 2003; Micarelli & Sciarrone 2004]. While explicit feedback enables without any doubt better personalization, it has the drawback that each user must invest time in building his profile. In the light of the time constraints present in a corporate environment, we do not expect such approaches to be accepted by the professional user community. Therefore, we do not further consider this type of personalization in this thesis.

The fourth type, "personalization based on collaborative approaches" is about personalized recommendations. A common approach is for instance the application of social filtering algorithms. Here, the neighborhood of a searcher is identified. The neighborhood consists of users which are predicted to share a similar interest as the searcher. Having identified similar users, recommendations are given to the searcher based on his neighborhood. The recommendations typically include items which are known by his neighbors but not by the searcher. Recommender systems are introduced in more detail in Chapter 3.3.

The fifth type, "personalization based on clustering" is not a personalization approach in which the system re-orders or recommends search results. Rather, results are clustered depending on the given query so that a user can slice and dice through the result set. The personalization lies thus in the ability of the user to influence the browsing experience. Systems offering such an approach are for instance Vivisimo (Chapter 2.5.4) and their Web application Clusty[2]. A frequently mentioned drawback of search results clustering is that the cluster labels as well as

[2] http://clusty.com

the cluster hierarchy are ad-hoc. The problem is that both are determined automatically. In effect, labels as well as the hierarchy might differ significantly when the result set changes. Further, cluster labels might be named odd and the hierarchy might not be structured intuitively. This can confuse users. Therefore, we do not consider clustering in this thesis but classification (Chapter 6.3). Classification has the advantage that the categories are precisely defined a priori and that the class hierarchy is static. A personalized browsing of search results is achieved by enabling facetted navigation on the classification hierarchy.

The sixth type, "personalization based on hypertextual data" is particularly about personalization in the Web. The idea is to extend existing hyperlink-based algorithms (Chapter 2.4) by incorporating a user's preferences into the algorithms. In [Qiu & Cho 2006] for instance, a personalized version of the TopicSensitivePageRank [Haveliwala 2003] is described. Currently, we do not consider this type of personalization as relevant for intranet environments. First, the hypertext part of intranets differs from the Internet and thus methods based up on this structure might not yield the expected performance. Second, the majority of content in the investigated department is not located in the hypertext part (Chapter 5).

The seventh type, "personalization based on the working context" enables a personalized view of the search results based on the environment the user is working in. Particularly, information such as the employee's current research activity, involved projects, involved groups, department (task and area of the department), and the educational background is employed in the user profile. This kind of information is usually available in every larger corporation in form of administrative databases. This demographic data enables domain experts to define the adaptation for certain user stereotypes. This kind of adaptation is static, as the user model is not updated by a feedback mechanism. However, such a feedback loop could be incorporated by considering the search histories of a stereotype. We investigate personalization based on the user's working context in detail in the core part of this thesis (Chapter 6.4, Chapter 6.5.2, and Chapter 8.3.3).

Finally we want to mention, that there are of course various hybrid approaches making use of multiple personalization types. In [Sugiyama et al. 2004] e.g., the user preferences are not only determined by the search history but also by means of collaborative filtering (i.e. users with a similar interest).

3.2.3 Personalized search based on the search history

The search history contains all past search actions of a user. The user model typically contains queries, query-clicked item pairs (click-through data), and timestamps. Users can benefit from search histories in two ways: First, if they know they have seen an item in the past they can re-find it by searching the history. Second, click-through data can be used by search engines to fine-tune the ranking of results. The latter must not necessarily be used to personalize results but can also be used regardless of the searcher so that results which have been once selected by a searcher are boosted for future similar searches. As such we are going to discuss two use-cases of the search history. In this section, we describe how the search history

can help to personalize search results. In the following section, we briefly discuss how the search history can be used to fine tune the ranking of results in a non-personalized environment.

Approaches exploiting the search history can be distinguished in online approaches and offline approaches. In the first case, the user profile is updated immediately. In the latter case, search history data is first collected and then processed in a distinct step (e.g. over night) before the user profile is updated. The benefit of applying an offline approach is that more complex algorithms can be implemented.

In [Raghavan & Sever 1995] an online approach is described which is based on matching current queries with past queries. Past queries as well as selected results are stored in a database. In case a significant similar past query to the current query is found, the past results associated with the query are proposed to the user. The challenge is to correctly identify similar past queries – a difficult task due to the short nature of queries and the inherent synonymy and polysemy problem.

In [Bharat et al. 1998] a personalized Web-based news paper system is described. They also make use of term-term similarity. Instead of comparing short queries however, they collect a set of keywords from articles in which the user shows a significant interest (measured by implicit metrics such as selected articles, time spent reading an article, movement of scroll bars, etc.). These keywords are then stored and weighted in the user's profile. The weighting of the keywords depends on the occurrence of the keywords in the viewed articles, i.e. the more often a keyword occurs in the set of viewed articles, the higher the keyword's weighting in the profile. Search results are re-ranked at query time by boosting articles containing keywords of the user's profile. The importance of an article is not only determined by the fact that a keyword of the user profile occurs but also by the weighting of the keyword in the profile. In addition to that, searchers have the freedom to adjust the level of personalization by moving a slider. Notice, even though the described system is web-based it does not entirely run on the server-side. Indeed, the news-reading client is implemented in a Java-Applet. However, such a system could nowadays be easily implemented entirely server-side.

The previous two approaches are based on keyword similarity and suffer thus by the synonymy and polysemy problem. Several papers [Liu et al. 2004; Middleton et al. 2004; Pretschner & Gauch 1999; Speretta & Gauch 2005] have been published which circumvent this issue by capturing a user's interest in terms of higher-level topics instead of individual keywords. In other words, each document is classified into a topic taxonomy and the interests of a user into a specific topic is stored in his profile. In case of the World Wide Web, the taxonomy of the Open Directory Project is usually applied. The user profiles are updated similarly to term-term approaches, i.e. the topics of the pages a user visits are stored in his profile. The difference is that instead of keywords, topics and the interest strength into these are stored.

Topic-based personalization of search results is done as follows. After transmitting a query, a result profile is calculated for the documents. The result profile contains the

mapping of the documents into topics. Given the result profile and the user profile, a conceptual match (reflects how well the topics of the results match the topics the user is interested in) between the two is calculated. The conceptual match is finally linearly combined with the native similarity score of the search engine so that the final relevance score is obtained:

$$\text{FinalRank} = \alpha\text{NativeRank} + (1-\alpha)\text{ConceptualRank}$$

The influence of the conceptual rank on the final score can be manipulated by adjusting the parameter alpha. The closer to 1, the more importance has the native rank, and the closer to 0, the more importance has the conceptual rank.

Even though the described approach is shared by the various topic-based approaches at the core, there are differences. The topics for instance, are captured in some cases by regular taxonomies and in other cases by ontologies. Ontologies have the advantage of offering stronger semantics. Ontology-based personalization approaches are discussed in the next chapter, after having introduced the foundations of semantic technologies. Another difference concerns the method by which a document is assigned into a topic: manual vs. automatic. A further difference related to the classification is that some methods allow a document to have multiple categories while others consider at most one topic per document. Last, we want to point out that also the construction and the representation of the user profile may differ from one approach to the next.

A completely different approach is taken by the CubeSVD algorithm [Sun et al. 2005]. The CubeSVD method targets the user space, the query space, and the document space as well as their associations, i.e. user associations, query associations, and document associations. In order to do so, they analyze the search history and extract click-through triples, i.e. {user, query, viewed document}. Given these triples they perform a higher-order singular value decomposition of the triples so that they receive quadruples, i.e. {user, query, viewed document, weight}. The output captures the latent factors that govern the relationships among users, queries, and documents. Compared to other standard methods, CubeSVD achieves encouraging search results. However, the offline computation of the higher-order singular value decomposition is quite expensive. Another benefit of CubeSVD is that it can be applied also in non-web environments as long as three-way relations exist.

Personalization algorithms have already found their way in several general all-purpose search engines. A popular example of a search engine which is based on the search history is Google's personalized search. Google exploits past page views and immediately updates the user profile using an online approach. Unfortunately there are no details known about the performance of their algorithm. Further, the public knowledge about their approach is limited to the information filed in their patent application [Zamir et al. 2005].

3.2.4 Adaptation of search results based on the search history

The search history can be also used to adapt search results regardless of the current searcher. This kind of adaptation is thus neither user-specific nor stereotype-specific, as it affects all users the same based on their common search and clicking behavior.

In [Joachims 2002] an approach is described which uses feedback data to learn the relative order of result items. Joachims conducted an eye-tracking study in which he analyzed how people read a search engine's results. The study revealed some interesting patterns which he formulated into preference rules. For example, if in context of a query, the third item is clicked but not the first and the second, then the third item is considered more relevant than the first two. This and similar rules have been used to optimize a meta-search engine.

In [Xue et al. 2004] click-through data is used to introduce more accurate metadata for web pages and to improve search performance. Each clicked page receives the query terms as metadata. Using a neighborhood approach, metadata is also introduced to related, so called co-visited, pages. As a result, documents whose metadata match parts of the query terms are ranked higher due to the tf-idf measure.

Because the investigation of this thesis is conducted in a corporate environment, it does not surprise that privacy concerns arise in case of personalized search. Hence, prior to the investigation of personalization approaches based on the search history, we first verify whether non-personalized adaptation of search results based on the search history does significantly improve information retrieval (Chapter 8.3.2). The method we are going to apply and evaluate corresponds to the naïve feedback mechanism described in the paper by Xue et al. (Chapter 6.4.3).

3.2.5 Other applications of personalized search

The adjustment of search results is usually just one piece of the features which revolve around user modeling. Current personalized search tools like iGoogle[3] or LeapTag[4], offer a large selection of profile based features such as: Social search, search history, personalized advertisement, data privacy, location-based services, and recommendation.

Social search (e.g. WikiaSearch[5] or Google's SearchWiki) is a recent Web 2.0 trend in search engines. The idea is to let people re-arrange and annotate a search engine's result items. Re-arrangement enables a user to explicitly determine a new order of result items in context of a query. Consequently, reused queries will show the items in the modified order. Annotations enable users to attach any kind of textual information to a result item. [Noll & Meinel 2007] describe a personalization approach based solely on annotations, i.e. social bookmarking and tagging: Search results which are bookmarked or tagged are ranked higher than other results. Because the feedback can be shared, individuals can benefit from the crowd's re-

[3] http://www.google.com/ig
[4] http://www.leaptag.com/
[5] http://search.wikia.com/

ranking and annotation activities. In essence, the more people participate in social search the better the improvement over traditional search engines. A user model in social search will store explicit data such as annotations and preference information (e.g. Web page P_i is more relevant than Web page P_j in context of a query q).

Personalized advertisement provides context-dependent ads to individuals and is widely used in e-commerce and in the search engine market (e.g. Google and Yahoo!). The technology is often provided by commercial marketing companies such as DoubleClick[6] Inc. DoubleClick collects and stores implicit user information in their database. The identification of users is usually conducted by means of cookies. The user profile is then used by DoubleClick's affiliated sites to decide which advertisement to show to the visitor. In case no user-profile is available, displayed commercials are often selected depending on the visitor's location, i.e. IP address.

User profiling often raises concerns about privacy. Indeed, Web surfers are often tracked by various cookies (such as those from DoubleClick) without their knowledge and explicit approval. However, there are also many portals which support the management of user profiles. In case of personalized search tools, such a feature allows users to toggle the recording of the search history, or to delete recorded data. Further, the modification of other explicit data is possible.

Recommendations have been made popular by Amazon's "People who bought this book were also interested in these books" feature. Similarly, this technology can be implemented in search engines. Instead of bought books, recommendations are given based on clicked items and transmitted queries.

3.3 Recommender systems

Recommender systems are a special type of adaptive systems. They are designed to assist users by guiding them in a personalized way through complex information landscapes and their vast amount of options [Burke 2002]. In contrast to personalization of search results, recommender systems usually explicitly show the user the suggestion instead of applying adaptation unobtrusively. It is thus much more transparent to the user that he gets information from the system. [Burke 2002] distinguishes five basic techniques in recommender systems: Collaborative filtering, content-based filtering, demographic-based filtering, utility-based filtering, knowledge-based recommender systems and hybrid-based recommenders.

Collaborative filtering is the most mature technique and nowadays used by many commercial systems (e.g. Amazon or eBay). The main idea of this approach is to let people rate objects (e.g. products of an online store) and let the system detect commonalities between users based on their ratings. New recommendations of previously unseen items are then generated based on inter-user comparisons. The object ratings are usually stored in a user profile. Collaborative recommender systems can be memory-based or model-based. In the first case, users are directly compared with each other using correlation or other measures. In the second case, a

[6] http://www.doubleclick.com

model derived from the rating history is used for making recommendations. A variety of learning techniques have been used by model-based recommenders, such as neural networks, latent semantic indexing and Bayesian networks. The advantage of collaborative filtering is that no machine-readable representation of the objects being recommended is required. Content-based filtering for instance requires this. The limitations are that a history of user ratings is needed, that new objects will not be recommended as long as they are not rated by others, and that changes in user preference are barely detectable.

Content-based recommendation systems take the opposite approach compared to collaborative filtering. Here, a profile is learned based on the features present in the objects a user has rated and not based on inter-user comparisons. The features used for calculating the similarity depends on the object type (e.g. pictures, video, text, etc.). In case of text objects, the features are word terms and the vector space model can be used to calculate similarity for correlation purposes. The profile is built by means of a learning method such as neural networks, decision trees, etc. Both, collaborative-based and content-based filtering, produce long-term models which are updated as more evidence about a user's preferences is observed. A limitation of the content-based approach is that object features are required for calculating the clusters. First, not all objects have good features for this task. For instance, a movie store might have only the genre metadata for calculating recommendations – a very coarse dimension. Second, because the filtering is based only on the content and not on user feedback, the quality of objects and thus the relevancy of recommendations are unknown. Indeed, automatically extracting an item's quality from its features (e.g. from a publication's text content) is usually not a feasible task.

Demographic-based recommenders classify users into demographic classes based on their personal attributes. It is therefore essential that users provide personal information about their preferences. The information for categorization can be gathered with surveys or by the usage of machine learning algorithms which analyze user profiles. Given the demographic data, the recommender system can identify demographically similar users to extrapolate e.g. from their ratings or purchases in case of e-commerce. Thus, demographic-based recommenders are similar to collaborative recommenders as both form "people-to-people" correlations, but use different data for doing so. The benefit of the demographic-based approach is that no history of user ratings is required. However, if users refuse to give sufficient demographic information, no highly personalized recommendation can be achieved.

Utility-based recommendation systems predict the value of items to each individual user on the basis of a model of the user's preferences. The relevancy of an item for a user is calculated by a user specific utility function. In effect, not only object features and user features can be incorporated into the utility function, but also other attributes such as delivery time of a product or vendor reputation. As a result a user profile might contain a utility function which prefers fast delivery regardless of the vendor's reputation. This approach uses a short-term model because no long-term generalizations about users are done.

Knowledge-based recommender systems (or "Editor's choice" method) provide suggestions based on reasoning over a knowledge base (KB). A KB can be used to model arbitrary objects, their attributes and their relationships (Chapter 4). Here, the KB contains the domain knowledge and the user profiles. In essence, the KB contains the user-item preference information. Based on the KB, a reasoning engine can give recommendations to a user. For example let us assume the recommender system knows about furniture and their semantic relationships (e.g. "part of" or "complements"). Let "table" and "chair" be two different furniture. If a customer buys a dining room table, the recommender system would suggest to buy chairs next. A regular recommender system might continue suggesting tables to the customer. Incorporating the semantic relationships into the recommendation process is a key feature of this approach. Knowledge-based recommenders often employ case-based reasoning methods. The benefits of this approach are that similar to utility-based methods non-product features can be employed, preference changes can be adjusted easily, and that no rating history is required. The limitations of this approach are that knowledge-engineering is required and that the knowledge is static, i.e. the system does not learn.

Any recommender system, which uses more than one of the previously mentioned approaches, is called a *hybrid-based* recommender system. Hybrid approaches try to combine the basic methods in such a way, that the overall recommendation performance is improved, while mitigating the disadvantages of the individual approaches. Table 3-1 gives an overview of how the basic approaches can be combined.

Table 3-1: Overview of hybrid-based filtering approaches

Method		Description
Weighted		A recommendation is computed by the weighted sum of each recommender system
Switching		Recommendation techniques are switched depending on the current situation
Mixed		Recommendations from different systems are presented at the same time
Feature combination	Cascade	One recommender refines the recommendations given by another
	Feature augmentation	Output from one technique is used as an input feature to another recommender
	Meta-level	The model learned by one recommender is used as input to another

3.4 Algorithms for recommender systems

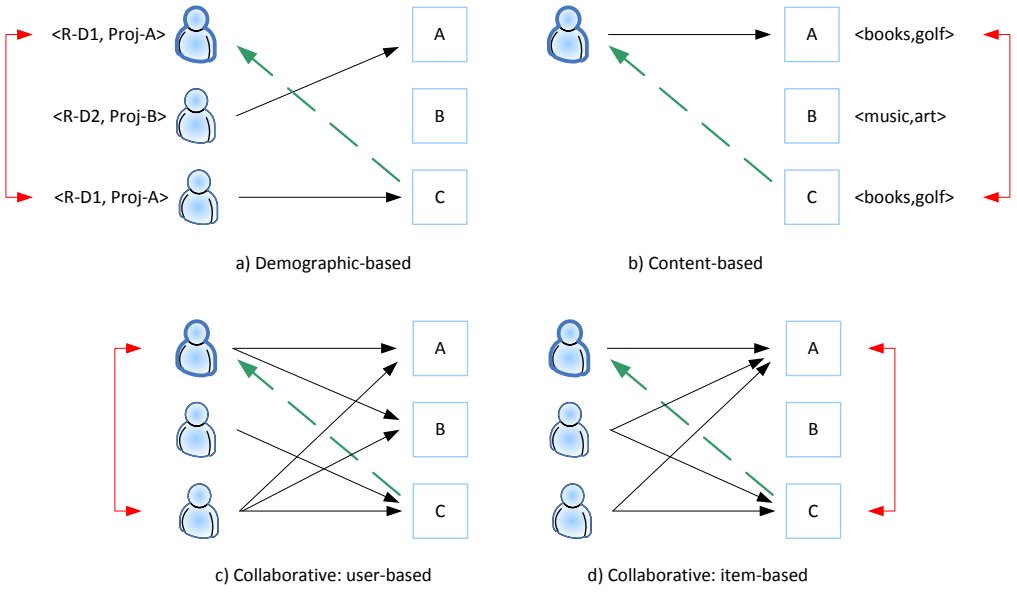
This chapter gives a brief overview of recommendation algorithms for demographic-based, content-based and collaborative-based filtering. Utility-based as well as knowledge-based approaches are omitted as both are highly application-dependent. The first requires a function to be defined, which differs from one use-case to the other. The latter operates on a knowledge-base, whose structure and content (relationships, facts and rules) differs from one application to the other. A specific application of semantic-based adaptive systems is e.g. described in [Henze et al. 2004].

The discussion of the algorithms presented in this chapter are restricted to two classes, namely memory-based algorithms and model-based algorithms [Breese et al. 1998]. We begin with the discussion of memory-based algorithms, which operate over the entire database of users, items and user-item relations. Then, we discuss model-based algorithms, which use the database to learn a user model. The user model is used by the recommender to predict items of interest to the user.

3.4.1 Memory-based algorithms

Memory-based algorithms can be characterized by three steps. First, a similarity function is defined, for detecting similar items or users. Second, the k-nearest neighbors are calculated, to limit the computational costs of determining recommendations. Depending on the considered recommendation technique, the neighbors can be either users or items. For instance, in user-based collaborative filtering, users are correlated, while in item-based collaborative filtering, items are correlated. Third, a prediction function is used to calculate the most relevant recommendations.

Fig. 3-3: Principles of recommender systems. A black arrow represents the interest (e.g. rating or purchase) of a user for an item (A, B or C). Features of users and items are denoted by "<…>". A red arrow marks correlated users or items. The green arrow represents the predicted recommendation for the active user (bold). After [Kim & Seoul 2006].



In *demographic-based* recommenders, the task is to find users with a similar profile, i.e. with a similar feature vector (Fig. 3-3 a). In principle any demographic data (such as age, gender, location, etc.) could be used to calculate the similarity of two users. In the following example, we consider two persons demographically similar, if they have approximately the same age and if they work in related departments. We thus define the similarity function sim(a,u) between a user a and u as follows:

$$\mathrm{sim}_{\mathrm{Demographic}}(a,u) = \mathrm{sim}_{\mathrm{Age}}(a,u) + \mathrm{sim}_{\mathrm{Department}}(a,u)$$

$$\mathrm{sim}_{\mathrm{Age}}(a,u) = \begin{cases} \dfrac{1}{|\mathrm{Age}(a) - \mathrm{Age}(u)|} & \text{if } |\mathrm{Age}(a) - \mathrm{Age}(u)| > 0 \\ 1 & \text{otherwise} \end{cases}$$

$$\mathrm{sim}_{\mathrm{Department}}(a,u) = \frac{|\mathrm{Dept}(a) \cap \mathrm{Dept}(u)|}{|\mathrm{Dept}(a) \cup \mathrm{Dept}(u)|}$$

The similarity age is simply defined as the inverse of the age difference. The set Dept(u) contains the department d in which user u is working, and all superior departments of d. The department similarity is thus defined as the normalized number of common top-level departments. The final similarity function is obtained by a linear combination of both demographic metrics. Given the similarity function, the k-nearest neighbors of the active user are determined, in the following denoted by the set U. In the last step, the prediction function is defined as the weighted average of the ratings $r_{a,i}$ of a user a for an item i:

$$P(a,i) = \frac{\sum_{u \in U} r_{a,i} w_{a,u}}{\sum_{u \in U} w_{a,u}}, \text{ and } w_{a,u} = \mathrm{sim}_{\mathrm{Demographic}}(a,u)$$

In this case, the weights $w_{a,u}$ reflect in the demographic similarity of two users.

In *content-based* recommenders, the task is to find items with features similar to the known preferred items (Fig. 3-3 b). In case the items are scientific publications, the text content of the abstracts could be used as feature vectors, i.e. index term vectors. In effect, similarity between items could be calculated using the cosine similarity measure (as introduced in Chapter 2). Let I be the set of all items, and let $r_{a,j}$ be the rating of the active user a for an item j. Further, let $w_{i,j}$ be the cosine similarity of two items (e.g. abstracts). Then, the prediction function is defined as:

$$P(a,i) = \frac{\sum_{j \in I} r_{a,j} w_{i,j}}{\sum_{j \in I} w_{i,j}}, \text{ and } w_{i,j} = \mathrm{sim}_{\mathrm{Vector}}(i,j)$$

In *user-based* collaborative filtering the task is to find highly correlated users, given explicit user ratings or transaction data (Fig. 3-3 c). Correlated users are determined by using the Pearson correlation function. The Pearson correlation ranges in [-1;+1]. A value of 1 represents perfect correlation, a value close to 0 indicates no correlation, and a value of -1 means that the data is inversely correlated. Let $r_{u,i}$ be the rating of a user u for an item i. Then, the weighting $w_{a,u}$ is defined as follows:

$$w_{a,u} = \frac{\mathrm{cov}(a,u)}{\sigma(a)\sigma(u)} = \frac{\sum_{j \in I}\left(r_{a,j} - \overline{r_a}\right)\left(r_{u,j} - \overline{r_u}\right)}{\sqrt{\sum_{j \in I}\left(r_{a,j} - \overline{r_a}\right)^2}\sqrt{\sum_{j \in I}\left(r_{u,j} - \overline{r_u}\right)^2}}$$

The prediction function can be defined as the weighted average of the user ratings:

$$P(a,i) = \overline{r_a} + \frac{\sum_{u \in U}\left(r_{u,i} - \overline{r_u}\right) w_{a,u}}{\sum_{u \in U} w_{a,u}}$$

Item-based collaborative filtering is very similar to content-based filtering. The difference is that items are correlated and not users (Fig. 3-3 d). Hence, the task is to find items that are highly correlated with the known preferred items. The similarity function used can be the Pearson correlation or the vector similarity.

3.4.2 Model-based algorithms

In model-based algorithms, a probabilistic user model is learned based on user-item preferences, user features, or item features. The learned model is then used to predict items of interest to the user. Given the model, i.e. the previously gained knowledge about the user a, the system calculates the expected rating values $r_{a,i}$ of previously unseen items i:

$$P_{a,i} = E\left(r_{a,i}\right) = \sum_{k \in R} \mathrm{Prob}\left(r_{a,i} = k \mid r_{a,j}, j \in I_a\right) k$$

In order to learn a probabilistic model, several machine learning approaches can be applied, such as decision trees, neural networks, cluster models, Bayesian networks, etc. These approaches are not covered here. However, the interested reader should refer to [Breese et al. 1998] for a discussion of cluster models and Bayesian networks in the context of collaborative filtering.

User models can be updated iteratively by constantly observing the user's interaction with the system. Let P be a user profile, and let D be a document a user observed. Then, the user profile is updated as follows:

$$P' = \alpha P + \beta D$$

The parameter alpha controls the diminishing of the existing interests and the parameter beta determines the relative importance of the viewed item. The latter parameter could be e.g. a user's rating of a publication he just read.

3.5 Discussion

Most of the previously introduced methods would fit well into a professional search tool. For instance it would be convenient, if recommendations (such as scientific

articles) are given for the current research topics. Similarly, a ranking of results tuned towards the information needs of professional searchers would be welcome. The question however is, which of the state-of-the-art approaches are most suitable for professional search in a corporate intranet environment. Several key aspects revolving around this question, such as privacy and feedback sparsity, are discussed next.

Personalization heavily relies on user profiles so that especially in companies, *privacy issues* arise. The following policies describe options of how to abate them. Most importantly, the works council and the users have to be informed about the stored data. Keeping the profiles transparent is also crucial. Users should have the possibility to view their profiles and eventually delete data they do not want to be stored. A third approach is anonymization. This could be achieved by applying personalization on the group level instead of the individual. Here, roles, tasks, projects, etc. are pre-defined and the user can select between them in a multiple-choice manner (Chapter 6.4). Depending on the groups a user has subscribed to, the information portal is adjusted. Given that a set of users belong to a common group, their actions will contribute to changes of the group profile. Thus, in group recommenders, the system tries to fulfill the needs of all group members by maximizing the average member-satisfaction. While this approach guarantees a high anonymity, it has several drawbacks: (a) the initial choice of the proper groups is difficult, (b) group members whose interests have drifted away gain a poor personalization and (c) personalization can't be as accurate as applied on an individual level.

Sparsity, i.e. the low density of “people-item” and “people-people” correlations, is another issue in adaptive systems and mainly affects collaborative filtering approaches. Sparsity is influenced by three factors: the number of items, the number of people, and the frequency of contributed feedback data. In the Internet, especially in e-commerce, these numbers can be beneficial. Amazon for instance, has over 2.3 million books and approx. 50 million visitors per month [Anderson 2006; Wikipedia 2009]. These are very good conditions for providing recommendations based on past purchases. In particular not only the popular items can be targeted but also the long tail, i.e. books which are sold rarely. This picture is quite different when looking at intranet environments. First, the amount of people employed by a single company is by several orders of magnitudes smaller than the amount of people having access to the Internet. Second, the amount of data in intranets usually surpasses the number of workers. This discrepancy is especially large in pharmaceutical companies, which have hundreds of different data repositories. Third, only few data sources are frequently accessed, while the majority of data sources is barely accessed (Chapter 5). Hence, we expect the ratio of feedback per document to be very low with a bias towards popular items.

Under such circumstances collaborative filtering will not be able to work well [Herlocker et al. 2004; Konstan et al. 1997] and should thus be only applied with care. Regarding approaches which use feedback data for ranking purposes the conclusion is twofold. On the one hand, most items will have no feedback data. On

the other hand, we can expect the popular items to gain feedback data relatively fast. In essence, the usage of a feedback based re-ranking algorithm might improve performance [Hawking et al. 2006].

The sparsity could be reduced if generalized concepts instead of individual users and objects are considered. For instance, one could organize people into interest groups such as the educational background "molecular biologists" or the research area "humanization of antibodies". Then, the collaborative process could be applied to the group level instead of the individual. In just the same manner, objects could be organized into concept classes. In case of text documents, they could be organized into a topic hierarchy like "meeting note" or "experimental results of antibody humanization".

The other types of recommender systems, i.e. content-based, demographic-based, utility-based, knowledge-based and hybrid-based are not or barely affected by the previously described sparsity issue. Indeed, particularly the demographic-based approach could benefit from a company's a priori knowledge about its employees. A company usually has a rich source of administrative data of the staff. The database can contain a user's name, department, contact details (phone number, e-mail, office location, site …), educational background (biology, chemistry, statistics, bioinformatics …), involved projects, involved teams, research area, etc. This information is ideally suited to bootstrap a recommender system. Of course, the amount of stored data varies from one institution to the other, but a minimal set of administrative data can be expected to be available everywhere.

Knowledge-based approaches could also be applied very well in intranet environments. A company has a defined area of expertise, i.e. they operate in a closed domain. Parts of the domain could be modeled into a knowledge base which could contain information about a company's jargon, research processes, or any other domain topic. The investment of building such a knowledge base could be high. However, it might pay of in the long term if the time spent for finding information is significantly reduced. As a valuable byproduct, the company gains a better consciousness of the skills and interests of the employees. As outlined in Chapter 1 ("in-house knowledge significantly improves ranking of results"), this is a key hypothesis which is examined in this thesis. In our case, the knowledge base is represented by F-Logic ontologies (Chapter 4). For this purpose we create several ontologies which describe the professional context of the investigated department at Roche (Chapter 6). In particular, we introduce an adaptation ontology which represents the user model. User interests are linked to domain concepts by means of rules (e.g. a rule might state that a user is interested in documents about antibody humanization projects). The modeled knowledge is used by a search engine to adjust the ranking of results. In Chapter 8 we give an evaluation of the knowledge-base approach. In particular, we examine the performance improvement over the baseline (vector space model) approach.

Chapter 4

Semantic Technologies

"The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest number of hypotheses or axioms."

Albert Einstein (1879 – 1955)

4.1	Semantic Web	71
4.2	F-Logic	78
4.3	Semantic technologies in Information Retrieval	82
4.4	Semantic technologies in life sciences	85

Semantic technologies enable machines to understand information, and thus to reason on information. They also provide the lingua franca for communication and exchange of information between systems. The functionality of existing applications could therefore be enhanced in many ways if semantic technologies were used. Search for information could be improved by providing rich metadata for media and content. Consequently information could be discovered easier and viewed in the appropriate context. Semantic technologies could also be used to provide common access wrappers for the integration of disparate systems. Further, discovery and composition of web services could be improved by providing rich metadata descriptors.

We begin this chapter with an introduction to the Semantic Web, and its official language standards. Then, F-Logic is introduced, which is a de-facto industry standard for semantic information systems. Following that, an overview of the state of the art in Ontology-based Information Retrieval is given. Finally, semantic technologies in life sciences are discussed, because the life sciences are considered a driving force of the Semantic Web.

4.1 Semantic Web

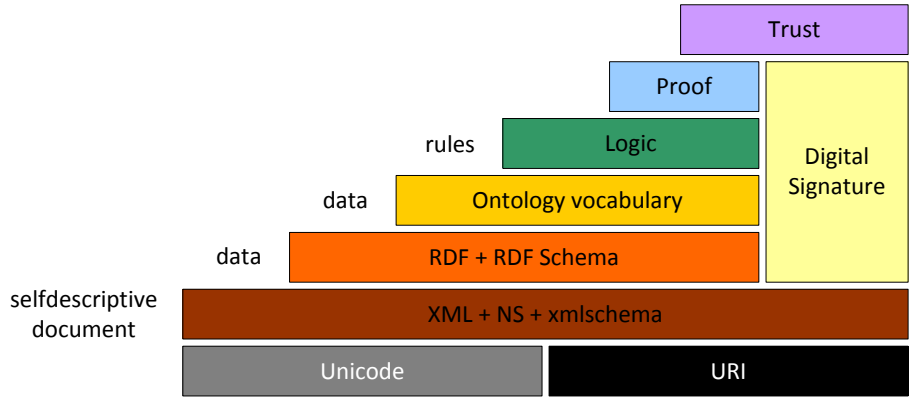
Today's Web content is made for human consumption. Computers do not understand the meaning, i.e. the semantics, of content in the Web. For instance, it is a difficult task for computers to recognize the different meaning of "It always rains on Mondays" and "It might rain on Mondays". Only humans are able to fully combine and properly interpret information from web pages.

The vision of the Semantic Web [Berners-Lee et al. 2001; Berners-Lee & Fischetti 1999] is to represent Web content in a form that can be processed by machines. Ideally, if large portions of Web content are machine readable, intelligent software systems can reason about the data. Indeed, with a Semantic Web and intelligent software on top, we would have an Intelligent Web. Sophisticated services would become possible, such as agents, which would be able to arrange appointments between patients and doctors. Of course, the vision of a large scale Intelligent Web is still utopian. However, the first steps towards such an Intelligent Web are taken by the Semantic Web community.

The Semantic Web is not a new Web but an extension of the World Wide Web. The idea is to extend existing Web content with metadata in such a way that it seamlessly integrates into today's Web. A seamless integration means that both, humans and machines would be able to understand the content.

The Semantic Web is propagated by the World Wide Web Consortium (W3C), an international standardization body for the Web. In order to enrich Web content with metadata, the W3C released several recommendations for Semantic Web modeling languages (RDF, RDFS, and OWL). RDF (Resource Description Framework) is a language for describing information on the Web, RDFS (RDF Schema) is a primitive ontology language and OWL (Web Ontology Language) is a full-fledged ontology language for describing classes and their relationships. The modeling languages are implemented in layers of several technologies as can be seen in Fig. 4-1. The benefit of a layered approach is that people can begin adopting standards without having to wait for the entire Semantic Web layers to be fully implemented. Further, it can enable downward compatibility. In order to do so, software which fully supports one layer, should also support information from any lower layer. For instance, OWL aware agents should be able to understand RDF(S) information.

Fig. 4-1: Semantic Web layers

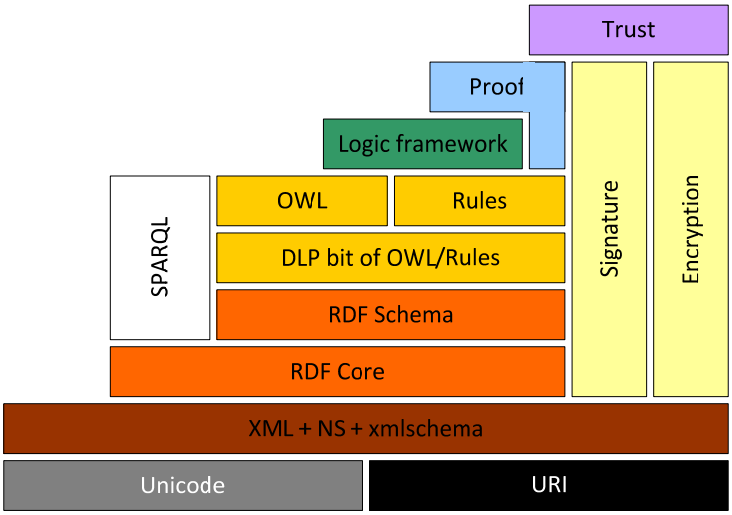


The Unicode and URI layers build the fundament of the Semantic Web layer cake. The first makes sure that international characters sets are used and the second makes sure that objects can be identified. The next layer consists of XML, namespace and schema definitions. It makes sure that Semantic Web standards can be integrated with current XML based standards. The RDF layer is used for writing simple statements about Web resources. RDF Schema provides a primitive modeling

framework for defining vocabularies. The ontology layer builds on top of RDF(S) and enables the definition of classes and complex relations between them. The logic layer enables the writing of application-specific rules for reasoning. The Proof layer executes the rules, i.e. the actual deductive process is done here. Further, it represents proofs in Web languages and validates proofs. The Digital Signature layer is used to prove the authenticity and to detect modifications in documents. The top layer, Trust, is used to test whether to trust the given proof or not. [Koivunen & Miller 2001]

The current Semantic Web stack, as shown before, is currently debated and an alternative version is shown in Fig. 4-2 [Antoniou & Van Harmelen 2008].

Fig. 4-2: Alternative Semantic Web layers



In contrast to the classic Semantic Web layer, the ontology layer is split into two alternatives: the standard Web ontology language (OWL) and a rule-based language. This opens the ontology layer to non W3C recommendations like F-Logic. The second layer which has been added is the Description Logic Programs (DLP) layer. DLP is the intersection of OWL and Horn logic, so that it can serve as a common foundation for the ontology layer. As long as only the DLP part is used for modeling ontologies, an OWL ontology can be mapped with no loss of information to an F-Logic ontology and vice versa.

4.1.1 Resource Description Framework (RDF)

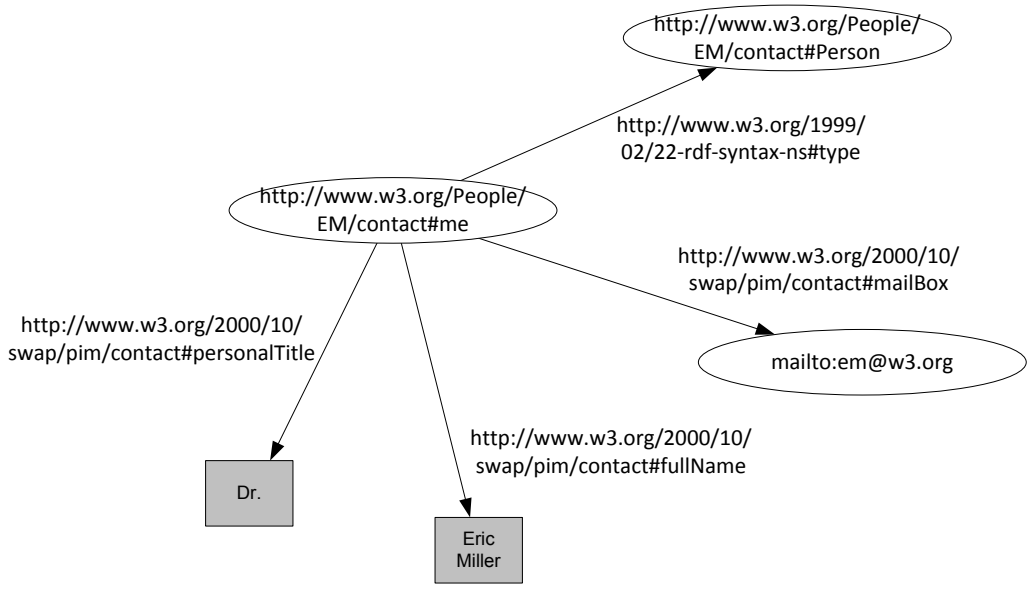
The fundamental modeling language of the Semantic Web is the Resource Description Framework [Klyne & Carroll 2004] which is a recommendation of the W3C since 2003. Using RDF, statements about Web resources like a document's title or an author's main writings can be formulated. The notion "Web resource" is not restricted to retrievable resources. In fact, it is sufficient if a resource is identifiable. For instance, database entries and user preferences are identifiable resources but not Web accessible resources. As indicated, a key idea of RDF is that any resource has a global identifier, i.e. a Uniform Resource Identifier or short URI, which represents the resource. In case of the Web, URLs (which is a special type of URIs)

are often used as identifiers. Even though URLs are used as identifiers, they are allowed to point to arbitrary addresses which might not even exist in the WWW.

RDF is a formal language with a defined syntax, grammar and precise model-theoretic semantics [Hayes 2004]. The semantic foundation provides meaning to RDF statements by applying an interpretation function. The interpretation function can make only few inferences as almost no implicit information is available from RDF statements. RDF statements alone have thus no semantic meaning. They are just a bunch of words and sentences which the machine can't interpret. Only if a vocabulary is defined, can the RDF statements be interpreted. The W3C currently recommends two such languages, namely RDF Schema and OWL, which will be introduced in the next chapters.

Before going on with the basic concepts of RDF, we give a brief example of an RDF graph (Fig. 4-3) which uses URIs to represent a simple statement: "There is a Person identified by http://www.w3.org/People/EM/contact#me, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr." [Manola & Miller 2004]. In an RDF graph, statements are displayed as nodes and arcs. Nodes are used to represent resources, and a directed arc is used for the properties, pointing from the resource to its property-value. Regarding the Semantics, every RDF graph is true because contradictions are not expressible in an RDF model and thus every RDF graph becomes satisfiable [Hayes 2004].

Fig. 4-3: An RDF graph describing Eric Miller [Manola & Miller 2004].



The RDF graph depicted in Fig. 4-3 contains several fundamental concepts of RDF. It shows resources (e.g. ".../contact#me"), properties (e.g. ".../contact#fullName), and their composition into statements. The latter is very important, as it illustrates how small pieces of information (single statements like "…/contact#me" has the name "Eric Miller") can be aggregated into a larger knowledge base (merged set of statements). In this example, four sub-statements are aggregated to receive the contact information of a person.

RDF statements are triples which consist of a "subject", a "predicate", and an "object" – just like in natural language sentences. The subject is the resource about which a statement is given. The predicate describes the property of the resource, and the object is the value of the property.

Subjects can be an identified resource (a resource with a URI) or an unidentified resource, referred to as blank node. A blank node is a node that is not a URI reference or a literal (data value). Blank nodes are used in RDF when structured information needs to be captured.

Predicates can be only identified resources. Hence, blank nodes and literals are not allowed. Predicates describe relations between subjects and objects, like "hasFullName", "hasMailBox", "hasPersonalTitle", etc.

Objects can be any element, i.e. an identified resource, a blank node, or a literal. Literals are atomic data values (e.g. numbers or strings). RDF defines only the generic XML Literal. However, XML Schema data types can also be used, so that a rich repertoire of standard XML data types becomes available. Often, additional operations are defined for typed literals such as subtraction or multiplication. An overview of the various kinds of literals can be found in [Klyne & Carroll 2004].

The RDF abstract data model can be serialized in various formats: RDF/XML [Becket 2004b], N-Triples [Becket 2004a], Turtle [Becket 2007] and N3 [Berners-Lee 2006]. RDF/XML is the official W3C format which is designed for computers, while the other formats are characterized by a lot of syntactic sugar to simplify readability. Namespaces for instance can be easily abbreviated by means of prefixes when using the Turtle notation. The Turtle notation of the statement "John is 24 year old male" illustrates this:

```
@prefix ns: <http://www.mydomain.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ns:John ns:gender ns:male .
ns:John ns:age "24"^^xsd:number.
```

RDF also supports reification of statements, i.e. statements about statements can be modeled – a feature which is very useful, especially in context of the Web. This mechanism could be used in discussion groups, where users want to refer to an existing statement. Another feature of RDF is the support of containers and collections by which resources can be grouped into bags and lists.

4.1.2 RDF Schema (RDFS)

RDF Schema [Brickley & Guha 2004] was designed to provide the vocabulary for RDF. RDFS can be used to create new classes, sub-classes, class-memberships of resources, properties (including domain and range restriction), and sub-properties. It can thus structure RDF resources by creating simple ontologies. RDFS is not only used to define new vocabularies but also for the basic RDF and RDFS vocabulary.

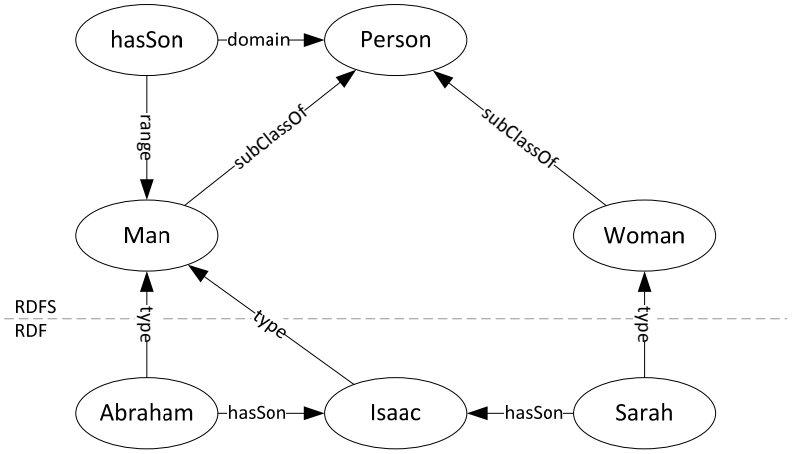
An overview of all RDFS primitives is given in Table 4-1. A class for instance, is a named term and members are defined using rdf:type. A new class named car is created by ":Car rdf:type rdfs:Class" and Ferrari is added as a member of this class by ":Ferrari a :Car" (N3 notation).

Table 4-1: RDF Schema primitives

Category	Type	Description
Core Classes	rdfs:resource	The class of all resources.
	rdfs:Class	Defines a set (the class of all classes).
	rdfs:Literal	The class of all literals.
	rdf:Property	Defines a predicate.
	rdf:Statement	Defines reified statements.
Core Properties	rdf:type	Relates a resource to its class.
	rdfs:subClassOf	Relates a class to its super-class.
	rdfs:subPropertyOf	Relates a property to a super-property.
	rdfs:domain	Defines the domain of a property. Every resource which has this property is an instance of the domain class.
	rdfs:range	Defines the range of a property. The values of a property are instances of the range class.
Utility Properties	rdfs:comment	Provide comments to resources.
	rdfs:label	The label of a resource.
	rdfs:seeAlso	A reference to another resource.
	rdfs:isDefinedBy	A subproperty of rdfs:seeAlso and relates a resource to its definition.

The interplay between RDF and RDFS is illustrated in Fig. 4-4. The ontology consists of a Person which subsumes the classes Man and Woman. Further, a hasSon property is defined which relates the Person class and the Man class.

Fig. 4-4: Sample RDF(S) graph.



The resources Abraham and Isaac are instances of the class Man, and Sarah is an instance of the class Woman. Due to the subclass relationship, all instances are also members of the class Person. The hasSon predicate sets the resources Abraham and Isaac as well as Sarah and Isaac in relationship.

The RDF Schema Semantics defines several simple inference rules [Hayes 2004]. One such rule is the type propagation rule:

```
P rdfs:subPropertyOf R .
IF
A P B.
THEN
A R B.
```

The property is thus inherited. This rule is very similar to the inference rule for sub-class relationships. Another example is the type inference rule. Given:

```
P rdfs:domain D.
P rdfs:range R.
```

The above statement says that the predicate P relates values from the class D to values from the class R. D and R must not be disjoint or distinct. The following inference can then be done:

```
IF
P rdfs:domain D.
and
x P y.
THEN
x rdf:type D.
```

The range is defined with an analog rule. We restrict to the given examples and advise the interested reader to look into the official RDFS Semantics or in the books [Allemang & Hendler 2008; Antoniou & Van Harmelen 2008].

RDF(S) was designed to operate in a large scale Internet environment in which anyone can say anything about any topic (AAA). The AAA principle is a consequence of the Semantic Web's open world framework (cf. next section) which does not assume that all information about a resource is available. Therefore, anyone can make inconsistent assertions as no constraints exist. Thus, everything in RDF is satisfiable so that it becomes possible to compare apples and oranges. As a result, applications employing RDF(S) must be able to deal with such conflicts. The benefit of the AAA approach is its scalability: Distributed RDF/XML statements can be combined into a giant RDF(S) knowledge base, which captures the viewpoints and opinions of millions of people. Something like this would not be possible with XML, because XML aims at giving complete and well-formed information for an application.

4.1.3 Web Ontology Language (OWL)

The Web Ontology Language (OWL) is recommended by the W3C and it is based on RDF(S) [McGuinnes & Van Harmelen 2004]. It has the three building blocks: classes (similar to RDFS classes), individuals (similar to RDFS objects), and roles (similar to

RDFS properties). However, OWL goes well beyond RDFS in terms of expressiveness. Hence, not only simple hierarchies can be modeled but also classes and their relationships can be axiomatically defined, resulting in ontologies which give "an explicit and formal specification of a conceptualization" [Gruber 1993].

OWL comes in three different versions: Lite, DL, and Full (Table 4-2). OWL Lite is the simplest version which has a lot in common with RDFS. In contrast to RDFS, modeling of simple constraints is possible. OWL DL maximizes the expressiveness of the ontology language while maintaining tractability. The abbreviation DL stands for Description Logic, indicating its roots. OWL Full is a highly expressive language. However, due to the expressiveness of the language, complete reasoning is not possible anymore. In fact, with OWL Full one has all freedom of RDF including self-modification. OWL DL may be considered as an extension of OWL Lite and OWL Full an extension of OWL DL, i.e. OWL Lite $\subseteq$ OWL DL $\subseteq$ OWL Full.

Table 4-2: OWL language features

OWL Fragment	Features	
OWL Light	RDFS subset	(sub)classes, individuals
		(sub)properties with domain and range restriction
		conjunction
	(in)equality	
	0/1 Cardinality restrictions	
	Datatypes	
	Inverse, transitive, and symmetric properties	
	Domain restrictions: HasValue, someValuesFrom, allValuesFrom	
OWL DL	Negation	
	Disjunction	
	Full cardinality	
	Enumerated types	
OWL Full	Allow meta-classes (i.e. a class can be an instance of another class) etc.	
	Object properties and datatype properties are not disjoint	

OWL, like RDF and RDF Schema, makes the open world assumption (OWA). The OWA means that we cannot assume something doesn't exist until it is explicitly stated that it does not exist. In other words, because something hasn't been stated to be true, it cannot be assumed to be false — it is assumed that the knowledge just hasn't been added to the knowledge base.

OWL does not have a unique name assumption (UNA), i.e. just because two names are different does not mean they refer to different individuals. In OWL it must be explicitly stated that two individuals having different names are the same.

OWL does not support property chaining (no rule chaining, no composition), arithmetic operations, string operations, partial imports of other ontologies, definitions of views, and procedural attachments.

4.2 F-Logic

Frame-Logic or F-Logic [Kifer et al. 1995] is a formal language for knowledge representation which combines several structural concepts of object-oriented and

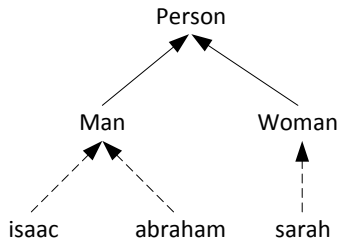
frame based languages. As a result it supports features such as classes, complex objects, inheritance, polymorphism, methods, modularization, meta-reasoning, rules, queries, etc. Actually, F-Logic stands in the same relationship to the object-oriented paradigm, as classical predicate calculus stands to relational algebra. F-Logic has a model theoretic semantics as well as a sound and complete resolution-based proof theory.

We restrict this chapter in describing only the aspects of F-Logic which are relevant for this thesis and use examples for introducing F-Logic. For a comprehensive description of F-Logic (which is beyond the scope of this thesis), the interested reader should refer to the original paper in [Kifer et al. 1995].

4.2.1 The is-a hierarchy

Fig. 4-5 shows a hierarchy of classes and individual objects. Solid arrows represent subclass-relationships and dotted arrows represent class membership of an instance.

Fig. 4-5: An exemplary class hierarchy with instances



The is-a-hierarchy of the running example is expressed with is-a-F-atoms and subclass-F-atoms in F-Logic as:

```
abraham:Man.
isaac:Man.
sarah:Woman.
Woman::Person.
Man::Person.
```

In F-Logic syntax, the ":" character is used to denote class membership and "::" is used to represent the subclass-relationship. For instance, the statement "abraham:Man" states that "abraham" is a member of the class "Man", and the statement "Man::Person" states that "Man" is a sub-class of "Person".

Classes in F-Logic are reified, i.e. they belong to the same domain as individuals. Both, classes and objects, are represented using id-terms. Hence, classes and objects are not discriminated; they can be manipulated in the same language, enabling the usage of a class as a member of another class or as a sub-class of another class.

4.2.2 The object base

Relationships between objects are represented by attributes and methods using signature-F-atoms. A signature-F-atom declares methods for a class and gives type

restrictions to parameters and results. Consider for instance the signature of the running example:

```
Person[name => xsd#string].
Person[father =>> Man].
Person[son =>> Man].
Person[ancestor =>> Person].
```

The first line states that the single valued attribute “name” is defined for members of the class “Person” and that the result object has the data type “xsd#string”. The second and third lines define the multi-valued methods “father” and “son” for members of the “Person” class, and result objects are restricted to members of the “Man” class. The last line defines a method for ancestors. The above signature can be written compactly by combining multiple F-atom statements into an F-molecule:

```
Person[name => xsd#string; father =>> Man; son =>> Man; ancestor =>> Person].
```

Attributes and methods can be overloaded, i.e. methods having the same object name can be used by instances of different classes. For instance we could declare the same attribute “name” to the class “Car”:

```
Car[name=>xsd#string].
```

Using the signature of the “Person” class we can now define instance level statements:

```
isaac[father->abraham].
isaac[son->>{jacob,esau}].
```

The first line states that “abraham” is the “father” of “isaac”, and the second line states that “jacob” and “esau” are the sons of “isaac”. Notice that curly brackets have been used in the second line to enclose the multi-valued attribute.

4.2.3 Rules and queries

In F-Logic rules are used to describe relationships between existing facts and for deducing new facts from existing ground facts. The rules consist of a rule body and a rule head which are separated syntactically by a left-handed arrow “<-“. The rule body can be an arbitrary logical formula consisting of P-atoms or F-molecules which are combined by one of these connectives: OR, NOT, AND, <-, ->, and <->. Variables in the rule body are quantified either existentially or universally. Every variable which occurs in the rule head must also occur in a positive F-atom or P-atom of the rule body.

```
FORALL X,Y X[ancestor->>Y] <- X[father->Y].
FORALL X,Y X[ancestor->>Y] <- EXISTS Z X[father->Z] AND Z[ancestor->>Y].
```

The first line states that if Y is the father of X, then Y is an ancestor of X. The second line extends the definition of ancestor by applying recursion. It states that, if an object Z exists who is the father of X, and if Y is the ancestor of Z, then Y is also the ancestor of X. In case the "ancestor" and "father" methods are overloaded the above rule might also apply to other classes. Therefore, it would make sense to explicitly state the class of X, Y, and Z.

Queries are very similar to rules. In fact the only difference is that queries have an empty head. For instance, the following query retrieves all ancestors of all persons:

```
FORALL X, Y <- X:Person[ancestor->>Y].
```

All variables which fulfill the precedence are bound and returned:

X	Y
isaac	abraham
jacob	isaac
esau	isaac
jacob	abraham
esau	abraham

A rule in F-Logic is stratified if it does not contain a stratification-relevant cycle over a negation. Further, the semantics of F-Logic statements are defined by the well-founded semantics [Van Gelder et al. 1991], i.e. if an answer can't be deduced the affected variables are set to unknown.

Regarding the semantics of F-Logic, they are based on the semantics of first-order-logic, i.e. F-Logic statements are transformed into logic programs: F-molecules are transformed into F-atoms, and F-atoms are transformed into predicates (P-atoms).

4.2.4 F-Logic vs. OWL-DL

In contrast to the object oriented approach taken by F-Logic, the official W3C ontology language OWL uses a property-based approach for describing ontologies, i.e. roles and classes are used for classifying instances. Further, F-Logic uses the closed world assumption (CWA). In CWA the default negation used is "negation as failure", i.e. facts which are not explicitly stated are assumed not to hold. OWL on the other hand uses the open world assumption. Thus, if new knowledge is added in OWL, previous conclusions are not changed. Another difference is that F-Logic uses the UNA, i.e. two individuals with different names denote different individuals. Just like deciding whether CWA or OWA is the better choice, also deciding on UNA vs. non-UNA depends on the use case. As a general rule, in closed domains, like companies, using the CWA and UNA is probably the better choice in most of the cases. In open domains like the Web, on the other hand, OWA and non-UNA are probably the better choice.

In contrast to OWL-DL, F-Logic programs, i.e. query-answering, are not decidable. However, retaining decidability limits the expressiveness of OWL. For instance, it is

impossible in OWL to capture relationships between a composite property and another property (e.g. “parent” and “brother” properties and the “uncle” property) [Horrocks 2005]. This and other expressions can be modeled with the F-Logic rule language.

In F-Logic both, classes and instances are modeled as first-order-logic (FOL) terms while OWL-DL models instances as FOL-terms and classes as FOL-predicates.

Similarly to OWL, F-Logic does also support namespaces. Arithmetic operations, comparisons, aggregations and other tasks which are not well suited to be described by rules are integrated by means of built-ins, i.e. external programs which can be referenced by means of P-atoms.

Last but not least, F-Logic is not an official ontology standard like OWL-DL. Nevertheless, F-Logic can be used as an alternative modeling language for semantic information. F-Logic is for example implemented in OntoBroker, a Java-based software product of the Ontoprise[7] company.

4.3 Semantic technologies in Information Retrieval

Traditional Information Retrieval systems ignore the semantics of indexed content and supplied queries. For instance, if a user searches for vehicles, a document about cars won’t be retrieved because the IR system does not understand that a car is a subclass of vehicle. However, if the IR system would “understand” the content’s meaning, such queries would give better results in terms of precision and recall. In fact, it is a goal of the Semantic Web vision to significantly improve search for information by adopting semantic technologies into the IR process.

4.3.1 State of the art

Nowadays, the combination of IR and semantic technologies is actively investigated by scientists. We discriminate three research areas in semantic-based IR:

1. Search in formal ontologies
2. Domain ontologies as a support for document search
3. Ontology-based adaptation in IR

The first research area investigates retrieval and ranking of facts from ontologies, and the conceptualization of a user’s information needs [Stojanovic 2005]. The retrieval process starts with an “ideal” query which is used to retrieve “ideal” results from a formal ontology. An “ideal” query is a formal, ontology-based query such as SPARQL [Prud'hommeaux & Seaborne 2009], which has precise semantics. A user’s information needs are thus ideally expressed in such a query. Given the “ideal” query, a Boolean search model is used to search a formal ontology, containing non-redundant and non-ambiguous pieces of information. The search result contains all tuples which satisfy the query. Because the query is “ideal”, the results are assumed to have a precision of 100%. Hence, there is no notion of approximate answers and the only concern is the ranking of results (ontological instances) by relevance.

[7] http://www.ontoprise.de

Therefore, semantic search can be viewed as a data retrieval task (Chapter 2.5.3) in contrast to an information retrieval task.

For real-world corpora this approach is currently utopian as the majority of information is in unstructured text documents [Castells et al. 2007]. The problem of formalizing this knowledge on a large-scale is unsolved. Replacing the documents by formal knowledge bases is not feasible – for humans – because a written text document has another quality than a couple of facts buried in an ontology. A better way to deal with this issue is to extend existing unstructured sources with formal knowledge, as proposed by the Semantic Web vision. Regarding the construction of formal ontologies, at latest when the properties of individuals are filled with string values, unstructured information is created again. This time however, hidden in an ontology. Last but not least, the average user will not be up to the task of formulating an "ideal" query in terms of using a formal query language. Several studies, which examined the search behavior of users, show that the large majority relies on a few keywords for describing their information needs [Silverstein et al. 1999]. Boolean operators and other features such as advanced search forms are barely used. With this in mind, the argument of achieving a precision of 100% gets quite weak.

The second research area investigates ways and means for combining the benefits of ontologies with traditional information retrieval approaches [Castells et al. 2007]. Here, the focus is set on documents and not on knowledge bases. This has several consequences: First, queries are not formulated using a formal ontology language. Second, results cannot be expected to be 100% precise. Third, returned results are text documents (possibly including semantic annotations) and not formal ontology facts. The role of the ontology is reduced to a mere mean for improving the retrieval performance by providing additional metadata to unstructured text documents. The metadata or annotations can be gathered manually or automatically. In [Castells et al. 2007] annotations were used to relate documents with domain concepts. The automatic annotation algorithm employed by them uses several heuristics which are outlined in their paper.

In their approach two indexes are created: An annotation index, where the documents' annotations are stored, and a classic document index, where the documents' content is stored. Queries are transmitted to both indexes and the returned results are merged using a linear combination. Of course, in order to query the annotation ontology, the keyword query has to be transformed into a formal ontology query language like SPARQL. The transformation is done automatically. In this step, special attention has to be paid as the keyword's intended meaning must be matched as good as possible by the transformed query. Otherwise, precision and recall could drop significantly. The transformed query is used to find matching documents in the annotation ontology while the original keyword query is used to find matching documents in the document index. As already written, both results are merged, achieving the final ranking of results. [Castells et al. 2007] has shown that the combined approach (annotation ontology + document index) is on average better than the single usage of either of them. The user-centric view is a big

advantage of this approach, because average users can benefit from semantic technologies without the need to learn a formal query language.

Thesauri / taxonomy based IR systems are a relaxed form of the previously described research. Such methods have already been applied before the announcement of the Semantic Web standards. The typical usage of a thesaurus (e.g. WordNet [Miller et al. 1990]) is query expansion. Here, query terms are expanded with their synonyms, super- or sub-concepts, yielding a higher recall.

The third research area is about ontology-based recommender systems [Middleton et al. 2004]. Middleton et al. focus their research on the scientific community and their need for relevant scientific literature. In their approach two ontologies are employed: A research topic ontology and a user profile ontology. The first ontology is used to classify scientific publications into research topics. The classification is done automatically by means of a trained machine learning algorithm. The second ontology stores the user profiles, i.e. the persons' current interests in research topics. The interests are gained by tracking viewed publications and their ontological topics. Using a hybrid approach, consisting of collaborative filtering and content-based filtering, recommendations are given to the users. In addition to previously seen items, the recommendation algorithm exploits the hierarchical topic ontology: 50% of a user's interest in a class is inherited to its super-class. Therefore, broader recommendations can be given. Another benefit of using ontologies for user profiling is the ability to employ external knowledge bases. These can help to bootstrap the recommender system, i.e. to reduce the cold-start problem.

4.3.2 Discussion

Search in formal ontologies is currently not relevant for pharmaceutical research. First of all, formal ontologies are not used yet in industry, i.e. data is not stored in ontologies but rather in databases or in unstructured sources. Further, users would be required to use a formal language for querying the data. However, workers of the pharmaceutical industry are not computer scientists, but biologists and chemists. Hence, access and usage of such a formal language must be circumvented by means of a user friendly interface. We believe, however, that in a long-term, formal ontologies can become quite important in pharmaceutical research, e.g. in the area of pathway analysis, where thousands of facts and rules define if and how a molecule interacts with another one.

Domain ontologies to support document search seem to be a promising approach as a mid-term solution. Data in form of unstructured text documents is available in abundance as text documents are the primary mean of capturing knowledge. While creating a domain ontology for structuring knowledge is feasible, annotating the documents is an issue. Because it requires time and effort, users could be reluctant to contribute manual annotations. Therefore, metadata is ideally created by an automatic approach. If the automatic annotation gives reasonable results, we believe that such an approach could improve IR in professional search. For instance, an automatic annotator could classify documents by topics: in a pharmaceutical company e.g. disease areas.

Ontology-based adaptation is also a promising approach for dealing with the heterogeneity of the information space. A portal with adaptation capabilities such as a recommender system for context-dependent literature, or a dynamic filtering of irrelevant sources at query time, or even the context-dependent adjustment of search results could be very relevant for pharmaceutical companies, especially for researchers.

4.4 Semantic technologies in life sciences

Health care and life sciences (HCLS) are often considered as one of the most affine areas for semantic technologies. Researchers believe that semantic technologies could lead to a significant improvement in terms of data integration, data retrieval, and generation of new knowledge by means of reasoning. Indeed, the amounts of data created in HCLS during the last decades and the need to handle these amounts are beyond comparison. In the beginnings, most data was generated by sequencing projects, like the Human Genome Project[8]. Today, the majority of the data is created by experiments, such as multi-parallel chip experiments for gene expressions, proteins, metabolites, etc.

4.4.1 State of the art

The HCLS information landscape on the Web is probably best characterized as a clustered space of heterogeneous data having a low degree of semantic connectivity and an impressive growth rate. The problem of heterogeneity has been recognized early, so that nowadays dozens of controlled vocabularies are available for maintaining a common jargon across different databases. The controlled vocabularies are usually taxonomies in which domain specific concepts are described. The Open Biomedical Ontologies[9] (OBO) is a portal for HCLS ontologies that includes amongst others the popular Gene Ontology[10]. Most ontologies of the OBO Foundry are not in a Semantic Web format. Indeed, the Semantic Web format is not officially supported by OBO. The only effort towards Semantic Web standards conducted so far is the re-factoring of existing ontologies, so that mapping to Semantic Web ontology languages can be done easier. Taxonomies such as those in OBO are widespread. “Real ontologies” (which go beyond simple taxonomies) as propagated by the Semantic Web however, are still the exception.

Most research projects, which adopted semantic technologies in HCLS, focused on the need to integrate and analyze data across databases, applications and communities [Cheung et al. 2009]. Probably the biggest issue in integrating HCLS databases is the fact that most bioinformatics resources are in the Deep Web [Vandervalk et al. 2009]. Hence, the URI-centric Semantic Web approach to address a resource by a URL can’t be applied in HCLS, because the data only exists if requested by means of a query form. A common approach to deal with this issue is to build Semantic Warehouses, i.e. to create RDF dumps of databases using arbitrary URIs. Doing so requires the original data to be converted to RDF (in case it is not

[8] http://www.genome.gov/10001772
[9] http://www.obofoundry.org
[10] http://www.geneontology.org

available in this format) and because cross-references are missing, the resource-references must be rewritten to an invented, warehouse-specific URI scheme [Vandervalk et al. 2009]. As a consequence, common entities across different warehouses have different URIs, so that integration and mapping between resources becomes an issue. Another problem of Semantic Warehousing is that queries are limited to a fixed set of databases. Ad-hoc queries across random databases are not possible.

The Semantic Warehouse approach is used for instance by the Bio2RDF[11] project, which tries to make in a short term as much data as possible available in RDF format. In the long term various data sources are integrated, enabling SPARQL queries.

4.4.2 Discussion

Wrapping it up, HCLS has still a long way to go until semantic technologies bring the expected improvements in terms of integrating the information landscape in the Web. Similarly, semantic technologies in industrial pharmaceutical research and development are creeping. In-house databases are either not integrated at all or if they are, a warehousing approach is used. Ontologies as a canonical schema for federated database integration are not yet used.

Regarding the usage of unique identifiers, the issues encountered on the Web can be mapped to the problems encountered in intranets. In addition, life science companies have their own and unique jargon for chemicals, molecules and compounds. The in-house jargon can differ significantly from the names used in the Web. Therefore, it is common, that external vocabularies are mapped to in-house terms. In fact, the usage of ontologies (e.g. the GeneOntology) at Roche and probably in other pharmaceutical companies as well, is restricted to controlling the professional jargon, i.e. the terminology used within the company.

[Cheung et al. 2009] expect that Semantic Web technologies will be further adopted by industry and academic research and development, with a focus on linking data across silos. We share this view, but we believe that only when semantic technologies are an established and widely recognized technology in industry, will global pharmaceutical companies adopt them. Until then, most open issues will be solved pragmatically, namely on a syntactic rather than a semantic level.

[11] http://bio2rdf.org

Part III

Core

Chapter 5

Characteristics of professional search in pharmaceutical research

"Never regard your study as a duty, but as the enviable opportunity to learn to know the liberating influence of beauty in the realm of the spirit for your own personal joy and to the profit of the community to which your later work belongs."

Albert Einstein (1879 – 1955)

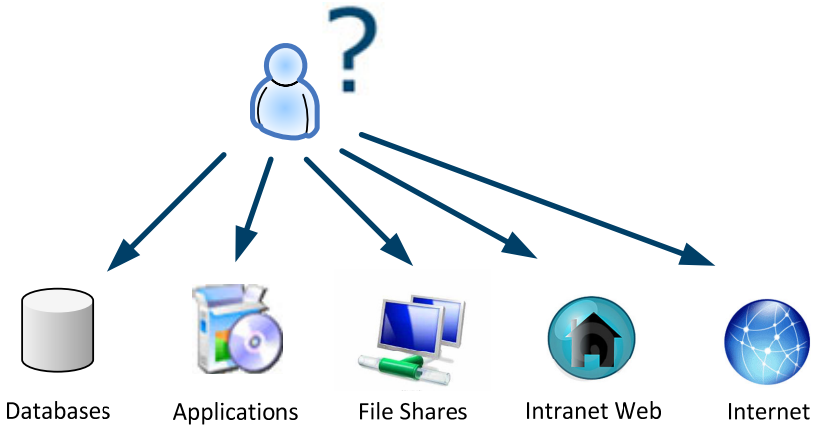
5.1 Evaluation of the initial situation in the department investigated 89
5.2 Quality characteristics of a professional search tool.....96

Professional search has other aims than general search, as has been already indicated in Chapter 2.4.3, where general and specialized search engines were introduced. The purpose of this chapter is to provide an understanding of the needs of professional searchers. In order to gain this knowledge, we conducted an evaluation of the initial situation in a specific department of Roche, namely Pharmaceutical Research in Penzberg, Germany, having several hundred employees. The evaluation is the foundation for deducing the requirements for a professional search tool in pharmaceutical research. Although the user needs recognized from the evaluation are specific to employees of Roche, we argue at the end of this chapter that the deduced requirements do well convey to other domains.

5.1 Evaluation of the initial situation in the department investigated

A researcher at Roche is faced with an overwhelmingly large amount of disparate internal and external data. Internal sources are intranet web pages, file shares, databases and applications, while external sources are all freely accessible Internet pages as well as licensed portals (e.g. literature databases). Because of the diversity of sources and due to the variety of navigation and search tools, searchers can get easily into trouble while trying to find their way through the information jungle (Fig. 5–1).

Fig. 5-1: Typically, an employee having a specific information need is confronted with five main sources of data, each having countless data records.



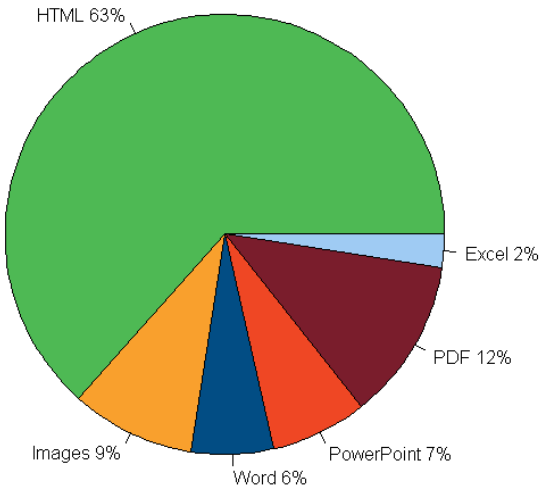
In advance of describing the various user studies conducted at Roche, we first give a description of each source. The focus will be set on in-house data, i.e. the Internet as a source of information will not be discussed, as the Chapter 2.4.3 provided already some ideas of relevant search engines in the context of life sciences. The internal sources are described in terms of structure and content. In addition, specific issues regarding search for information are outlined.

5.1.1 Intranet web

The intranet web, i.e. the hypertext part of the intranet, provides information about departments, projects, administrative procedures, and a variety of domain specific knowledge. Further, it gives access to web-based applications, database portals, and web-based document management systems. The hypertext structure manifests in large parts Roche's organizational structure (Chapter 2.5.1). Departments belonging to a common area and site (e.g. pharmaceutical research in Penzberg) are usually bundled in an intranet web portal. Within such a portal, access to departments is enabled by navigating the organizational structure – a hierarchical tree arranging departments into sub-, super-departments and groups. Typically, a department provides information about offered services, its members and their expertise. Alongside departmental-driven intranet portals, there are also portals which focus on a specific domain of interest. The "Bioinformatics" portal for instance, enumerates domain experts and gives access to applications as well as life sciences databases.

The web site of the Pharmaceutical Research department in Penzberg (in the following denoted as PRPZ-WebSite) contains in total approximately 1.1k accessible files. A small number, given that globally the Pharmaceutical area has over 130k pages.

Fig. 5-2: Distribution of file types on the PRPZ-WebSite.



Among these 1.1k intranet web files, the majority are in HTML format (Fig. 5–2). The remainders are office documents and images. Office documents are often about site maps, manuals, SOPs, forms, organization charts, etc. Most images depict persons and are used to list a department's employees.

A key disadvantage of the intranet web is the thin coverage by intranet web search engines. First, despite the fact that the pharmaceutical division and diagnostic division are part of one organization, they use different search engines[12] which ignore the sources of the other division. A meta-search engine providing access to both divisions does not exist. Hence, a searcher must know in advance in which division the required information is located. Second, the seed pages are not complete, i.e. intranet web pages which are not part of the common web sites (e.g. wikis known only by a small group of people) are often omitted. A searcher might thus not find a relevant answer page. Third, deep intranet web pages such as database portals are not indexed and thus not searchable.

Another issue is the lack of metadata. In effect, neither full-text search over metadata nor semantic search based on metadata is available. Consider for instance the structure of intranet portals. Even though many of them are organized by the departmental structure, this knowledge is not explicitly expressed in a semantic language. Similarly, the description of departmental expertise and a lot of other information is only encoded in human readable form. As a result, queries like "mass spectrometry experts working in the bioinformatics department" are not supported.

5.1.2 File shares

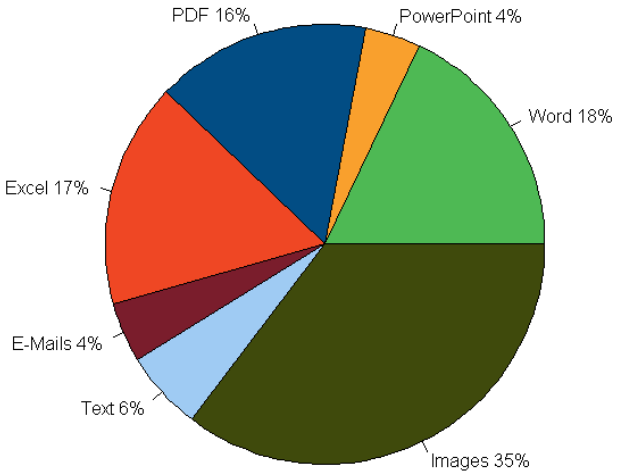
File shares are an important mean of storing and sharing information within and between departments at Roche. File shares correspond to intranet web portals and are structured in a similar fashion. A typical file share contains an "organization" or a "project" top-level branch. The "organization" branch is structured according to the organizational structure, i.e. the folder structure reflects the hierarchical

[12] This will change towards the end of 2009

departmental structure. The "projects" branch, on the other hand, contains project related subfolders, where relevant data from several departments is consolidated.

The amount of stored information differs from one file share to another. In the case of the Pharmaceutical Research Penzberg the file share (in the following denoted as PRPZ-Share) comprises about 600k files. The distribution of the most common file types on the PRPZ-Share is displayed in Fig. 5–3.

Fig. 5-3: File type distribution on the PRPZ-Share.



The file shares' content ranges from office documents to device generated data. Office documents are often about meetings, SOPs (Standard Operation Procedure), lab device descriptions, experiments, projects, scientific studies, etc. Device generated data are usually images (e.g. Gel Electrophoreses pictures), ASCII data, sequence files, etc. Typical document formats of the share are Word, PDF or E-Mail. In addition, many Excel files exist which often encapsulate protocols and analyses of experiments conducted in labs. The plain text files usually contain sequences of genes or proteins. PowerPoint presentations cover a diverse range of different scientific as well as administrative topics. Approximately one third of the files are images. The images depict schematic illustrations of biological processes, gel electrophoresis experiments, and charts generated by lab devices. Beside the listed file types there are a couple of other file formats which are not displayed in the chart. The reason is that they are either irrelevant for user consumption (e.g. technical files generated by lab devices) or they are too few to be displayed properly (e.g. MS Project files and reference databases).

While the intranet web of Roche is dominated by two search engines having a relatively large coverage of their areas, file shares lack a common search tool. We could identify two reasons causing this: Security and technical issues.

Enabling access to a share from other areas is a matter of trust. Indeed, only a small minority of documents (like patents, contracts or salaries) should be highly secured while the majority could have relaxed security settings. However, the reality is differently. In general people from one area have only access to their own file share

and not to others. Only few employees have access to multiple file shares and in case they can access another share, their read rights are restricted to a small sub-branch.

While security restrictions are alleviated, technical barriers need to be surmounted too. The file shares do not have a common security schema. Therefore, a user must be assigned to many different groups in order to have read access to more than one share. However, due to a limitation[13] of the ActiveDirectory, a single user can be assigned only to 1,015 security groups. In effect, giving a user full access to two shares is impossible because the total amount of groups would be higher than allowed.

Similar to the intranet web, file shares do also lack metadata. In particular, it would be useful if the folder structure would be represented by an ontology. Then, a semantic search by concepts and relationships would be enabled. Creating such an ontology should be feasible, because people usually label folders so that the content beneath it is described well. Further, the relationship of folder and sub-folder has also an implicit meaning to the user. The concept of a folder structure is thus related to ontologies having a taxonomical character. The difficulty however is that folder names are barely re-used and that the relations between folders are not explicit.

5.1.3 Databases and applications

Databases are typically used to store large amounts of business relevant information. The in-house databases are closely coupled with their corresponding applications. The applications are used to collect data automatically from lab devices or as a mean for manual data input. In addition, they also provide functionality for searching and browsing the data. What an application understands by “search” and “browse” though, differs from one implementation to the next. In once case it can be a form driven application where an exact search is done, in another case full text search with query term suggestion might be implemented, and so forth.

Currently, discovering databases and efficiently retrieving information from them is not trivial. There is no directory listing all databases, security schemas are tight, and the content of databases is neither described nor interlinked by means of ontologies. Therefore, no software tool for ad-hoc queries across multiple databases exists.

While the first two points (directory of databases and security) could be solved “rather easily”, the last point (integration by means of ontologies) requires a much higher effort. Nevertheless, this effort could pay-off as can already be seen in the company’s Bioinformatics area. There, the GeneOntology together with synonym lists of gene names enabled the linkage and integration of various databases. The gained benefit in terms of search performance is very good considering the fact, that only a few columns (GeneOntology Terms and gene names) have been interlinked.

In chemistry, database integration is much more difficult. A key problem is the naming of the chemicals. First, the name of a compound used within a company is

[13] http://support.microsoft.com/kb/328889

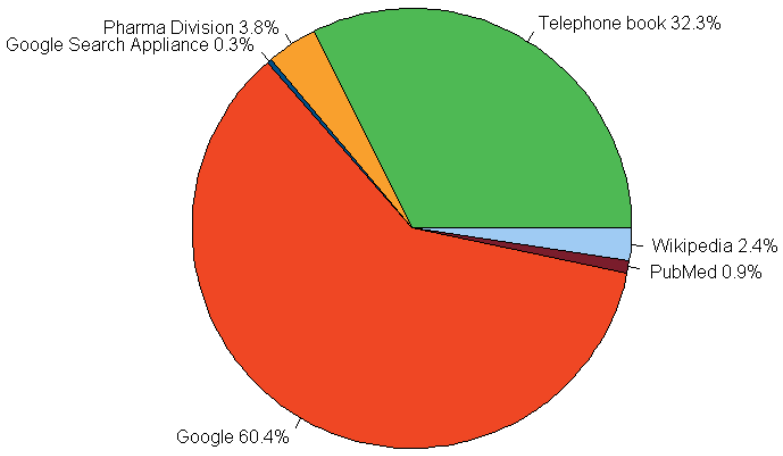
heterogeneous and additionally often different from the name by which it is merchandised. Second, IUPAC[14] (a standard nomenclature for chemicals) names are long and complex. Third, a chemical can have more than one valid IUPAC name. Search for chemicals is thus usually done by means of structure rather than names: A chemists describes a substructure which is then used to find similar chemicals.

Beside the in-house databases, Roche has several licensed databases from third party companies, like patent databases or journal databases. Access to these is granted to an employee after a registration procedure. Alternatively, an employee has the possibility to ask the local library to do a professional inquiry for him. Librarians usually incorporate a large amount of licensed literature databases in their search process in order to achieve the coverage.

5.1.4 Search engine usage on the PRPZ-WebSite – A log file analysis

The web portal of the Pharmaceutical Research division in Penzberg is the default homepage for several hundred employees. In order to determine its effectiveness in retrieving information we conducted a log file analysis measuring the relative usage of the search engines accessible via the portal. We monitored the six search engines available on the PRPZ-WebSite over a period of one month in 2007, resulting in 29,053 logged queries. Four of the six search tools are in-house: Google Search Appliance, FAST, and a telephone book application. Google's Search Appliance is used to index the local PRPZ-Share having approximately 600k files. The FAST search engine indexes most web pages of the Pharmaceutical division at Roche. These are approximately 130k web pages. Last but not least, the telephone book application is a simple web interface which enables full text search on the company's telephone book. The three public domain sources linked from the local homepage are Google, Wikipedia and PubMed. The relative access to each of the six search engines is depicted in Fig. 5–4.

Fig. 5-4: Usage of search engines linked from the PRPZ-WebSite.
Data logged over a period of one month in 2007.



[14] http://www.iupac.org/

The results show that external sources are used more than in-house sources. Particularly Google remains unchallenged. Without any doubt, this is due to its ranking performance and maybe due to its search performance in databases like PubMed, US patents, Wikipedia, etc. On the other hand, we were surprised about the low usage of the Google Search Appliance indexing the PRPZ-Share containing most relevant documents (Chapter 5.1.2). We assume that the main reasons for its low usage were the additional manual log-in procedure and a poor ranking of results. We conclude that people located the files by means of the Windows Explorer, i.e. navigation of the folder structure, rather than by search.

5.1.5 Empirical studies of information acquisition

Parallel to the log file analysis, two independent studies [Maßun 2008; Mühlbacher 2008] were conducted with the goal to determine how and which information employees at Roche access.

Approximately 80 employees participated in the survey described in [Mühlbacher 2008], which targets the use of scientific information sources by scientists. The interviewees claim to use approximately 2 different sources for retrieving information. 67.50% additionally state to rely mainly on standard tools for retrieving information. In detail, 94% use Google, 71% use PubMed, 65% use MicroPatent and 41% use SciFinder. Even though the previously conducted log file analysis does not completely match the information sources mentioned by Mühlbacher's participants, we can at least see a similar tendency in our results: A few search engines are heavily used while the majorities are less used. Although the majority of the scientists claimed to use Google, 31% of them emphasized that they use it only for trivial searches. In fact, Google is felt as the least reliable tool regarding scientific information especially compared to PubMed and internally licensed databases. Nonetheless, the majority (75%) of the participants declared that they are in general satisfied with their search results in Google, as well as in literature databases.

Advanced search capabilities are only used by a minority of the interviewees. The Boolean operators AND / OR are quite important and frequently used by participants. The NOT operator on the other hand is considered irrelevant. Search refinement by author and year are also considered important. Refinements by article type, journal as well as sorting by year are used moderately.

Mühlbacher also asked the participants' assumption about the coverage of various search engines. Approximately 50% of the scientists believe that Google indexes the entire WWW – a wrong assumption. Regarding PubMed and other databases, the majority assumes that it indexes only a subset of the Web – a correct assumption. Interestingly, only one third of the participants combine a variety of search tools for their inquiries. This supports the previous result, that only a few search engines are used by scientists at the risk of getting only a partial coverage of the relevant information.

Mühlbacher's study is about search in scientific information sources. These are in particular freely accessible or licensed external sources. The study reported in

[Maßun 2008] on the other hand, targets mainly the internal sources. 15 employees participated in the questionnaire. Her main findings revolve around information access, storage, and search.

Many participants complained about information structure in the intranet. In detail, they agreed that too many repositories exist and that they are not sure where to store information. Further, they complained about deep folder hierarchies resulting in "lost" files. According to them, most information is stored in E-Mail, departmental file shares, or personal shares. Public folders, intranet web, local disk, document management systems, and SharePoint are barely used for content creation and storage. In fact, only 10% of the documents at Roche are stored in a document management system. In addition, participants complained that there is no overview of available tools and their functionality, making helpful tools barely known. In the same context some complained about many isolated software solutions which lack a common platform.

Regarding search for information, several participants mentioned that search performance is poor and that intranet search needs to be improved. In this context they also complained about the fact that search for synonyms is not supported. Further, data redundancy and the inability to express custom views on information objects were mentioned as problems.

5.2 Quality characteristics of a professional search tool

The results of the previous studies suggest that professional search has only little in common with general search. In fact, professional search differs in various aspects from public search. Based on the similarities as well as the differences derived from the studies, we hypothesized on several quality characteristics of professional search tools. The architecture of a professional search tool which incorporates the characteristics mentioned below is described in Chapter 6.

5.2.1 One single entry point

Why are *Google, Yahoo!* & co. so successful? A reason is their good coverage of a large part of the web. As a consequence, they perform well on databases like *PubMed*, public patent databases and *Wikipedia.* This binds users to such public search engines: *Google* & co. became for many a main entry point, even for specialized information. Public search engines can be compared to shopping malls that bring a variety of goods in a variety of shops under one roof, thus making shopping simple and a comfortable experience – especially for people with limited time. With public search engines, information seems just one mouse click away. Professional search does not differ in this respect from public search. Especially in the workplace, users rarely have time to gather information from different places. This is time consuming and error prone because integrating the gathered information is done with little, if at all, help from the systems.

5.2.2 Role-specific ranking

Public search engines reflect the *"wisdom of crowds"* [Surowiecki 2004]. This wisdom is relevant to professional search as well. In the introduction (Chapter 1.1) we

mentioned an exemplary search for *"polycythemia vera*" at *Google*. The top ranked web pages for that query are surely not irrelevant to researchers. However, depending on the specific interests of a researcher, other pages about *"polycythemia vera"* might deserve higher ranks. Thus, a professional search agent should have a ranking reflecting as well as possible professional interests.

It is worth stressing that, to some extent, this also applies to public search: Search engines like *Google* and *Yahoo!* have recognized it and offer so-called personalization services [Pitkow et al. 2002; Tachau 2007]. In specific areas, like book selling, personalization is an important success factor. *Amazon's* recommendations for example, based on prior buys and formerly visited offers, are very appreciated by its customers: it is convenient to receive "purchase recommendations" for newly published books.

However, there are two interesting differences in this respect between public and professional search. First, public search is about the *"wisdom of the crowd",* that is, what is generally known or acknowledged in a society. Professional search, in contrast, should reflect as much as possible the professional groups. Indeed, it is beneficial that professional users share the wisdom of their *"professional crowd":* The information relevant to an expert in the treatment of *"polycythemia vera"* is relevant to others as well. Furthermore, if in a company *"polycythemia vera"* points to *"department C",* then this department is relevant to a company's worker searching for *"polycythemia vera".* Second, while *personalization* in a strict sense might be appropriate for public search services, it is rarely appropriate for professional search. In addition, privacy considerations are important to the success of professional search engines. Thus, while personalization is often useful for public search, *adaptation* to a group is preferable for professional search. In the following, *adaptation* will be used in this sense, sometimes called *"group personalization".*

5.2.3 Guided navigation

In order to be effective, professionals need to know where information can be found and which software tools exist. According to the introduced studies though, this is exactly the area where they lack support. They loose track of previously stored information and they have difficulties to discover new information due to the vast amount of different tools and repositories available in the intranet.

While the previously proposed "one single entry point" would be already a great support to solve this issue, we believe that it will not suffice. Imagine a search engine which is capable of indexing the entire intranet, containing billions of data records. Further, imagine that the company has office sites all over the world. In effect, a searcher will be confronted with two striking problems. First, in a global company it is likely that queries produce myriads of heterogeneous answer pages, i.e. many files have a high similarity to the query. In effect, the ranking of results is turned ad absurdum. Second, different sites might use a different jargon. At Roche for instance, a drug might be named differently depending in which development stage it is. If the searcher is not aware about the different jargon he might not be able to find the appropriate answer page.

A promising way to deal with these issues is guided navigation (also called faceted navigation). In guided navigation, answer pages are sorted into predefined categories which can be used by the searcher to further drill down search results. This helps if a searcher receives too many heterogeneous answer pages, and it helps if he is not well aware about the targeted search domain

5.2.4 Exploit existing knowledge

The point of using search engines is to find information which is not necessarily well organized. As [Weinberger 2007] puts it, with search engines, everything can be classified in the *"miscellaneous"* category of traditional classification systems. Companies are especially prone to large *"miscellaneous"*, that is, non-catalogued data. Indeed, their primary concern is not to catalogue data. Thus, professional search should harness a company's electronic data, to exploit as much as possible the hidden *"company's wisdom".* This raises the question: What data is worth considering? Three answers can be given: Everything, ontologies, and logs.

Everything: A search agent can be deployed on all possible data – and it should. Indeed, there are no reasons to exclude any available data because all data might help uncover interesting relationships. Hence, a company's organizational structure, project relevant literature, etc. should all be incorporated in a professional search agent.

Ontologies: At the workplace, people use all sorts of ontology-like data: controlled vocabularies, taxonomies, thesauri and of course, more and more full-fledged ontologies modeling the knowledge of specific domains.

Logs: A rich source of knowledge is buried in the logs of software applications. Of particular importance are the search logs from which one can extract associations between queries and subsequently selected documents: *(query, selected document)* pairs. The query can be used as a tag assigned by the user to the selected document. In the following, such a tag is referred to as *"implicit tag".* Before selecting an item from a search agent's result set, users read the short items' description and select the item that seems most relevant to their query [Joachims et al. 2007]. Thus, *(query, selected document)* pairs can be expected to be of high quality. Moreover, this quality can be expected to grow as time passes. Indeed, search queries are in general not posed only once [Rose & Levinson 2004], for discovering unknown resources (informational queries), but also again and again for coming back to known resources (navigational queries).

5.2.5 Professional search beyond research and development

Arguably, professional search in research and development reflects well the needs of professional search in general. There is only a difference in degrees: degrees of specialization (work in research and development in general leads to a higher specialization than work elsewhere) and degree of data coverage (a scientist might need more data sources than a producing engineer). Thus, conceiving a professional

search agent, deploying and evaluating its performance in a controlled and systematic way seems to be a promising way to investigate professional search.

Chapter 6

An ontology-based information retrieval approach for professional search

"The most likely way for the world to be destroyed, most experts agree, is by accident. That's where we come in; we're computer professionals. We cause accidents."

Nathaniel Borenstein (1957 –)

6.1	Beyond traditional search for information	101
6.2	Ontologies	105
6.3	Assigning metadata to "unstructured" text documents	108
6.4	Search results adaptation	119
6.5	Weighting document relevance	124
6.6	Discussion and related work	129

Previously we identified that one single entry point, role-specific ranking of search results, guided navigation, and exploitation of existing knowledge are reasonable approaches to a professional search tool. This chapter presents an ontology-based information retrieval (OBIR) approach which meets the listed characteristics.

We begin this chapter by giving a brief overview of our concept for OBIR as well as the targeted information sources. Details about each part are then given in the subsequent chapters as follows. First, the ontology concepts are introduced because these play a central role for the entire study. Second, classification as a mean of automatically gathering metadata is discussed. In this context we compare the two paradigms knowledge engineering (KE) and machine learning (ML). Third, we present our KE procedure for achieving a role-based ranking of results. In addition, the incorporation of log-based feedback algorithms for improving the ranking of results is discussed. Last, we outline how the weighting of document relevance is computed.

The chapter ends with a brief discussion of the introduced concepts.

6.1 Beyond traditional search for information

Common search tools do not meet the requirements of professional searchers. They do not target all sources which are relevant for professionals, they do not offer role-based adaptation nor guided navigation, and finally they do not exploit existing in-house knowledge.

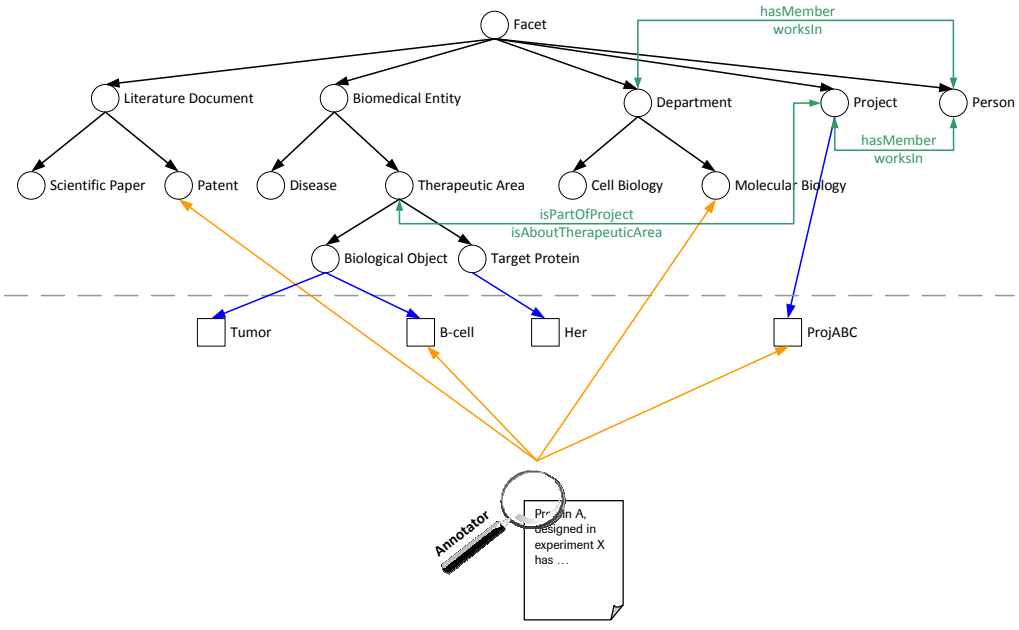
The incorporation of the listed features (Chapter 5.2) in a professional search tool bares the potential to resolve these issues. Especially the enrichment of existing data with metadata by means of semantic technologies could help to improve search and navigation. For instance, keyword-based search could be extended by concept-based search, and review of search results could be supported by pre-defined taxonomies, and so forth.

Next, we outline the corner stones of such an OBIR approach at a glance. Following that, we describe for which information sources our OBIR approach has been optimized, tested, and evaluated.

6.1.1 Key approaches

Two key characteristics of professional search are guided navigation and adaptation. Semantic technologies could enable a comprehensible navigation experience to the user as it could exploit and display all kind of relationships between the objects. Similarly, if semantic data about the searcher and the document corpus exists, adaptation of the result ranks would become possible. However, even though semantic technologies already exist since a couple of years now, keyword-based search is still the state-of-the-art in most search engines. The reason is that most information available is in form of unstructured text. An automatic conversion of unstructured text into a conceptualized ontology (Fig. 6-1) is a difficult task [Reeve & Han 2005]. Therefore, rather than conducting a complete conversion of unstructured text we merely do a partial conversion, i.e. we restrict to the extraction of a few business relevant concepts.

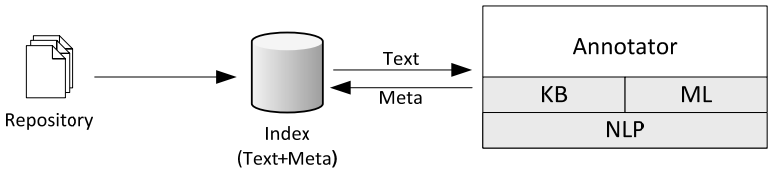
Fig. 6-1: Assigning metadata to “unstructured” text.



The conversion is done by means of an annotator which scans the document and decides based on pre-defined or learned rules to which concept a given text document belongs. The pre-defined topics and relations of the ontology are used to offer guided navigation (faceted navigation).

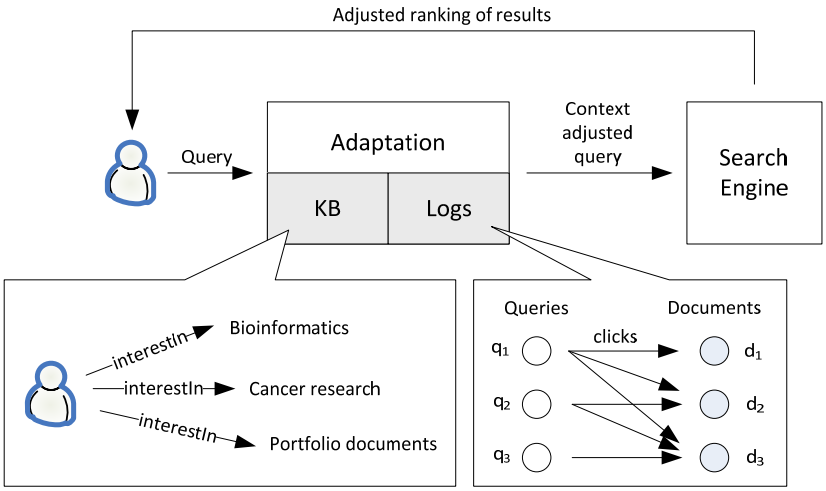
We use named entity recognition (NER) [Nadeau & Sekine 2007], machine learning, and knowledge engineering so as to partially convert unstructured text into structured metadata. Fig. 6-2 illustrates this process in context of a search index. In the first step, the data repository is crawled and the processed text is stored in the index. In the second step, the text index is scanned by an annotator which automatically extracts metadata. The extracted annotations are also stored in the index. Deploying the metadata together with the text into the index enables search by means of the information retrieval and the data retrieval paradigm (Chapter 2.5.3). The details of the annotator, i.e. the classification process, are described in Chapter 6.3. Notice that KE and ML are two distinct classification approaches. In KE an expert manually formulates the categorization rules while in ML the machine automatically learns how to categorize a document based on a pre-categorized training set. A performance comparison between the two is given in Chapter 8.2. There, we also show to which degree a classification improvement can be achieved by expanding the raw text with natural language processing (NLP) techniques.

Fig. 6-2: Converting "unstructured" text into structured metadata.



Another key feature of professional search is a role-based ranking of results. The adaptation process is described in Fig. 6-3. Basically, after a user has transmitted a query, the adaptation-module adjusts the query by boosting the searcher's context as well as by boosting the relevance of documents which are usually accessed in context of the query (Chapter 3.2). Finally, the search engine returns the adapted ranking of results.

Fig. 6-3: Role-based adaptation of the ranking of results.



As indicated beforehand, adaptation of search results is done by means of a knowledge base (KB) and a log-based method (Chapter 6.4). The KB contains preference rules, which are written by a knowledge-engineer. In particular, existing

administrative knowledge about an employee – such as his research area – is expressed by means of rules, resulting in statements like: "User A is interested in cancer research". The log-based approach on the other hand tracks the "queries-clicked document" pairs. This click-through history is used to boost the rank of documents which are often clicked in context of a given query.

Hence, the KB approach requires manual engineering of the preferences, while the log-based approach learns automatically the user's preferences by relying solely on past searches. At first sight, the log-based approach seems to be superior to the KB approach, because it constantly evolves in contrast to the static KB. However, one should consider that the KB does not suffer by the cold start problem, i.e. it works immediately while the log-based approach works only if sufficient log data is available. It is thus straightforward to do a linear combination of the rankings delivered by each method, effectively mitigating the disadvantages of each.

The last principle (one single entry point access to the information landscape) can be fulfilled in two ways. The first approach is to build a search engine covering all data of concern – and to offer only this search agent. Clearly, this is rarely feasible. Why not use specialized search agents that already exist, like search engines for technical literature or other fields? The second approach is that of *federated search* also known as *meta search engines* [Selberg & Etzioni 1997]. A meta search engine forwards queries to several search engines and aggregates their answers. This aggregation can be done in two ways: First by merging lists of answers relying on more or less sophisticated ranking functions or by presenting side by side the answers of the different search engines. Meta search engines have several advantages: They exploit existing search engines and they can relatively easily accommodate new ones; as a consequence, they hold the promise of a good coverage of the information space.

Therefore, our approach will follow the "meta search" approach using a side by side representation. By combining this with semantic annotations, guided navigation and adapted ranking of results a promising concept for professional search is achieved.

6.1.2 Targeted sources

Based on the studies conducted in the investigated Roche departments and on the feasibility of integrating the sources, we decided to consider the following information sources to study the validity of the hypothesized principles for professional search (Chapter 5; Table 6-1): PRPZ-Share (the file share of the pharmaceutical research area in Penzberg), PRPZ-WebSite (the web site of the pharmaceutical research area in Penzberg), several in-house databases, TagIt (a Roche tagging application for internal and external data), PubMed, DMOZ, and Wikipedia.

We are aware that this selection is far from complete. Indeed, many internal sources like intranet web portals, shares, and databases from other areas are missing. However, we argue that the selection suffices to investigate whether "one single entry point" is appropriate for professional search. Our selection contains a mix of

popular sources and less popular ones, so that we can test whether people prefer accessing the sources directly or via an integrated view.

Table 6-1: Overview of the examined features, the corresponding sources on which the features are evaluated, and the methods applied.

Feature	Evaluation on source(s)	Method
One single entry point	PRPZ-Share, PRPZ-WebSite, TagIt, PubMed, six in-house databases, Wikipedia, DMOZ	Meta-search using side-by-side presentation of search results
Adaptation of search results	PRPZ-Share, PRPZ-WebSite	Ontologies; Knowledge-based; Log-based
Guided Navigation	PRPZ-Share, PRPZ-WebSite, Application DB	Ontologies; Knowledge-engineering; Machine Learning; Natural Language Processing

The other approaches (adaptation and guided navigation) are only analyzed in context of the PRPZ-Share and PRPZ-WebSite. First, these are among the most frequently used sources of the investigated departments. Second, the demand to improve search on these sources is high. Third, enriching external sources with metadata is more difficult due to the heterogeneity of the professional jargon used as well as the inability to easily map to in-house databases. As a consequence classification and adaptation are also more challenging. Therefore, we investigate metadata extraction, classification, and adaptation only in context of the mentioned in-house sources.

6.2 Ontologies

In total, three ontologies are used to conceptualize the professional information space as well as the knowledge about it: *classification*, *annotation*, and *adaptation*. The classification ontology describes the overall topic of a text document while the annotation ontology is used to capture various concepts appearing within a text document. The adaptation ontology models a user's interests, i.e. it describes which topics a user is interested in and which not. In effect, ranking of results is adjusted according to the searcher's role in the company.

6.2.1 Classification ontology

The classification ontology models text documents and their properties. The central class of the classification ontology is thus *Document* (Fig. 6-4). A *Document* is identified by its URI. The URI also serves as a pointer to the document's location. Because we are in a corporate environment, we also included access control lists (ACLs; expressing access rights) in the model. The *hasACL* property thus tells which groups are allowed to view the document. Furthermore, a document is described by means of the classes *Entity*, *Project*, *Department*, and *Classification*.

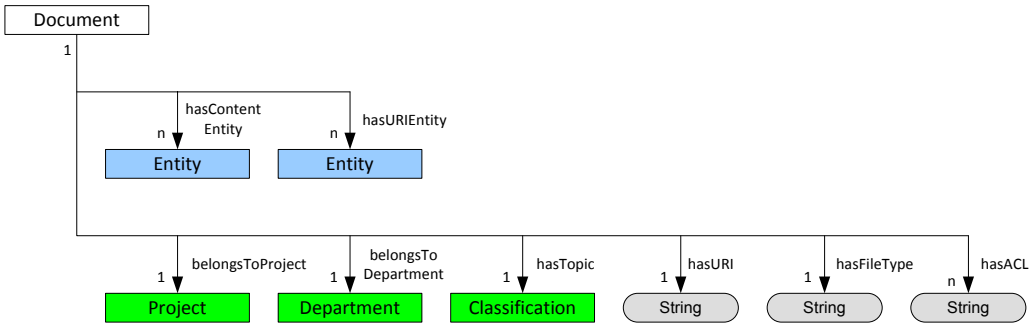
The class *Entity* captures concepts (like person, organization, date, etc.), which occur within a document's full text content or within a document's URI. Because this class is imported from the annotation ontology, we do not discuss it here but in the following section.

The class *Project* as well as *Department* are introduced to reflect a document's organizational embedding. In a research environment e.g., documents are often written as part of a project. Therefore, a document can be set in relation to a project by using the *hasProject* property. Similarly, the fact that a document is created in a certain department can be expressed by the *belongsToDepartment* property.

The class *Classification* is applied to specify the topic of a document. The class has various pre-defined subclasses into which a document can be categorized. An overview of the topics is given in Chapter 6.3.1.

Arguably, the modeled classes and their relations to documents are rather generic. Projects as well as departments are encountered in every larger institution. Further, a document's content is not arbitrary but about a certain topic. Variations occur merely on the instance level – different institutions have different projects, departments, and document topics.

Fig. 6-4: Excerpt of the classification ontology. Classes in gray are filled when reading the document from the source. The blue relationships are determined by means of named entity recognition. The green relationships are determined by reasoning or by machine learning.



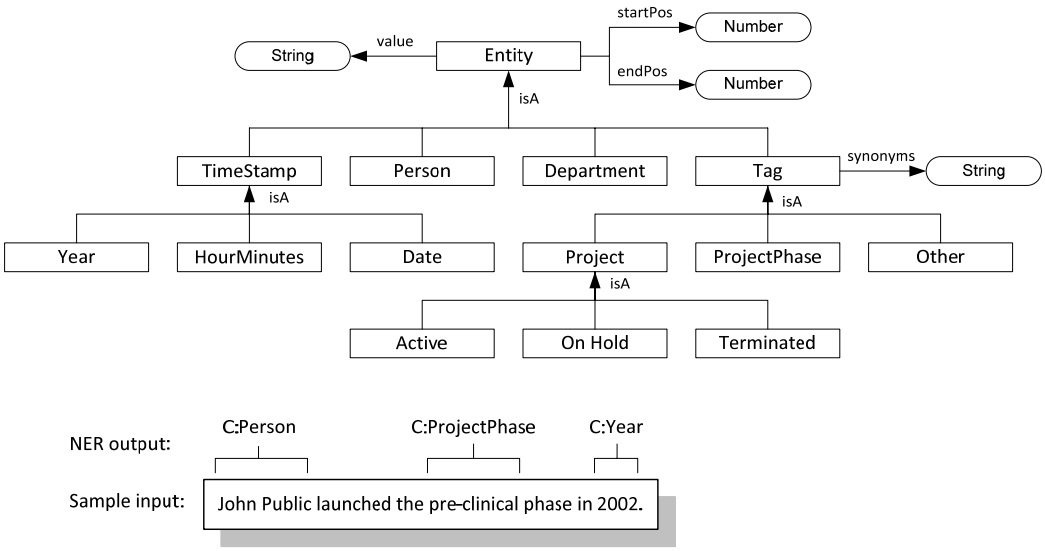
The *belongsToProject* and *belongsToDepartment* relations are determined solely by means of rule-based reasoning. The *hasTopic* property of a document on the other hand is determined either by means of KE or ML. The details of each classification approach are discussed in Chapter 6.3.

Wrapping it up, the purpose of the classification ontology is to capture additional metadata of a document.

6.2.2 Annotation ontology

The annotation ontology (Fig. 6-5) provides a structure for describing and capturing named entities occurring within unstructured text. A named entity represents a pre-defined category (such as persons, organizations, etc.) into which atomic elements of text are classified. The automatic classification of text elements into pre-defined categories is denoted as *named entity recognition* (NER). Before describing the applied NER pipeline in Chapter 6.3.3, we continue with the description of the ontology and the entity types to be recognized.

Fig. 6-5: Excerpt of the annotation ontology, displaying classes and their relationships. The given entity types are used as features to annotate whole documents.



The main class of the ontology is *Entity* and it has three important properties: *startPos*, *endPos*, and *value*. The parameter *startPos* is a numeric value which marks the first character position of the classified text element. The parameter *endPos* marks in just the same manner the last character position. The parameter *value* simply stores the classified text elements as a string. Having defined these attributes, we are now able to store the entity annotations of any text and their position in the original text document.

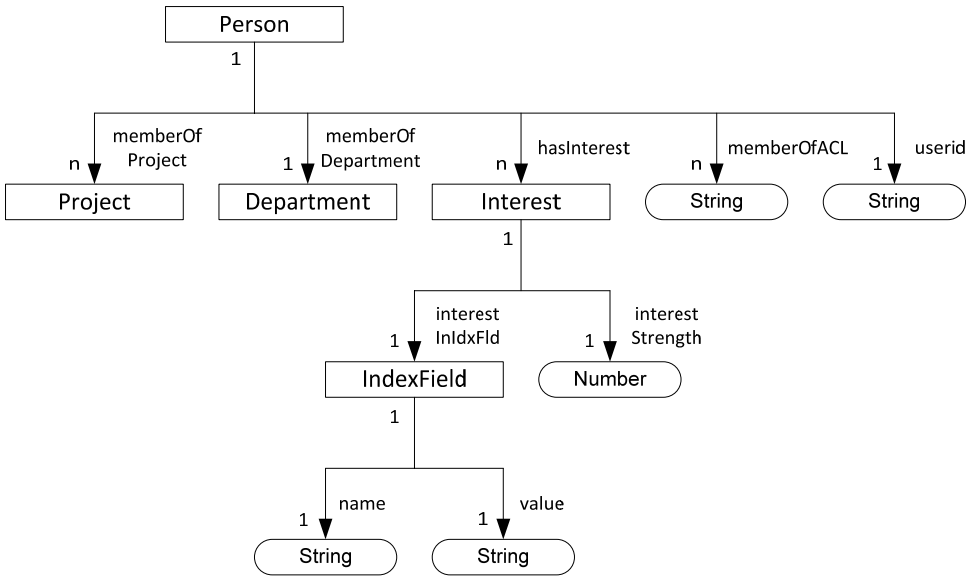
The named entities are used as features for the classification of whole documents into pre-defined topics. In addition the extracted entities could be used to improve guided navigation. Determining which entities are relevant and which not depends on the considered domain. In our case we distinguish the following types of entities: *TimeStamp*, *Person*, *Department* and *Tag* – all descendants of the class *Entity*. The class *TimeStamp* represents a temporal entity and it is further subdivided into *Year*, *HourMinutes*, and *Date*. By *Year* we understand any four digit number. *HourMinutes* represents the time of a day, and *Date* refers to a text snippet consisting of a year, a month and a day. The class *Person* refers – as the name suggests – to a person. The class *Department* represents a department or group of the organization. Arguably, these are quite generic entity types which might also be applied in other environments.

The last entity type we consider is *Tag*. The class *Tag* does not represent a traditional concept; rather it is a collection for various concepts identified by one ore more words. Therefore, a text passage is annotated with the type *Tag* if it matches the name of the predefined *Tag* instances. For example, *ProjectPhase* – a subclass of *Tag* – has among others the predefined instance *Pre-Clinical*. Thus, in Fig. 6-5 the word "pre-clinical" is annotated as a *ProjectPhase* entity. Additionally a *Tag* can have a set of synonyms. The instance *Pre-Clinical* for example has the synonyms "P0" and "Phase 0" – these are common abbreviations in the medical sector.

6.2.3 Adaptation ontology

The adaptation ontology encapsulates the user model (Chapter 3.1). The model describes the characteristics of a searcher and it is applied to adjust the ranking of search results. An illustration of the classes and properties used in the user model is given in Fig. 6-6.

Fig. 6-6: Excerpt of the adaptation ontology.



The class *Person* represents a user and his properties. A user is characterized by his identity (userid), the projects he is involved, his departments, and his access control list memberships. Given these facts, reasoning is applied to deduce the person's interests into specific document properties, i.e. index fields. The class *IndexField* represents the link between the ontology and the index fields of the search engine. In fact, the index fields cover all *Document* properties of the classification ontology. Therefore, we are able to model a searcher's interests in projects, departments, and specific document topics. The interests in an index field together with the interest strength are used to adjust the ranking of results accordingly. For instance we could formulate a rule stating that a searcher is mostly interested in the "Bioinformatics" area. As a consequence, for any search, documents belonging to this topic will be ranked higher. Chapter 6.4 outlines the rule-based adaptation approach in detail.

6.3 Assigning metadata to "unstructured" text documents

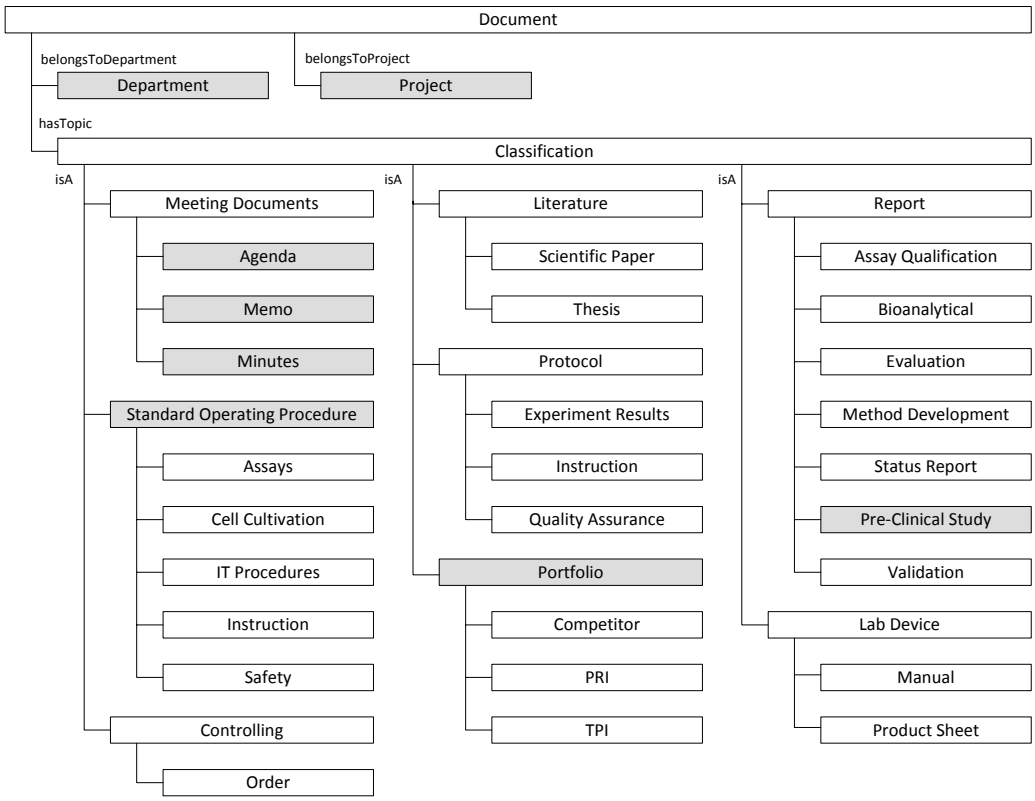
In this section we describe the process by which unstructured text is enriched with metadata. We examine the assignment of metadata to unstructured text by means of two paradigms: knowledge engineering (KE) and machine learning (ML). In KE, a set of rules is defined which encodes an expert's knowledge on how to classify a document into a given category. In ML, an inductive process automatically builds a classifier by learning from a set of pre-classified documents [Mitchell 1997; Sebastiani 2002]. The reader might wonder why we evaluate a KE approach considering the fact that it is barely used since the ML paradigm has become popular in the early nineties. The main reason is the availability of semantic technologies. It would be convenient to apply rule languages of semantic frameworks for

categorization purposes. Anyone could see and adjust the classification rules if necessary. Further, users would be able to understand why a document was categorized into a certain topic due to the explicit nature of rules. Beside this, we are interested to determine if the approach is feasible and at which costs a KE approach can provide better classification rules than ML. In this context we are also interested to find out which method to use for which type of document. The task of assigning metadata to "unstructured" text is conducted for the ~600k files located on the PRPZ-Share and PRPZ-WebSite.

6.3.1 Considered document classes

In a thorough analysis of the document corpus we identified several types of documents and organized them in a taxonomy (Fig. 6-7). Notice that this taxonomy is not complete. It merely represents the most common types. For the proof of concept we decided to reduce the identified classes to a small subset which we assume to be the most relevant classes for the professional users in the considered Roche departments. The subset reads as follows: Department, Project, Agenda, Memo, Minutes, Standard Operating Procedure, Portfolio, and Pre-Clinical Study. Next, we give an informal definition of each class of the final subset.

Fig. 6-7: Excerpt of the classification ontology. The grayed classes are detected by the classifier, while white classes are not considered in context of this thesis.



Department: The department which published the document.

Project: The project to which the document is highly related.

Agenda: A meeting agenda is a list of activities in the order in which they are to be taken up, beginning with the call to order and ending with adjournment. It usually includes one or more specific items of business to be considered. It may, but is not required to, include specific times for one or more activities.

Memo: A short note (a memorandum) about something, for the record.

Minutes: Minutes are the instant written record of a meeting or hearing. They often give an overview of the structure of the meeting, starting with a list of those present, a statement of the various issues before the participants, and each of their responses thereto.

Standard Operating Procedure (SOP): A set of instructions constituting a directive, covering those features of operations which lend themselves to a definite, step-by-step process of accomplishment.

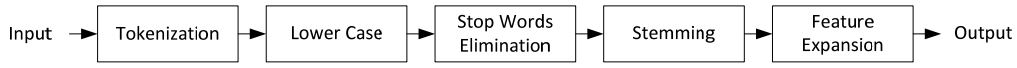
Portfolio: A collection of investments (projects) all owned by the same individual or organization.

Pre-Clinical Study: A pre-clinical study report about a Roche compound having a registered Roche number.

6.3.2 Text representation

Prior to discussing details of the KB and ML approach for text classification, we outline how text will be represented. Text representation – as part of the IR process – was already covered in Chapter 2. In content categorization, text is represented in just the same manner, namely as a word vector with specific weights. IR and text categorization share also many common steps in the text processing pipeline which is used to obtain the final word vector. Passing an input text through a chain of different analyzers (Fig. 6–8) reduces complexity and redundancy. In effect, classification performance can be improved in terms of precision, recall, and time.

Fig. 6-8: Text processing pipeline.



The initial step is the tokenization of sentences into single terms. Applying a lower case filter and a stop word filter next, reduces complexity while loosing almost no relevant information. Stemming can further reduce word density – at the cost of losing potential relevant information. The last step, feature expansion, is the only step which adds additional information to the input data. In our case we apply NER as the method for feature expansion, i.e. we determine the concepts of the tokens.

Having applied the filters, the next question is how to weight the output, i.e. the word vector. Again, the same measures as in the case of IR can be applied. We could thus use a Boolean metric, a tf measure, or any tf*idf measure (Chapter 2.2.3).

Notice that our text representation (choice of filters / term weighting) is different for the KB and ML method (cf. Chapter 6.3.4 and Chapter 6.3.6).

Next, we briefly outline how NER performs the feature expansion.

6.3.3 Named Entity Recognition

Initially we considered the usage of the Stanford Named Entity Recognizer[15] (based on ML) for detecting entities of the type *Person*. However, we were not satisfied with the precision and recall levels, given that we invested a considerable amount of effort in providing large quantities of training data (hundreds of manually annotated documents) to the Stanford NER algorithm. Therefore, we decided to switch to a list based approach. The person list is obtained by dumping all employee names (about 10,000 names) from the administrative database. Given the list, the annotator scans the text in a case-insensitive mode and marks recognized elements accordingly. A minor difficulty is that there are syntactical variations in the way names are abbreviated and in the order of surname and forename. The name "John Public" for instance, might be written as "J. Public", "Public, John", "Public, J.", etc. Therefore, we perform a scan for all relevant syntactical variants.

The entities of type *Tag* are processed in a similar fashion except that syntactic variants are explicitly captured in the synonym attribute of the class *Tag*.

Annotation of the entity *Department* is also based on a vocabulary list. The department names are extracted from the organizational chart which is encoded in an ontology. The ontology not only contains the department names but also their hierarchical organization. We can thus obtain the super-departments using the implicit is-a relationship. In contrast to the previous approaches, annotation does not target the full-text content but a file's security settings (Chapter 6.4.1) and URI. From our experience at Roche, at least one of them can be mapped well to the department names. Mapping is done using a combination of regular expressions (which might differ from one division to the next) and string matching. In case of the investigated department, the departments are extracted from the folder names. Hence, the annotator would extract the department "TR-IB" from the following URI "//SERVER/TR-IB organization/misc.doc". Using the is-a relationship of the ontology we additionally obtain the departments "TR-I" and "TR".

A different approach is taken for the entities inheriting from *TimeStamp*. Here, regular expressions are used to describe the patterns of *Date*, *Year* and *HourMinutes*. The regular expressions for detecting these are given next.

```
# A list with month names in English and German language
longMonthNames =
    "(January|February|March|April|June|July|August|" +
    "September|October|November|December|Januar|Februar|März" +
    "|April|Juni|Juli|August|September|Oktober|November|Dezember)"
```

[15] http://nlp.stanford.edu/software/CRF-NER.shtml

```
# A list with abbreviated month names in English and German language
shortMonthNames =
    "(Jan|Feb|Mar|Mär|Apr|May|Mai|Jun|Jul|Aug|Sep|Sept|Oct|Okt|Nov|Dec |Dez)"

# Date: Pattern matching numerical dates, e.g. 21.1.81
    "\\b(^|\\s)([1-9]|0[1-9]|[12][0-9]|3[01])\\." +
    "([1-9]|0[1-9]|1[012])\\." +
    "((19|20)\\d\\d|\\d\\d)\\b"

# Date: Pattern matching numerical dates, e.g. 1981/01/21
    "\\b(19|20)\\d\\d[- /.]" +
    "(0[1-9]|1[012])[- /.]" +
    "(0[1-9]|[12][0-9]|3[01])\\b"

# Date: Pattern matching medium length dates, e.g. 21. Jan. 81
    "\\b(0[1-9]|[12][0-9]|3[01])[\\.,]?\\s{0,5}" +
    "(" + shortMonthNames + "\\.?|" + longMonthNames + "),?\\s{0,5}" +
    "((19|20)\\d\\d|\\d\\d)\\b"

# Date: Pattern matching medium length dates, e.g. Jan 21 1981
    "\\b(" + shortMonthNames + "\\.?|" + longMonthNames + "),?\\s{0,5}" +
    "(0[1-9]|[12][0-9]|3[01])([\\.,]\\s{0,5}|\\s{1,5})" +
    "((19|20)\\d\\d|\\d\\d)\\b"

# Date: Pattern matching long date formats, e.g. January 21st 1981
    "\\b(" + longMonthNames + "),?\\s{0,5}" +
    "((1|21|31)st|(2|22)nd|(3,23)nd|(2?[4-9]|1[0-9]|30)th)(,\\s{0,5}|\\s{1,5})" +
    "((19|20)\\d\\d|\\d\\d)?\\b"

# Year: Pattern matching a numerical year, e.g. 1981
    "(19|20)\\d\\d"

# HourMinutes: Pattern matching a day time, e.g. 5.00 a.m.
    "(^|\\s)((?s)\\b([0-2]?\\d[:\\.][0-5]\\d\\s*" +
    "(AM\\W|A.M.\\W|PM\\W|P.M.\\W|am\\W|a.m.\\W|pm\\W|p.m.\\W)?)" +
    "(\\p{Punct}\\s|\\s|$))|" +
    "((?s)\\b([0-2]?\\d\\s*" +
    "(AM\\W|A.M.\\W|PM\\W|P.M.\\W|am\\W|a.m.\\W|pm\\W|p.m.\\W)" +
    "(\\p{Punct}\\s|\\s|$)))"
```

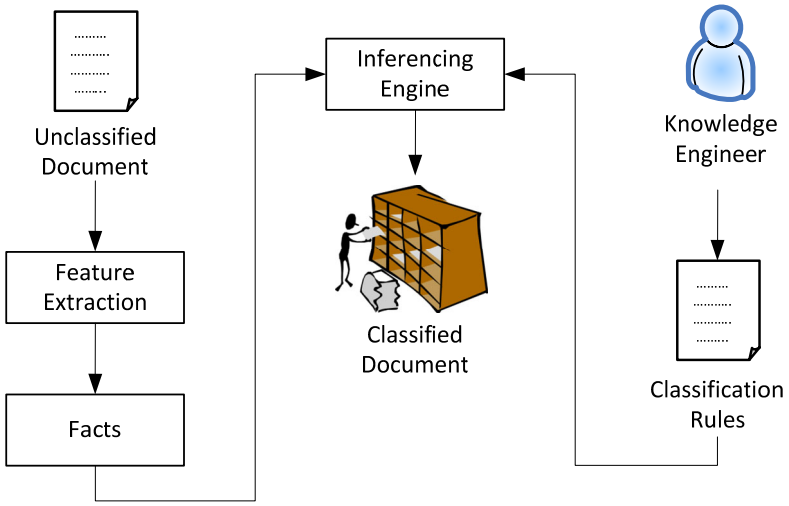
6.3.4 Classification using a knowledge-engineering approach

In a knowledge-engineering classification approach (Fig. 6-9) the first step is to create a taxonomy into which the documents are classified. Then, a knowledge engineer formulates rules based on the words which typically occur in the corresponding document category. For instance, a domain expert might define a simple rule for classifying literature documents such as: If the words “abstract”, “references”, and “keywords” appear in the text, then classify the input as a literature document. While this example is quite trivial it illustrates the basic approach in which the domain expert encodes his knowledge into classification rules. In reality, more words as well as their order should be considered.

Having defined the topic taxonomy and the rules, the next step revolves around processing the input document. Computational time of reasoning can get significant

if huge amounts of facts have to be considered. Therefore, we consider only the extracted features of the text and their position in the original text. Working with features not only reduces the number of terms but also the amount of rules. For instance, by referring to the entity *Year* in a rule we save the work of specifying all kind of variants like 2008, 2009, etc.

Fig. 6-9: KB classification pipeline.



One should keep in mind that F-Logic does not support fuzzy rules, i.e. a document belongs to a topic or it does not – there is no notion of maybe. Of course one could simulate this behavior by wrapping the rules in a meta-interpreter supporting the association of certainty factors to rules and facts [Schocken & Finin 1987]. However, this would add an additional layer of complexity. On the one hand, knowledge engineers must additionally define certainty factors to their rules, making modeling more difficult than it already is. On the other hand, computational time would get considerable if the rules are wrapped in a meta-interpreter. Moreover, the evaluation in Chapter 8 suggests that the added complexity is not worth the expected performance boost, and probably would neither be necessary in many other professional contexts as well.

Last, we want to outline, that despite the restriction to extracted features, the amount can get very large (over 10,000 entities) for a large document. To circumvent this, we restrict per default to the first 10,000 facts. This number is not arbitrary but was determined by examining the present corpus. A key observation was that the first three pages suffice to categorize a document. Thus, we determined the average number of words which could be observed on the first three pages and then defined an upper bound. Of course, in case the key characteristics of a document are not at the beginning but in the middle or towards the end of the text, then this heuristic can not be applied anymore.

6.3.5 Classification rules

In this chapter we present the classification rules for the topics *Department*, *Project*, *Agenda*, and *Portfolio* (Fig. 6-7). The rules for *Minutes* and *Memo* documents are omitted as these are very similar to the rules for classifying *Agenda* documents. The

categories *SOP* and *Pre-Clinical Study* are not classified by means of rules due to the complexity of finding a common set of terms with a high enough discriminative power. Hence, these are merely classified by means of ML as described in the subsequent section. Prior to the definition of the classification rules, we discuss four auxiliary rules.

The rules for deducing the properties *belongsToProject* and *belongsToDepartment* of a *Document* (cf. Fig. 6-4) are simple auxiliary rules. They just map the according *Entity* from the relationship *hasURIEntity* so that access is eased. The only restriction is to select the project or department from the deepest level. The reason behind this heuristic is the assumption that the deepest folder has the highest association with a file's content. Because of their simplicity we omit these rules and continue with the text categorization rules.

The third auxiliary rule, denoted "foundParticipants", is used to detect so called participants – these are text passages in which the word "participant" (or its synonyms "attendee", "presenters", etc.) is closely followed by a person name.

Rule: foundParticipants

```
FORALL aDocument, anEntity, Type, S, E, M
    aDocument[#hasParticipants->true]
    ←
    aDocument:#Document AND
    aDocument[#hasEntity->>anEntity] AND
        (anEntity:Type
            OR
        (unify(anEntity, #Participants) and unify(Type, #Participants))) AND
    #Participants:#Other AND
    anEntity[#startPos->S;#endPos->E] AND
    matchEntityPattern(aDocument, anEntity, Type, S, E,
        sequence(#Participants,15,#Person), M).
```

The rule foundParticipants uses two built-in functions named *matchEntityPattern* and *sequence* to detect whether the entity *Participant* is in a maximal distance of 15 characters to the entity *Person*. If this is the case, the attribute *hasParticipants* of the document is set to true. Ideally, the character distance should be only 1. However, the conversion of the various document formats to flat text produces sometimes special characters which can't be excluded from the stream. Therefore, the distance was set to 15 characters.

The fourth auxiliary rule, denoted "foundGroupOfPersons", detects group of persons, i.e. text snippets in which several person names appear. This rule makes use of the same built-in functions as the previous one. It sets the attribute *hasGroupOfPersons* to true, if five entities of type *Person* occur within a subsequent distance of 150 characters. Again, this value should be 1. The distance had to be increased due to the conversion of tables into flat text. In case person names occur below each other in a table, it doesn't imply that they will be also next to each other in the flat text. Indeed, the conversion of tables to flat text is problematic, because

the cell order is sometimes broken. Therefore, we had to increase the distance to 150 characters, so that we cover these problematic situations.

Rule: foundGroupOfPersons

```
FORALL aDocument, anEntity, Type, S, E, M
    aDocument[#hasGroupOfPersons->true]
    ←
    aDocument:#Document AND
    aDocument[#hasEntity->>anEntity] AND
    anEntity: Type AND
    anEntity[#startPos->S;#endPos->E] AND
    matchEntityPattern(aDocument, anEntity, Type, S, E,
        sequence(#Person,150,#Person,150,#Person,150,#Person,150,#Person), M).
```

Another auxiliary rule is "foundEventTime". This rule sets the property *hasEventTime* of the document to true, if a snippet is found where the token "begin" or "start" is immediately followed by an entity of the type *HourMinutes*. The F-Logic version of the rule is omitted because it is very similar to the foundParticipants rule.

Having defined these auxiliary rules, we are now able to list the first classification rule, denoted classificationAgend, which is used to classify agenda documents.

Rule: classificationAgenda

```
FORALL anEntity, S, aDocument
    aDocument[#hasTopic->>#Agenda]
    ←
    aDocument:#Document AND
    aDocument[#hasEntity->>anEntity] AND
    aDocument[#fileType->>".doc"] AND
        (aDocument[#hasParticipants->>true]
            OR
        aDocument[#hasGroupOfPersons->>true]) AND
    aDocument[#hasEventTime->>true] AND
    #Agenda:#Classification AND
    anEntity:#Entity AND
    unify(anEntity,#Agenda) AND
    #Agenda:#Other AND
    #Agenda[#startPos->>S] AND
    less(S,250.0).
```

A *Document* is classified into the category *Agenda*, if the file extension is "doc", and if the tag-entity *Agenda* occurs within the first 250 characters of the text and if an event time has been found. Further the document must have either participants or a group of persons. Putting it less formal, an agenda document is a document which contains the word "agenda" on the first lines, a list of participants and a time on which the meeting occurs. Finally, the file format must be MS Word. The limit of 250 characters was determined by observing several agenda documents. By doing so we noticed that the word "agenda" usually occurs at the beginning.

The rule for classifying documents into the *Portfolio* category targets only PowerPoint files. Further, these must contain at least one tag-entity *Project*. Other

requirements are that an entity *Year* occurs, and that all *ProjectPhase* entities appear. Notice, that the detection of a Project as well as assuring that all project phases have been found is done by auxiliary rules which are omitted here.

Rule: classificationPortfolio

```
FORALL aPhase, anEntity, S, aDocument, aYear, aProj, nrPhases
    aDocument[#classification->>#Portfolio]
    ←
    aDocument:#Document AND
    aDocument[#hasEntity->>anEntity] AND
    aDocument[#fileType->>".ppt"] AND
    aDocument[#foundProject->>aProj] AND
    aDocument[#countPhases->>nrPhases] AND
    #Portfolio:#Classification AND
    anEntity:#Entity AND
    unify(anEntity,aYear) AND
    aYear:#Year AND
    aYear[#startPos->>S] AND
    aPhase:#Phase AND
    xcount(a,aPhase,aPhase,nrPhases).
```

6.3.6 Classification using Machine Learning

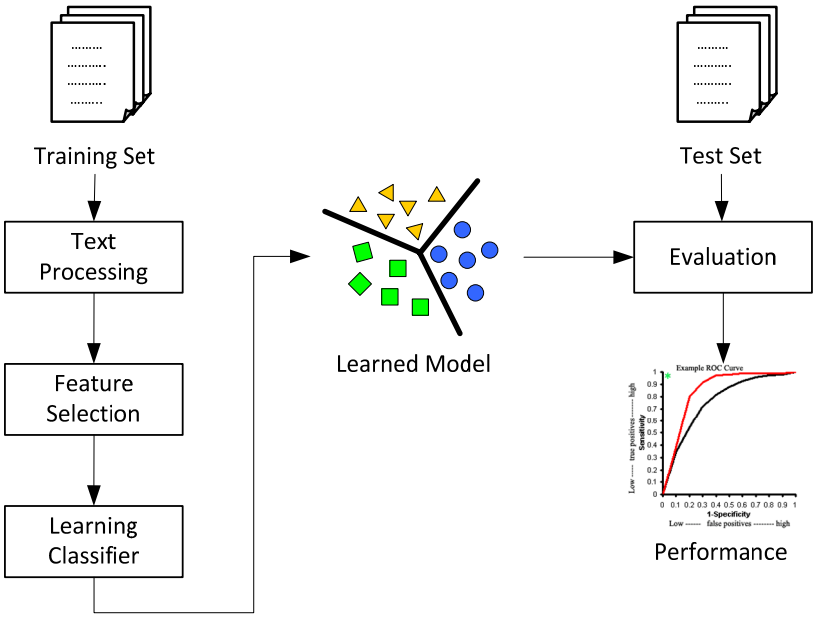
Machine learning is about algorithms that automatically improve their performance (i.e. their ability to predict) with experience. In ML two learning paradigms can be distinguished: supervised learning and unsupervised learning. Supervised learning algorithms are able to generalize from labeled examples and to identify core characteristics. Unsupervised learning methods do not need any examples, i.e. they decide autonomously which objects should be grouped together. Text categorization is a supervised learning task [Sebastiani 2002]. Consequently, the learning success strongly depends on the quality of the provided examples, i.e. is the data representative, is it complete, etc.

The learning algorithms work on a fixed set of features (attributes). Therefore, the first step, after a training set has been defined, is to process the text accordingly (Fig. 6-10). We apply the text analysis pipeline given in Chapter 6.3.2. The word vector weights are calculated by one of the following approaches: Boolean, tf, and tf*idf. An evaluation of which method works best is given in Chapter 8.2.

The following “feature selection” step can be applied to reduce the dimensionality of the data. Doing so results in less data needed for the learning process, it avoids overfitting for conventional learning methods, and reduces execution time of the learning algorithm, while the classification accuracy remains approximately the same. The task is thus to identify the most useful features for learning, i.e. to remove irrelevant and redundant features. There are two common approaches for feature selection: wrappers and filters. Wrappers evaluate the worth of feature sets by applying the learning algorithm. In effect they are optimized for the particular learning algorithm and are therefore less general, i.e. if other learning algorithms are to be used, the wrapper must be re-run. Filters on the other hand evaluate the worth of feature sets by using heuristics. In contrast to wrappers they execute much faster.

Overviews and surveys of feature selection methods are given in [Brank & Grobelnik 2002; Joachims et al. 1998; Rogati 2002; Sebastiani 2002].

Fig. 6-10: ML pipeline for learning a model.

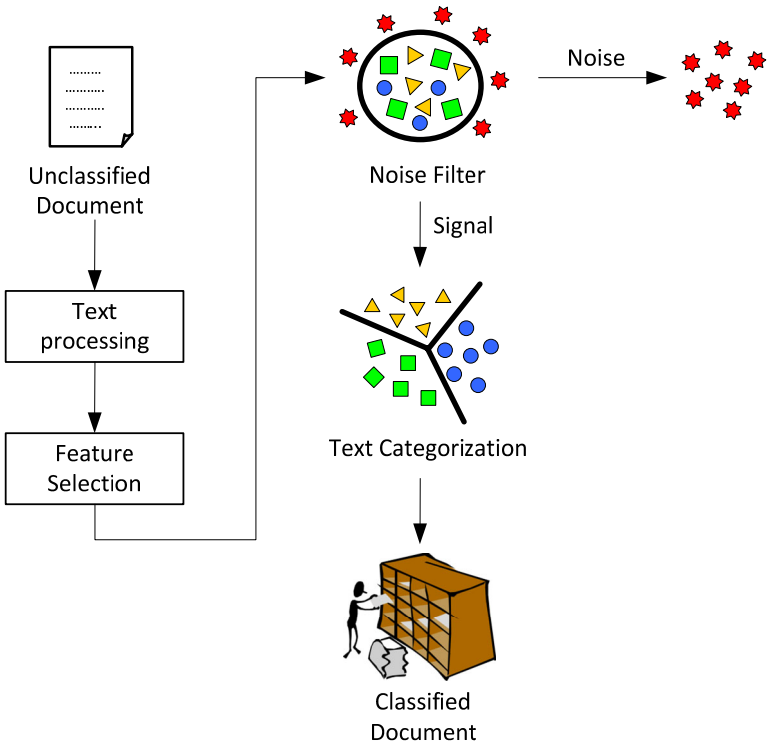


In the next step, the features are processed by a learning algorithm, which effectively creates concept descriptions (knowledge) from the training data. Nowadays, many different classifiers exist: Naïve Bayes [Domingos & Pazzani 1997], decision tree classifiers [Safavian & Landgrebe 1991], decision rule classifiers [Cohen & Singer 1999], Bayesian networks [Heckerman 1999], nearest neighbor [Dasarathy 1990], support vector machines (SVM) [Cortes & Vapnik 1995], etc. In literature, SVM are considered as the state-of-the-art approach for classifying text documents [Joachims et al. 1998]. The reason is that SVM can handle well a high dimensional (many words) and sparse input space (document vectors contain only few entries which are non-zero) as is the case in text classification. Therefore, we decided to use SVM with a linear kernel (i.e. $K(x,y)=<x,y>$) for the learning task. Testing is done using a 10-fold cross validation. The learned model is then evaluated in terms of precision and recall by means of a separate test-set.

Choosing SVM for learning also influences the previous feature selection step. Interestingly, the study given in [Joachims et al. 1998] reports that SVM are very robust even in the presence of numerous features. Another result of the study is that the majority of features are useful for text classification. The studies reported by [Brank & Grobelnik 2002; Rogati 2002] even show, that feature selection has no improvement or even degrades SVM performance. Therefore, it is disputed whether to use feature selection methods (such as chi-square) in combination with SVM or not. We decided to skip the feature selection step. Nevertheless, we still restrict the total number of words per class to the 1,000 most frequent words. We did not see the requirement to increase the number of features beyond 1,000 per class because classification performance did not increase significantly by doing so.

Our classification pipeline (Fig. 6-11) begins with the text processing and feature selection of the unclassified document. Then, two classification models are applied linearly. The first model separates noise (documents which do not belong to any of the pre-defined categories) from signal. The second model categorizes the signal into the learned classes. Using two models reduces the complexity for the learner. Further, they can be fine-tuned for their specific task.

Fig. 6-11: Classification pipeline.



In Chapter 8.2, we give a detailed evaluation of our ML pipeline for the investigated dataset. There, we give the size of the used training and test set. Further, we compare which document weighting schema gives the best results and the impact of named entity expansion on precision and recall. Last, we compare the performance of ML to that of KE.

6.3.7 Discussion

In this sub-chapter we discussed the problem of assigning metadata to unstructured text. Because manual annotations do not scale well with large datasets we decided to adopt an automatic approach. In this connection, we compare the KE and ML paradigms for enriching unstructured documents with metadata. Both have their advantages and disadvantages. In the KB approach, a knowledge engineer has to manually codify the domain knowledge, while in the ML approach a training and test set needs to be specified.

Text categorization provides the foundation for faceted navigation. Further, it also enables search result adaptation (cf. next section).

6.4 Search results adaptation

Adaptation of search results is about adjusting the ranking of result items to the user's assumed interests (Chapter 3.2). In our concept we focus on adaptation methods ("role-specific ranking") which do not require the storage of any person related information. It is thus particularly suited for companies. Indeed, laws and the work council often prohibit the explicit storage and processing of person related data, as they could be misused to track an employee's activities.

We begin this sub-chapter by introducing our knowledge-based adaptation method, which adjusts ranking of search results to the user's working-context. The working-context is extracted form "explicit" data sources by means of rules (encoded in the adaptation ontology). Then, we give a brief prospect of advanced user roles. Following that, we outline the log-based adaptation method of choice, which fine-tunes the ranking of results. The method uses anonymous "implicit" click-through data which is gathered by monitoring the users' search activities. In the last section, we discuss how knowledge-based and log-based adaptation could be aggregated so that we obtain self-adaptive "roles" shared by several individuals.

The details of how the inferred user interests as well as the interest strength are incorporated into a search engine are not given here. Rather, this topic is discussed in Chapter 6.5, which deals with the relevancy weighting of search results.

6.4.1 Knowledge-based adaptation

The knowledge-based adaptation method to be introduced is based on context evidence (Chapter 2.6). A key question is thus how to obtain context information?

The answer is "administrative databases", and especially the security groups they manage. The groups are used to control access to files, applications, etc. within the corporate network. Access control is typically enabled by matching the security groups of the source to the groups the employee is a member of. Security groups are of special value, because a) they are not created randomly but are usually well structured and b) security groups exist in every enterprise.

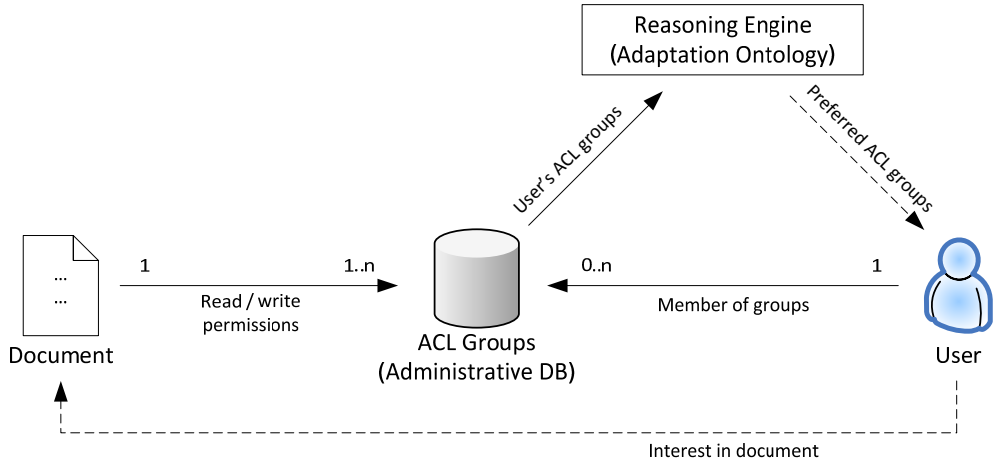
Access control lists (ACLs; i.e. security groups), are for instance created to hold all members of a department, a project or of an application. Knowing the semantics of the group's syntax is thus the key, to obtain the context of files as well as the working context of employees. Next, we illustrate the ACL syntax used at Roche.

```
# Structure encoding permissions to departmental data. Determined by the "_ORG_" infix.
<GLOBAL_AREA>_<LOCATION>_<AREA>_ORG_<DEPARTMENT>_<PERMISSION>

# Structure encoding permissions to project related data. Determined by the "_PRJ_" infix.
<GLOBAL_AREA>_<LOCATION>_<AREA>_PRJ_<PRJNAME>_<PERMISSION>

# The tags represent the following information
# <GLOBAL_AREA>: "Pharmaceutical Research" or "Diagnostics"
# <LOCATION>: The mnemonic of the city or site, e.g. "PZ" denoting Penzberg
# <AREA>: The mnemonic of the local area, e.g. "TR" denoting "Pharmaceutical Research",
```

<DEPARTMENT>: The mnemonic of the department, e.g. “TR-IB” denoting “Bioinformatics”
<PRJNAME>: The controlled name of the new medicine project, e.g. “Herceptin”
<PERMISSION>: “C” representing a change permission, or “R” denoting a read only permission

The knowledge about the ACLs’ semantics enables us to model the interest of a user into certain organizational areas (Fig. 6-12). We could thus state that users are interested in the projects in which they are involved, i.e. we would create a rule which filters the ACLs encoding an interest into a project. The reasoning engine would thus determine the preferred ACLs, which in effect are associated to a set of documents, of which we assume the user is interested in.

Fig. 6-12: Inferring a user’s assumed interests.



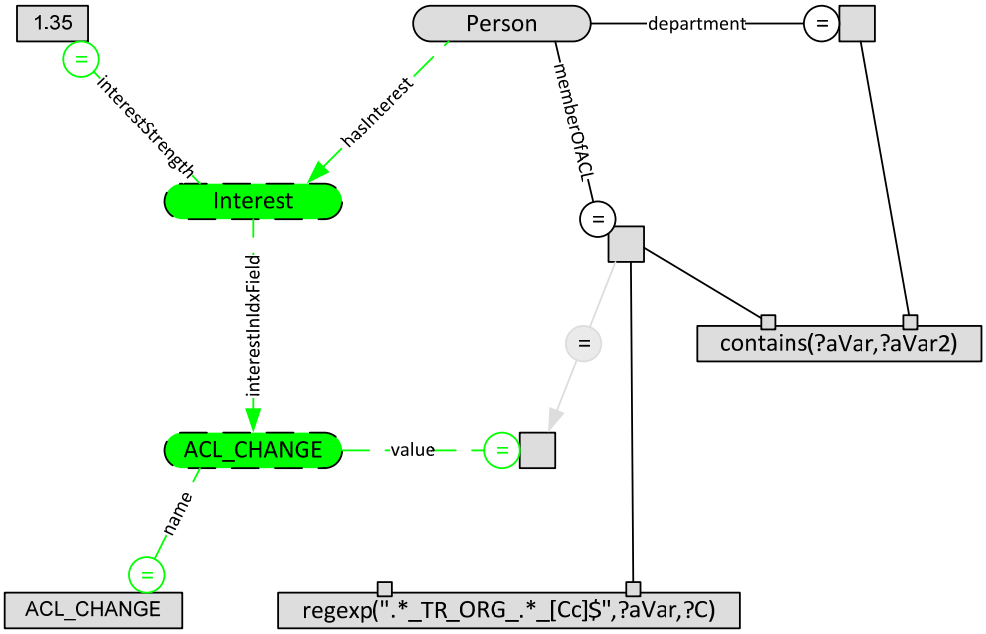
We capture the pre-defined user interests – i.e. the user model – in an ontology. The interests are expressed by means of rules which denote specific interest strength into files belonging to specific security groups. The default interest strength of a user is set to 1 – the neutral point of user interest. Therefore, a value below 1 indicates a disinterest and a value above 1 indicates an interest.

We define three different interest rules. The first rule models a user’s interest in files of the department he is working in. The second rule models a user’s interest in files of his division (e.g. Pharma Research or Pharma Technical Development). The third rule models a user’s interest into files belonging to projects he is involved in. Notice that in the rules, we explicitly refer to groups denoting change permissions. While a read permission can be assigned to numerous persons, change permissions are only given to the creators (i.e. responsible persons) of a file.

Instead of giving the F-Logic code of the rules, we illustrate the rules by means of a graph. In the text categorization chapter we had to use code instead of figures due to the difficulty of visualizing rules having OR connectives. An F-Logic graph is read as follows. An ellipse represents a concept, a square a value, a rectangle a built-in function. Lines connect concepts and values. A label on a line denotes a relation. A round circle on a line denotes a comparison, such as equality. The green colored graph is the head of the rule, while the gray colored part is the body. Negation is marked by a red background.

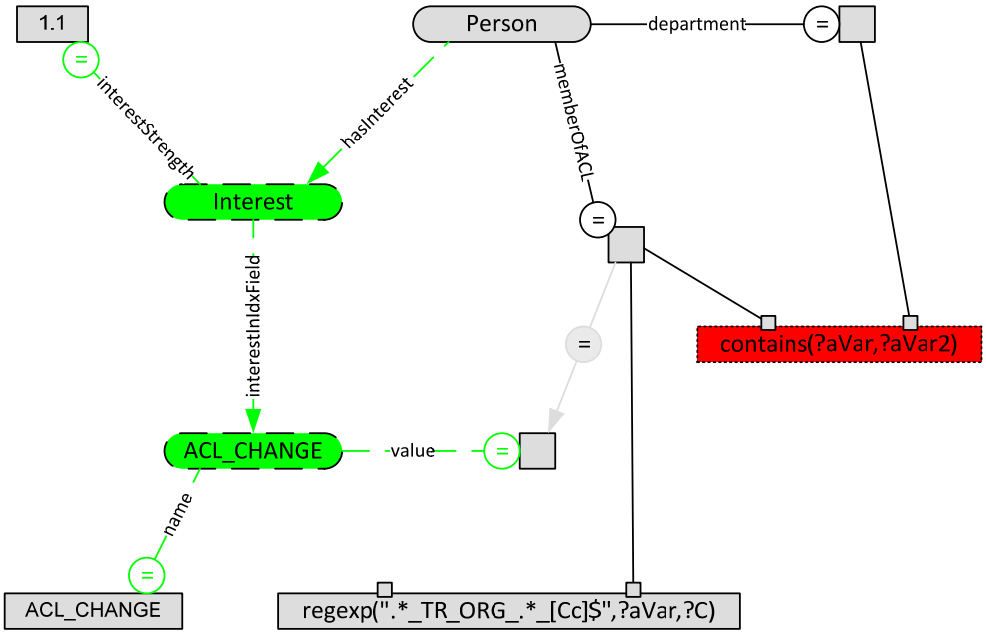
The first rule, “interestInDepartment”, reads as follows (Fig. 6-13): If a person is a member of an organizational ACL which belongs to the area of pharmaceutical research, and if this ACL contains the department the user is working in, then the person’s interest strength into this ACL has a value of 1.35.

Fig. 6-13: Rule “interestInDepartment”



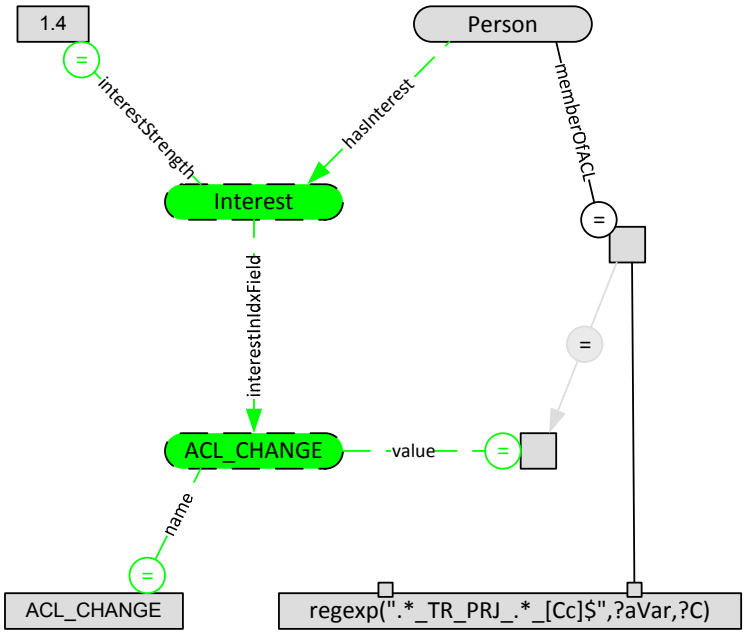
The second rule, “interestInArea” (Fig. 6-14), is very similar to the previous rule. The only difference is that the rule targets all organizational ACLs from TR which do not belong to his department. Further, we set the interest strength to 1.1.

Fig. 6-14: Rule “interestInArea”



The third and last rule, “interestInProject”, reads as follows (Fig. 6-15): If a person is member of a project ACL which belongs to the area of pharmaceutical research, then the person’s interest strength into this ACL has a value of 1.4.

Fig. 6-15: Rule “interestInProject”



The interest strength numbers have been carefully chosen by trial and error. An explanation of how these numbers manifest in the ranking of results is given towards the end of this chapter.

6.4.2 Advanced user roles

The administrative databases enable us to define more adaptation rules than just the previously given examples. We could define user roles which reflect typical work areas of the company such as “Molecular Biology”, “Bioinformatics”, “Lab Worker”, “Manager”, “Fermentation”, etc. Further, we could arrange the roles into a hierarchical structure (e.g. an ontology), so that the associations between them are reflected. Employees would be automatically assigned into a role based on existing administrative information. For instance, a user receives the role “Manager” if he is the head of several sub-groups.

While all of this is feasible so far, the problem occurs when defining the interests of a role. In order to associate documents with these roles we not only need to know the organizational embedding of a document, which can be extracted from its ACLs, but also the content it is about. Then, we could define interest rules such as “A *Manager* is interested in *Portfolio* documents”, “A *Bioinformatician* is interested in documents about *Gene Sequences*”, or “An employee working in *Controlling* is interested in *Accounting* documents”. Further, we could define disinterests like “A *Manager* is not interested in *Lab Device Manuals*”. Hence, we need to know the topic of a document and ideally the entities occurring in it. Creating the ontology containing all the topics and defining preferences for roles can be done with “little” effort. However,

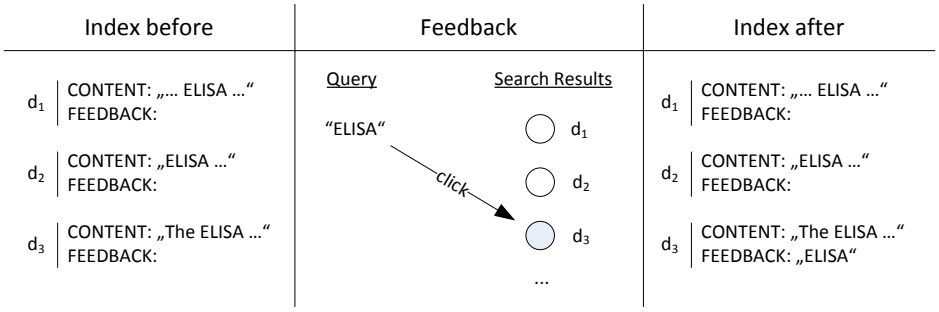
engineering the categorization rules or creating a ML training and test set for each topic is a very time-consuming task. Developing a classifier for each topic is thus out of the scope of this thesis. Therefore, we can only verify the idea for the selected rules and topics.

6.4.3 Log-based adaptation

Log-based adaptation of search results has already been introduced in Chapter 2.6.6 and Chapter 3.2.4. Log-based methods keep track of the user's actions. In particular, they store the implicit click-through data of a user's query session in order to adjust the ranking of search results. The main idea reads as follows: If a document is often accessed in context of a query, then it is more relevant than others, but only in context of the query. This principle can be applied because people usually do not click at random on search results but instead, people read the search result summaries before clicking [Joachims 2002]. The idea of associating queries with clicked documents is described in [Xue et al. 2004] as the "naïve algorithm".

Technically, the algorithm works as follows. Initially, the index contains two fields, a *CONTENT* field in which the full-text of the document is stored and a *FEEDBACK* field in which the click-through feedback is stored – per default this field is empty. If a searcher transmits a query and if he selects an answer, the query term is added to the *FEEDBACK* field. In the example shown in Fig. 6-16, the query term "ELISA" is added to the *FEEDBACK* field of the clicked document d_3. Subsequent clicks are treated analogous, i.e. if a second click under the same setting occurs, the *FEEDBACK* field would contain the text "ELISA ELISA". In future searches, document d_3 will be implicitly ranked higher due to the tf*idf measure. The more clicks, the more often the query word is written into the *FEEDBACK* field, and in effect the higher the tf value and thus the text similarity. This click-through method thus provides query-dependent evidence for the ranking of search results.

Fig. 6-16: Incorporating click-through feedback into the document index.



Whether such a log-based approach has any benefits in a small intranet environment is disputed. The main problem is the discrepancy between the corpus size and the number of users, i.e. clicks. Chapter 8 gives an empirical study of whether the log-based approach has any advantages over the baseline ranking.

Next, we outline extensions of the naïve log-based method by incorporating context information from the classification and adaptation ontology.

6.4.4 Extending knowledge-based adaptation with implicit feedback

The knowledge-based adaptation and the log-based adaptation methods have strength and weaknesses. The KB method works instantly as it is based on demographic data. However, the described KB is unable to learn by observation. The log-method on the other hand needs a certain amount of previously observed log-data in order to be effective. Further, it initially suffers by the cold-start problem because no log data is available.

In order to resolve these issues, we do a linear combination of both, i.e. the relevancy votes of each are linearly combined (cf. next section). Doing so still lets an issue unresolved, namely the fact that in a company the ratio of log data to the number of indexed documents will improve only slowly. The ratio could be improved if the feedback data wouldn't be stored only for one document but for many. This could be achieved if a method similar to [Pretschner & Gauch 1999] would be applied. Instead of considering only the clicked document, the *FEEDBACK* field would be updated for all documents which share the same topic. Of course to be effective, the clicked document must still retain the highest score, while the other documents should gain a lower score. This could be regulated by introducing an additional feedback field which has a lower relevance than the main *FEEDBACK* field.

The suggested method could not only be applied to the topic of a document but also to the user's role. For instance, one could add an additional feedback field denoted *FEEDBACK_ROLE* which stores the role of the person who clicked the document. The role would encode information such as department and projects of the searchers who clicked.

With the last two methods we would effectively expand the amount of reached documents and people.

6.5 Weighting document relevance

The text similarity measure (cf. 2.2.2) is without a doubt one of the most important methods for determining a document's relevancy given a query. Alongside, a number of other evidence factors exist which are more or less successful in improving the retrieval performance (cf. 2.6). Most noteworthy in our context are the previously introduced context-based and feedback-based evidence factors.

The introduced context-based ranking method is a query-independent evidence factor because the documents are scored according to their similarity with the searcher's context. Therefore, the method could be considered user-dependent. A user-dependent evidence factory implies that, depending on the current searcher a document is considered relevant or not. Because context-based adaptation is query-independent it must thus rely upon text similarity in order to achieve a query-dependent similarity ranking. Context is thus an additional evidence factor (just like PageRank) which reflects the overall interest for the searcher. Consequently, special care has to be taken when combining the text similarity measure with the context-based similarity score. On the one hand, if context is stressed too much, a query will not have any influence on the ranking of results. On the other hand, if context is

barely stressed, context will have no influence on the ranking of results. A good balance has thus to be achieved. We configured the parameters in such a way, that text similarity has the strongest influence on the ranking while context applies only subtle adjustments.

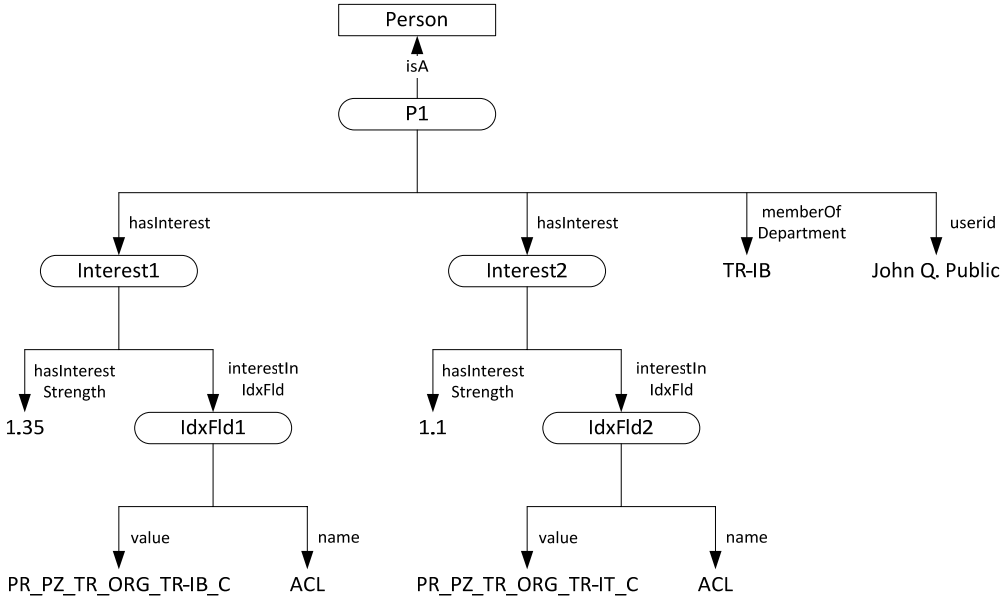
In contrast, feedback evidence is a query-dependent evidence factor. Therefore, adjusting this factor is a less sensitive task because it affects much fewer documents than the context based method.

We begin this section by introducing the computation of the similarity function. Then, we give an analysis of how the context-based evidence factor affects the ranking of results.

6.5.1 Similarity function

Let "John Q. Public" be an employee of Roche. According to his user profile (Fig. 6-17) he is a member of the "TR-IB" department, i.e. the bioinformatics department of pharmaceutical research in Penzberg. Further, the reasoning engine has deduced that he has two interests. Both interests are modeled as interests into documents having a specific security group. The first interest targets the ACL "PR_PZ_TR_ORG_TR-IB_C", which denotes the interest into files from the bioinformatics department. The interest strength is set to a value of "1.35". The second interest targets the ACL "PR_PZ_TR_ORG_TR-IT_C", i.e. documents from the technical informatics department. The interest strength is set to "1.1". Hence, the profile says that the user has a very strong interest in files from the bioinformatics department, a strong interest in files from the technical informatics department, and a normal interest in all other files.

Fig. 6-17: User profile example of a person.



Let's assume next, that the user "John Q. Public" is transmitting a query named "ELISA" to the system. In the first step, the query is modified in such a way that it

targets the index's full text field, namely the *CONTENT* field. In addition the query term is set to a boost factor of 1.0, denoted by the "^" character. The role of the boost factors are explained later. For now it suffices to know that a boost factor of 1.0 is the neutral point. A higher value means the term has more importance and a lower value means that the term is less important. In the subsequent step, the processed query is expanded by the context of the searcher, i.e. the preferred *ACLs* are written into the query and boosted accordingly. The query expansion continues with the last step, in which the *FEEDBACK* field is incorporated. Here, the query terms are simply written again and weighted accordingly. The process is illustrated next:

```
Step 1: Input query q and user u
q="ELISA", u="John Q. Public"

Step 2: Query pre-processing
q →  CONTENT:"ELISA"^1

Step 3: Context expansion
q →  CONTENT:"ELISA"^1
     ACL:" PR_PZ_TR_ORG_TR-IB_C"^1.35
     ACL:" PR_PZ_TR_ORG_TR-IT_C"^1.1

Step 4: Feedback expansion
q →  CONTENT:"ELISA"^1
     ACL:" PR_PZ_TR_ORG_TR-IB_C"^1.35
     ACL:" PR_PZ_TR_ORG_TR-IT_C"^1.1
     FEEDBACK:"ELISA"^2.0
```

So far, we have described how a user's query is expanded, so that it not only targets the document index field *CONTENT*, but also the fields *ACL* and *FEEDBACK*.

The raising question is how to incorporate the index fields and the boosting factors into the scoring function. Incorporating the index fields is straightforward. We simply apply the term weighting schema (cf. 2.2.3) of choice for each field. Then, the individual weighting factors are summed up. Following that, the boost factors are incorporated by multiplication. Assuming that we use a tf-idf schema, boosting is done as follows:

$$w_{i,j} = tf_{i,j} \times idf_i \times boost_i$$

Finally, the ranking scores are computed using the cosine similarity function.

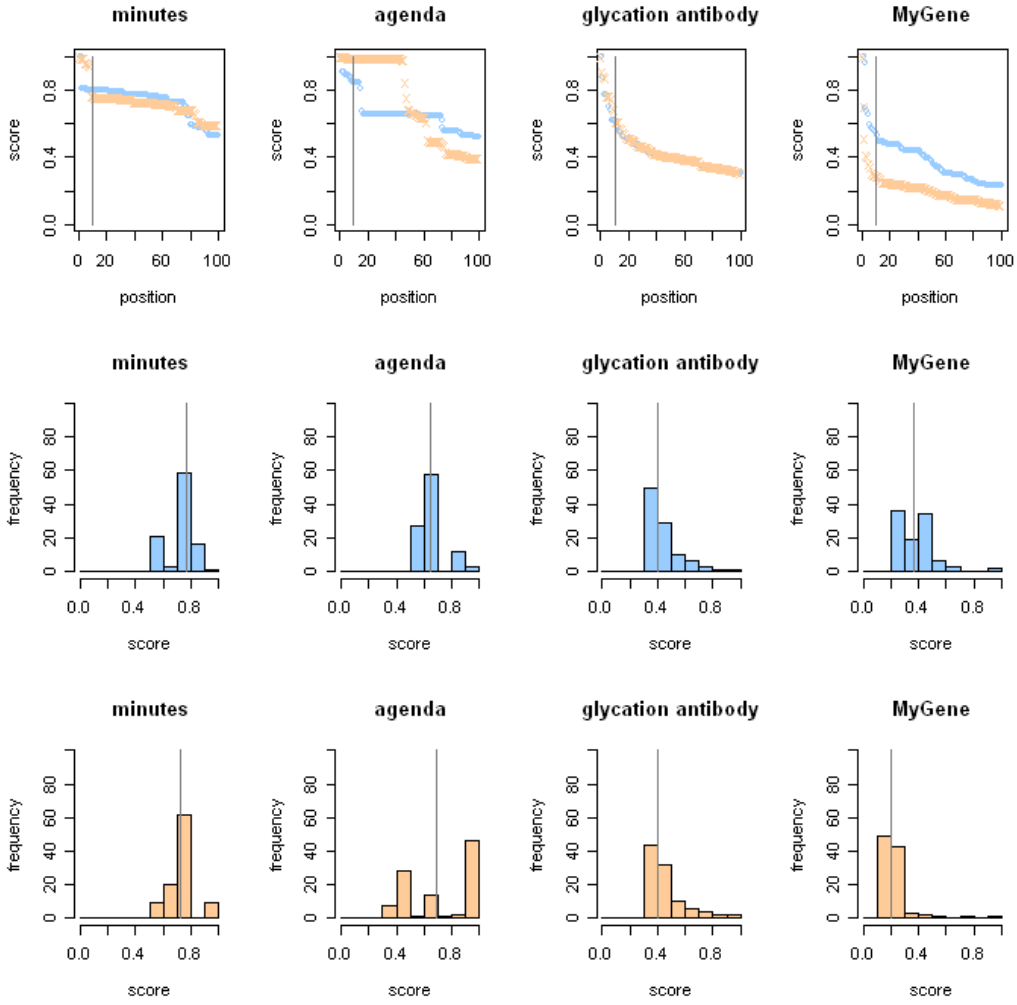
6.5.2 The context's influence on the ranking of results

Even though the methodology of knowledge-based adaptation has been outlined in Chapter 6.4.1, no details have been given about how context influences the ranking of results. Therefore, we outline next how our configuration of the user model influences the ranking of results. In order to do so, we define four representative keyword queries which are benchmarked against an index covering the PRPZ-Share.

The benchmark is conducted with the author's user profile and has thus the context of an informatics worker in pharmaceutical research.

The benchmark results of the four queries "minutes", "agenda", "glycation antibody", and "MyGene" (a human gene; not to be disclosed), are summarized in Fig. 6-18 and Table 6-2. Next, we discuss the results of each query.

Fig. 6-18: Each column illustrates the distribution of the similarity score of a different query. The blue graphs denote the baseline ranking, i.e. the default VSM and the red graphs denote the context-boosted ranking. The gray line marks the median position / score.



The first query, "minutes", is used to retrieve so called "minutes" documents. These are template-based documents whose content give a short summary of a meeting. As a result of this query we get a total of 19,163 hits of which 3.41% are from the searcher's context. Further, hits are found across the majority of departments. Looking at the position – score diagram depicted in Fig. 6-18, we see that the curve has a very low decay. In fact all documents within the top 100 are "minutes" documents and can thus be considered relevant. This observation is supported by the mean similarity score of 0.79. We receive such a high score as this document type is common across several departments and because the majority is based on a common template. Even though in the original baseline ranking not a single

document from the searcher's context occurs in the top 100, adaptation pushes 8 documents from the user's context into the top 10. Context evidence has here a rather strong influence and separation effect on the ranking, because the similarity curve has a very low decay. Hence, a slight boost by context evidence suffices to push an item from the end of the result list towards the front.

The second query, "agenda", is very similar to the previous one. In just the same manner, the intention is to retrieve template-based documents. In contrast to "minutes", "agenda" documents do not summarize the decisions taken in a meeting but they list the topic to be discussed in the session. When searching for "agenda" we get 5,177 hits. This is slightly less than before because not every meeting has an agenda. Nonetheless, the amount of hits matching the searcher's context is almost the double with a value of 7.77%. The position-score curve has a low decay. Only between position 10 and 20 we can observe a high slope. The reason is that the first documents have the word agenda in their path name which gives them a considerable boost in similarity compared to the others. The mean similarity in the top 100 of the baseline ranking has a value of 0.65. This high value indicates that relevant documents are also listed towards the end of the list. In contrast to the previous query, 46 items are from the searcher's context in the top 100. The items from the searcher's context are clearly separated from the others if context boosting is applied.

The query "glycation antibody" is very different from the previous queries as only few documents have a high similarity, while the majority has a low similarity. The mean similarity value of 0.4 supports this. The decay of the similarity curve is thus much stronger. In total 7.51% of the hits are from the searcher's context. In the top 100 only 4 documents are from the searcher's context. Applying context adaptation shifts 3 documents from the user's context into the top 10, but has beside this no major influence on the ranking of results.

Table 6-2: Statistics of result distribution for each sample query.

		minutes	**agenda**	**glycation antibody**	**MyGene**
Total hits		19,100	5,177	346	4,601
Total nr. of hits from the searcher's context		654 or 3.41%	399 or 7.77%	26 or 7.51%	146 or 3.17%
Baseline ranking	Nr. of hits from the searcher's context in top-100	0	46	4	5
	Nr. of hits from the searcher's context in top-10	0	0	0	2
Context ranking	Nr. of hits from the searcher's context in top-10	8	10	3	5

The query "MyGene" has an even stronger decay of the similarity score than the previous query. The mean similarity value is 0.38. In total 3.17% of the hits are from the searcher's context. In the top 100, 5 documents are from the searcher's context. Applying adaptation results in a decrease of the mean similarity value to 0.3. The

reason for this is that in the top 10 of the baseline ranking, 2 items are already from the searcher's context. Hence, if the items, which have already a high score, are boosted, then the difference to the items which do not belong to the searcher's context gets even larger.

We have demonstrated the effect of context based ranking using four different queries. The examples cover two types of queries, namely queries where the similarity score has a low decay and queries where the similarity score has a strong decay. Wrapping it up, context boosting is configured in such a way that it has a strong effect on queries having many similar result items, and that it has a subtle effect on queries having only few result items with a high text similarity.

6.6 Discussion and related work

In this chapter we have shown the concepts of a professional search tool which incorporates the quality features outlined in Chapter 5. We introduced three ontologies which build the "semantic glue" between various parts of the described concept. In detail, they provide the structure for capturing entities appearing within a text document as well as the topic of a document. Further, they provide the foundation for user modeling so that context-based adaptation of search results is enabled. The ontologies are filled with facts by copying existing data from databases into the ontology, by logical inference, and by means of ML and NER.

Interestingly, most parts of the ontologies are in the strict sense only taxonomies, i.e. only few concepts are defined by means of rules. In this respect we could have also used OWL to describe the classes and their relationships. However, modeling the F-Logic rules into OWL would not be feasible. First, OWL has no variables. In effect, modeling becomes difficult, because there is not a "thing" one can refer to. Instead everything has to be described by means of axioms. Second, OWL uses the open world assumption – a paradigm which is good for the Internet but not for closed intranet environments. Third, there is no option for integrating built-in functions in order to externalize calculations which are not suitable to logics.

Regarding relevancy ranking of documents, we incorporated three evidence factors: text similarity based on the tf-idf measure, user context, and search history. These factors were combined by trial and error until the best parameter setting was found. Other evidence factors such as URI length / depth are not incorporated. Similarly, we do not incorporate a document's recency into the ranking of results. Instead the user has the freedom to decide between a ranking by relevance or by date. In future, we might incorporate the timestamp of documents into the ranking process. When doing so, we would apply a knowledge-based approach. In particular, we would encode the relevant time period of common queries in the KB. For instance, we could define a rule which strongly stresses recency if the user is looking for documents of type agenda.

In Chapter 4.3.1 we discriminated three research areas in OBIR: "Search in formal ontologies", "Domain ontologies as a support for document search", and "ontology-based adaptation in IR". The first research area is not covered by our OBIR approach.

Instead we focus on the second and third area. Our OBIR concept is a mixture of "domain ontologies as a support for document search" and "ontology-based adaptation".

Our approach has some overlaps with the approach described in [Castells et al. 2007]. They use in just the same manner ontologies to capture the metadata of documents. In particular documents can be categorized into pre-defined topic taxonomies such as DMOZ. In contrast to our approach they use a simple method based on word-occurrence to determine whether a document belongs to a certain topic or not. This can be compared to some extent with the annotation of the entity type *Tag* conducted in our system. However, we use the entities extracted only as a mean to gather features on which ML or KB text categorization is applied. This additional layer is not described by Castells et al. However, they mention that such an extension would be possible. Another key difference is that we do not transform the initial queries into a rule-based query language such as RDQL. Instead we keep all relevant data in the index and just expand queries to other fields of the index. Therefore, we can use the default VSM without applying any major changes. The last difference is that we offer context-based adaptation while Castells et al. do not.

In [Middleton et al. 2004] an ontology-based recommender system is described. The main difference to our approach is that we use adaptation to adjust the ranking of results and not to provide recommendations. A commonality of the two approaches is that both make use of a topic ontology into which documents are classified. In the case of Middleton, a sub-branch of the DMOZ ontology is used. In this respect they benefit double. First, they do not need to create a taxonomy for their classification task. Second, they can use pre-categorized samples as a training set and can thus circumvent the time-intensive task of creating a training and test set, respectively. This is another difference to our work. External ontologies can usually not be applied as is to the corporate environment. Hence, existing ontologies need to be adjusted or even created from scratch (as was the case in this research). Similarly, creating a training set is a task which comes up in every domain.

The following chapter describes the technical details of the prototype YASA, which is the result of implementing the concepts and features described in this chapter.

Chapter 7

The professional search agent prototype YASA

"The difference between a great design and a lousy one is in the meshing of the thousand details that either fit or don't, and the spirit of the passionate intellect that has tied them together, or tried. That's why programming on the basis of 'lists of features' is a doomed and misguided effort. The features can be thrown together, as in a garbage can, or carefully laid together and interwoven in elegant unification, as in APL, or the Forth language, or the game of chess."

Ted Nelson (1937 –)

7.1 Implementation decisions 131
7.2 Architecture of YASA 134
7.3 Discussion 139

YASA (Your Adaptive Search Agent) is the result of implementing the features and concepts outlined in the previous chapters. It is a prototype of a professional search agent which is made available to scientists in the Pharmaceutical Research division of Roche in Penzberg, Germany. About 400 employees have access to the pilot.

This chapter is organized as follows. First, we describe the implementation decisions taken in YASA, i.e. which components have been used for which part of the system. Second, we describe YASA's architecture. Then, we finish the chapter with a brief discussion.

7.1 Implementation decisions

The first thing we decided was to use Java as the main programming language for developing YASA. Java is platform-independent and more importantly, it enables us to access a large repertoire of public domain software libraries. Another central decision was to make YASA available as a web application (deployed on Apache Tomcat[16]) rather than a stand-alone application. Thereby, users as well as developers profit. Users benefit because no client software needs to be installed – they can access YASA from every standard Roche computer using the web browser. Developers benefit by reduced maintenance costs as updates and patches need only to be applied to one central system.

[16] http://tomcat.apache.org/

Chapter 6 described in detail the architectural components of a professional search tool. The mentioned key technologies were information retrieval, semantic technologies, natural language processing, and machine learning. Given that these technologies are already applied since several years or even decades, software developers had enough time to develop full-fledged frameworks which cover many parts of the mentioned areas. Not only commercial software packages have been shaped in the past but also many public domain solutions. Therefore, we do not re-invent the wheel but rather incorporate existing solutions into our professional search architecture. The following sections give an overview of existing libraries and frameworks on which YASA depends.

7.1.1 Search engine: Lucene

YASA builds upon and extends the Lucene search engine. Lucene provides basic indexing and retrieval functionality and is published by the Apache foundation. Using Lucene instead of a commercial search application (c.f. 2.5.4) has several advantages. First, it does not require a commercial license and regular updates are provided by the community at no costs. Second, the Java source code as well as a comprehensive technical documentation of Lucene is available online. Third, extensions can be easily built on top. Fourth, developers for Lucene are much more likely to cause less costs compared to developers of commercial search engines.

Compared to other search engines, Lucene has a fast retrieval time and a low index size (approximately 25% of the original corpus size). The high compression of the index comes at the cost of indexing time, which takes considerably longer compared to others [Middleton & Baeza-Yates 2007]

Because the positive aspects outweigh, we decided to use Lucene as the search engine on which the prototype was built.

7.1.2 Semantic web middle-ware: OntoBroker

In Chapter 4.2.4 we compared F-Logic with OWL and concluded that F-Logic is better suited for companies due to its closed world assumption, unique naming assumption, and due to its support for rules. F-Logic is distributed commercially by the company Ontoprise, and implemented in the product OntoBroker – a semantic web middle-ware. A framework for ontology modeling is provided by the OntoStudio environment. Since the beginning of 2009, OWL is also supported by OntoBroker and OntoStudio, respectively. Arguably this is a convenient feature as it mitigates potential transition problems.

Another important aspect is that F-Logic can be extended rather easily by means of so called “built-ins”. These are Java routines which can be accessed from F-Logic by means of function names. This is particularly useful if computations need to be conducted which are not well suited for reasoning algorithms. In Chapter 6 we showed some F-Logic snippets in which built-ins were used. In our rules, we use e.g. built-ins to determine if a sequence of entities occurs within a maximal distance of each other – a computationally expensive task if done in F-Logic but a cheap task if done in Java.

The last point regards access to OntoBroker's functionality. All communication is done by means of the WebService interface. OntoBroker is thus only loosely coupled with YASA. A disadvantage of using a WebService is the overhead produced by wrapping all communications in XML. While the milliseconds lost during wrapping have no impact for ad-hoc requests, they sum up quite fast if requests are sent frequently. The latter is the case when crawling and annotating the documents. This was also a reason to limit text categorization to the first 10,000 entities (c.f. 6.4).

7.1.3 Text analysis platform: UIMA

For the implementation of the NLP tasks, we considered two platforms: *GATE* (General Architecture for Text Engineering) and *UIMA* (Unstructured Information Management Architecture). GATE[17] is provided by the Natural Language Processing Research Group of the University of Sheffield since 1995. GATE is a widely accepted and a heavily used toolkit for Text Mining. UIMA[18] was first developed and published by IBM Research in 2005 before it was re-published in 2008 under the hood of the Apache foundry. UIMA particularly focuses on performance and scalability, i.e. the analysis of large volumes of unstructured information to discover new knowledge.

Both software architectures are based on Java (UIMA additionally provides a C++ Framework) and they enable developing and deploying of text mining applications. Indeed, the processing pipeline architecture of UIMA is similar to GATE. The software architecture of UIMA specifies interfaces for components, data (documents), and representations. A particular strength is the possibility to aggregate analysis engines (a component that analyzes text and infers information about them) recursively by composing them from other analysis engines. Wiring and configuration of the analysis engines is done by means of XML files – a convenient feature.

The key difference to GATE is that UIMA uses strongly typed features. In GATE on the other hand, annotations can have any features with any values. Arguably, being able to specify input types and output types of an annotator is beneficial. In particular, typed annotations fit perfectly with ontologies, as these are also strongly typed. Despite the fact that UIMA has not been around so long as GATE, there are already a considerable amount of UIMA compatible annotators available in the public domain. Further, several third-party annotators have been integrated. For instance, the Stanford Named Entity Recognizer – a statistical annotator – is integrated[19] in UIMA. There are even efforts to integrate GATE into UIMA. Therefore, the choice of using UIMA or GATE was strongly pushed towards UIMA as it seems to be the upcoming standard for NLP tasks.

7.1.4 Machine learning platform: WEKA

WEKA (Waikato Environment for Knowledge Analysis) is a software platform for ML developed at the University of Waikato. It is written in Java and distributed under the GNU Public License. The main features offered are: data pre-processing,

[17] http://gate.ac.uk/
[18] http://incubator.apache.org/uima/
[19] http://www.florianlaws.de/

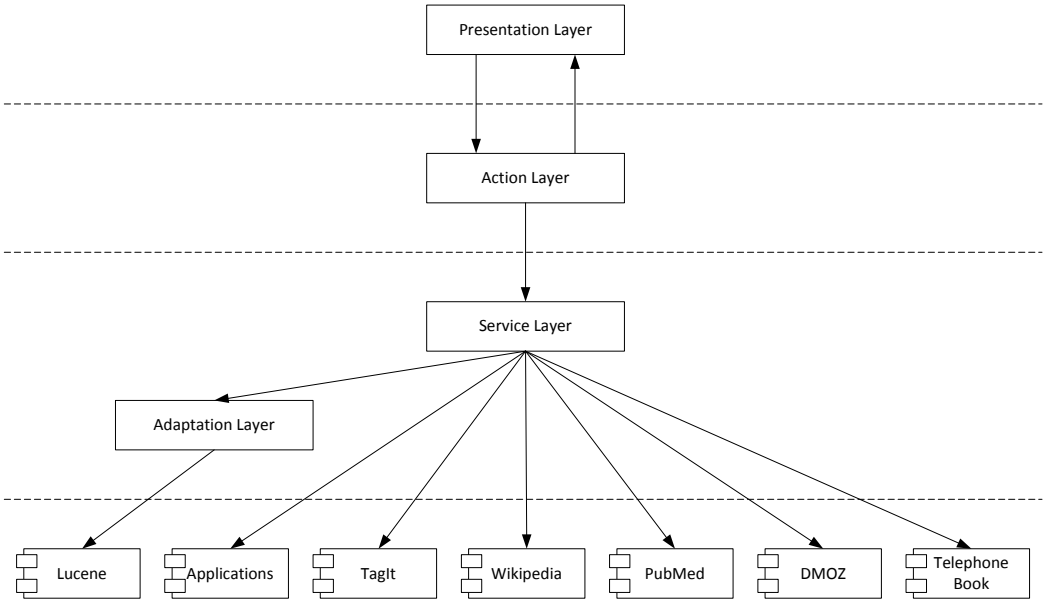
classification and regression, clustering, association rule mining, attribute selection, evaluation, and data visualization. A particular strength of WEKA is the amount of implemented filters and algorithms as well as the ease by which the various components can be combined in a process pipeline.

Because of the rich features WEKA provides, its performance, and its usage in the scientific domain we decided to use WEKA as the machine learning platform.

7.2 Architecture of YASA

Having introduced the components YASA is using, we now go into the details of its architecture. YASA makes use of a multi-tier architecture (Fig. 7-1), which consists of a presentation layer, an action layer, a service layer, and an adaptation layer.

Fig. 7-1: Multi-tier architecture of YASA's search & retrieval part.



The presentation layer is realized using Stripes[20] (a presentation framework for building web applications), JSP (Java Server Pages) together with CSS (Cascading Style Sheets). AJAX is used for fetching data, like the total number of results or tagged documents, asynchronously.

The action layer contains all the "ActionBeans" from the Stripes framework. ActionBeans are responsible for handling actions that are coming from the client (search requests, refine search, export results, etc.) by delegating calls from the client to the service layer.

The service layer stands in the middle between the user's requests and the sources containing the data. Basically, it delegates the search request to each source and then returns the results. In case the queried source supports guided navigation, then facets are also returned. Currently, faceted navigation is provided by the source

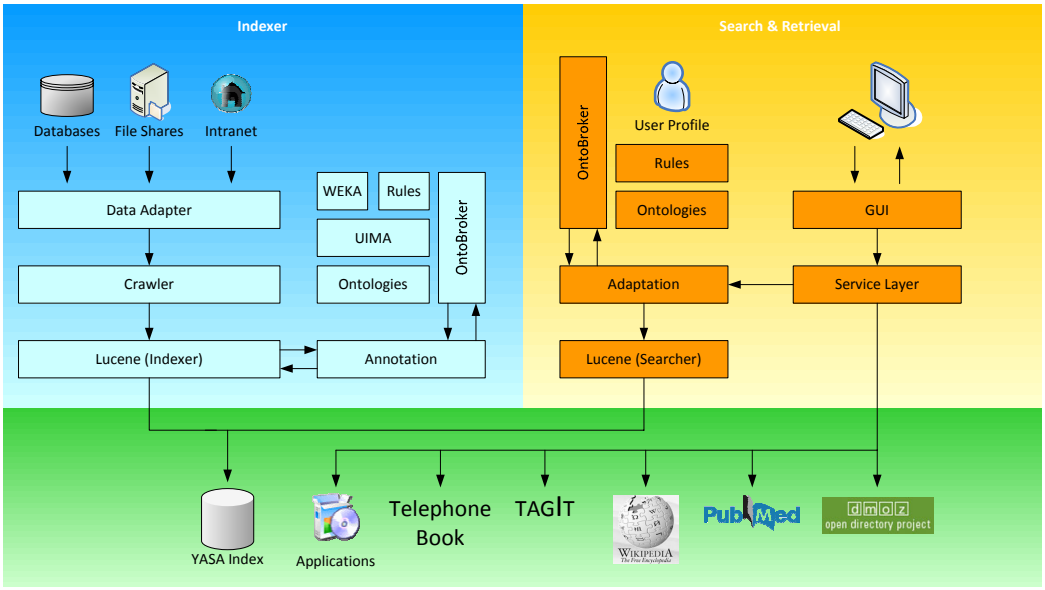
[20] http://www.stripesframework.org/

"Applications", and by YASA's local index. Communication with the sources is done by means of WebServices, direct HTTP requests, or by means of direct API calls.

The adaptation layer builds the bridge to YASA's internal index. In particular, the current user is delegated to OntoBroker so that his context is returned. Given the searcher's working context, the initial query is adjusted so that the user's assumed interests are reflected. The expanded query is then transmitted to the index and the results are fetched.

YASA consists of two parts (Fig. 7-2). The first part contains the index and annotation functionality, which enables YASA to create a full-text & metadata index of documents located in databases, file shares, and intranet web sites. The built-in indexer is used to scan local sources of Pharma Research in Penzberg, i.e. a workflow database, a plasmid database, a hardware inventory database, the PRPZ-Share, and the PRPZ-WebSite. The second part of YASA contains the search & retrieval functionality. The core components of this part are the GUI, the service layer, and the adaptation layer. The service layer delegates queries and results between the sources and the GUI. The adaptation component adjusts the ranking of results for the sources indexed by YASA.

Fig. 7-2: The components of YASA.



Next, we discuss some key components of YASA in more detail, namely the data connectors, the crawler, the annotator, the GUI, the service layer, and the interfaces of the searchable sources.

7.2.1 Data adapter

The data adapter of YASA enables access to databases, file shares, and intranet web sites. The content of files is parsed with the help of several third party tools such as the Apache POI[21] and the Apache PDFBox[22] libraries. In effect, we are able to parse

[21] http://poi.apache.org/

plain text files, all MS Office formats, PDF files, and the metadata of various image file formats.

Integrating databases requires several issues to consider. First, a user with read permissions is required so that a connection to the database can be established. Second, the tables and relations to be indexed must be specified. Third, the database columns must be mapped to fields known in YASA. Fourth, the text processing pipeline (e.g. tokenization) should be configurable for each database column, i.e. index field. Fifth, the view of result items must be defined. The value of a single database cell is useless without its context, i.e. its relations to other columns. Sixth, the security settings must be defined, i.e. who is allowed to view the items and who is not. Indeed, this a big issue because database systems do not have a common schema of how to store security settings. Inherent to this problem is the inability of database systems to incorporate the security schema of the operating system.

In order to solve these problems, database sources crawled by YASA are described using XML files. Such an XML configuration file typically contains connection parameters, select statements, field mappings, view settings and global security settings. XML files are also used to manage the fields used in YASA. In particular, new fields can be added dynamically, and it can be specified how they should be processed (i.e. which text processing methods to use) and how they should be displayed in the search results. YASA parses the XML files and then takes the required actions. XML is not always the cleanest way to represent domain knowledge. Indeed, the mapping of "database columns" to "index fields" would be better expressed in an ontology. However, at the time the DB module was developed, we had not yet licensed the semantic web middle-ware OntoBroker so that a pragmatic approach was chosen. In future, we will migrate to an ontology-based mapping approach.

The file shares are accessed by means of the samba networking protocol. Because the Java Development Kit v6 does not support this protocol, we had to use a third party component named JCIFS[23]. This is an open source library which implements the samba networking protocol. In addition it enables us to extract the files' security permissions – an information on which adaptation heavily relies.

The integration of the local web site was – except one minor issue – straightforward. The PRPZ-WebSite contains a common frame across all pages, which provides links for navigating the content. This frame posed a problem for search: if e.g. the frame contains the word "antibody" and a user searches for the term "antibody", then all pages of the web site would be returned because all pages share the same frame. Arguably, this is not what a user expects. Therefore, we developed a filter by which layout frames could be removed. The exclusion of the surrounding frame removes potential false positive hits.

[22] http://incubator.apache.org/pdfbox/
[23] http://jcifs.samba.org/

7.2.2 Crawler

The crawlers are executed sequentially, i.e. the file share is scanned first, then the web site, and finally the databases. In addition we have two other crawling routines. The first one scans the log data and extracts the relevant information so that the *FEEDBACK* fields of the index can be updated accordingly (c.f. 6.4.3). The second crawler scans the index and updates the metadata in case the annotators, i.e. the classification rules or the learned models change. This might be the case if new document topics are incorporated.

In order to improve crawling speed, the file shares as well as the web pages, are scanned in parallel using multiple threads. This way the available network bandwidth is used more efficiently.

The crawling process can be scheduled to run in a certain interval. In our case crawling runs once a day, starting at night – a period in which we expect the lowest network traffic.

7.2.3 Annotator

The annotation module communicates with UIMA, WEKA, and OntoBroker. The full-text content is passed first to UIMA which extracts the entities. These are then either transmitted to the OntoBroker so that the knowledge-based classification can be conducted or they are transmitted together with the full text to WEKA so that categorization based on machine learning is done. In case annotations could be extracted they are transmitted back to the indexer.

The annotation functionality of UIMA is encapsulated in a web service, so that other applications can also benefit from the annotators. Similarly, OntoBroker is accessed by means of a web service interface. Only WEKA is directly accessed by means of a local API call.

7.2.4 Indexer and searcher

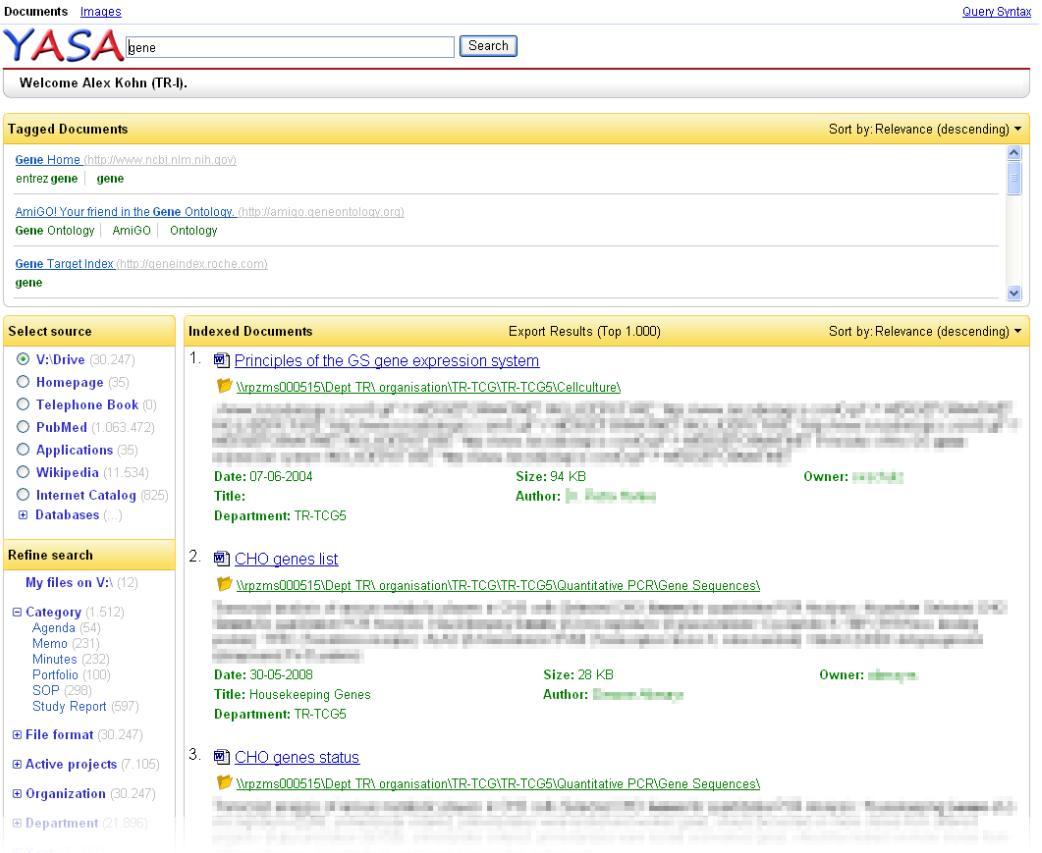
The Lucene indexer receives the full-text content (if available) and all metadata information which could be extracted from the documents. This data is then written to the index. YASA keeps three separate indexes – one for each type of source (database, file share, intranet web) – so that maintenance costs (index optimization, index updates) are reduced. Separating the indexes does not involve any serious disadvantages. Indeed, one could even merge the results of all three indexes seamlessly, because the same ranking algorithms are applied on all of them.

7.2.5 Graphical user interface

The graphical user interface of YASA consists of five sections which are described next (Fig. 7-3). The top of the GUI contains the query input box. Below that, the section “Tagged Documents” is located. Here, search results from Roche’s tagging application (TagIt) are displayed. A result answer in this pane consists of the document’s title, its URL, and the assigned tag labels.

The section below the “Tagged Documents” pane is split into two. The left hand side contains various filters for the result items displayed on the right hand side. The upper filter pane, named “Select source”, is used to restrict results to a certain source. The lower filter pane, named “Refine search”, offers several facets by which results can be sliced and diced. The visibility of a facet depends on the result set and on the selected source. The displayed results in the “Indexed Documents” pane are updated depending on the selected filters. Because of the heterogeneity of the sources, the amount of information displayed in the result pane varies. In case of file shares, information such as the document’s name, its author, the department to which it belongs, and its topic are displayed. In case the selected source is a database, the displayed information might be completely different. Search results in the “Tagged Documents” and “Indexed Documents” pane can be sorted by relevance or by date using descending and ascending order, respectively.

Fig. 7-3: Screenshot of YASA’s search interface.



YASA also provides image search, which can be reached from the very top. Image search is structured similarly to document search. A difference is that result items consists of thumbnail picture previews instead of text snippets.

The user interface is also the place in which the user’s interactions with the system are tracked. In particular, transmitted queries as well as selected result answers are recorded. In addition anonymous information is stored, such as the department and involved projects of the searcher.

7.2.6 Service layer

The service layer contains the client-side code for querying the sources YASA (databases file shares, web sites), Applications, TagIt, Wikipedia, PubMed, and DMOZ. Each of the sources is made accessible by a remote interface, i.e. they offer at least a method by which the data can be queried. The DMOZ interface additionally offers a method for retrieving facets. The remote interface can be a simple CGI interface as is the case for PubMed and Wikipedia, or it can be a web service interface as is the case for TagIt, Applications, and DMOZ. Only the index of YASA is accessed by a local API interface using Lucene's searcher module.

The external sources (Wikipedia, PubMed and DMOZ) are mirrored in-house in order to prevent confidential information from leaking to the public. The mirrors are updated once a month. The PubMed mirror was already available when YASA was developed. The Wikipedia and DMOZ mirrors on the other hand had to be implemented from scratch. Similarly, the services for the sources Applications and TagIt have been implemented while YASA was under development.

All services except TagIt make use of Lucene in order to index their data. In contrast, TagIt does not have a full-text index but merely a database driven search functionality. This solution is not optimal in terms of information retrieval. First, no ranking by text similarity is provided and second, word variants are missed at search due to the lack of stemming. The first issue is handled by sorting tags by the number of times a tag has been assigned to a URL, and the second issue is simply ignored.

Arguably, the service layer is a place where overhead is created because queries are transmitted blindly to every source. Indeed, it would be better if we knew which sources are relevant for the query and which not. Then, we could restrict result fetching to the relevant sources only. In context of this thesis we did not investigate this issue. However, we believe this would be an interesting research field, especially if the number of considered sources is large.

Another service which is neither depicted in Fig. 7-1 nor in Fig. 7-2 is the Feedback service. This service provides an interface for storing and querying log data. The service is called from the GUI (using AJAX technology) each time a user is searching, navigating, or selecting a query. The service is also called during the crawling process in order to update the *FEEDBACK* fields in YASA's index.

7.3 Discussion

We introduced YASA, a domain-aware complement to standard search engines. YASA gives access to multiple sources, it offers role-specific ranking of results, and it exploits a company's knowledge, i.e. existing metadata and log data to improve several aspects of search for information.

YASA applies a federated search approach in which search results of the sources are not merged but rather displayed separately. Using this approach for achieving a one single entry point has several implications. First, we circumvent the problem of

merging search results from heterogeneous sources [Hawking 2004]. Second, federation is not a highly scalable solution.

On the one hand, federation increases the network traffic because search requests are transmitted to each source. On the other hand, relying on third-party search services means that an accurate result list cannot be built until all services have returned their answers. Consequently, the speed at which the results are displayed depends on the response time of the slowest search service. In fact, we have to admit that we can already notice short delays until the results from PubMed, DMOZ, Wikipedia, and TagIt are returned.

Despite these issues, we argue that YASA's approach is feasible because YASA incorporates relatively few external search services. Indeed, the frequently queried internal sources (databases, file shares, web sites; Chapter 8.1.1) are indexed by YASA itself so that no overhead is produced. Nonetheless, merging and displaying results from databases, file shares, and web sites is not trivial. Therefore, we do not yet offer a unified view across all sources.

In the annotation as well as the adaptation components, web services are used for communication purposes. Because the services are remote and because data needs to be wrapped into an XML data exchange format, a certain amount of latency occurs. During annotation this latency can sum up considerable. Nonetheless, a user does not notice this because indexing is done offline in the background. In case of adaptation of search results – which is online – the latency is also not noticeable because a single request takes only a few milliseconds.

Another aspect concerns the search interface. Per default, a keyword based search approach is applied, in which most indexed fields are queried. Additionally, a user can refine search results by using the GUI's refine search options. Further, advanced users can directly restrict the search scope by using YASA's advanced query syntax. The query "AUTHOR:John" e.g. would restrict search to the index field *AUTHOR*.

The last issue we want to discuss concerns the usage of text processing on corpora containing more than one language. Indeed, on the PRPZ-Share we can find documents in German language, English language, and even documents written in both languages: 21% of the documents are in German, 54% of the documents are in English, and 25% of the documents contain a mix of English and German sentences. Considering this distribution and especially the large amount of documents using both languages, we created our text processing pipeline as follows. Tokenization is done with the Porter Stemmer – we thus ignore the German language in this step at the risk of obtaining wrong stems. Stop word removal is done for both languages, i.e. we incorporate a stop list consisting of German as well as English words. Arguably, performance could be improved if we would use a German word tokenization algorithm for the 21% documents written in German language. However, doing so would have added more complexity to the system. Even though we renounced on this step in the prototype, it could be added in future releases if necessary.

Chapter 8

Evaluation

"A theory is something nobody believes, except the person who made it. An experiment is something everybody believes, except the person who made it."
Albert Einstein (1879 – 1955)

8.1	Usage, access, query, and session statistics	141
8.2	Text categorization performance	150
8.3	Retrieval performance evaluation using click-through data	155
8.4	Controlled experiments	161
8.5	Discussion	176

This chapter presents empirical results obtained by evaluating the YASA prototype in the professional context of research and development. The analysis is focused on determining which of the principles implemented in YASA bring a significant improvement in professional search. We performed online as well as offline evaluations of the system. Online evaluations (observational studies) were conducted by recording the click-through data. Offline evaluations were done by means of controlled experiments. The online evaluation has the advantage that a lot of data can be captured to insignificant costs. However, guessing but not knowing the user's intention, the interpretation of results could be misleading. The offline evaluation on the other hand does not have this disadvantage. However, only few persons can be convinced to participate in such studies.

We begin this chapter by giving general statistics about YASA in the deployed context. In particular, we analyze how the usage of YASA and other search engines changed with time. Further, we provide an analysis of the query sessions in YASA. Following that, we focus on the text categorization performance. Then, the retrieval performance is evaluated. Next, the results of controlled experiments are shown which target various aspects of the prototype. Finally, we conclude the chapter by giving a wrap-up and interpretation of the obtained results.

8.1 Usage, access, query, and session statistics

This section examines several aspects of the search engine prototype YASA and its usage within the investigated departments. We start by investigating how the usage of search engines on the PRPZ-WebSite has changed since 2007. Then, we focus on the prototype YASA: First, we examine which sources are primarily accessed by the

scientists; Second, we conduct statistics about individual queries; Third, query duplicates are investigated; Fourth, the characteristics of query sessions are outlined; Fifth, click statistics are given.

All statistics are based on log data which were mostly gathered between January '09 and June '09. Prior to the calculation of the statistics, we pre-processed the log-data so that the queries have a canonical form. In particular, all queries were lowercased, special characters were removed, and queries were split into terms using a whitespace tokenizer. Further, query terms were paired with any given operators (AND, OR, and NOT) and ordered alphabetically. The last step could be done because the term sequence has no influence on the result set produced by YASA.

Before proceeding, we give some general statistics about YASA's index. The index contains approximately 675k documents (Table 8-1). Most documents are located on the PRPZ-Share having about 670k files. The PRPZ-WebSite and the indexed databases contain only 1.7k documents and 4k records, respectively.

On the PRPZ-Share we counted in total about 96M distinct terms. The total is extremely large because it includes also non-dictionary words such as arbitrary numbers (e.g. "1234"). A filter counting only the words from the dictionary was not applied. The language of the files distribute as follows: Approximately 50% of the documents are written in English, 20% are written in German and the rest contain a mix of sentences written in both languages.

Table 8-1: YASA index statistics in July '09.

	Nr. of Documents / Records	Nr. of Terms (in CONTENT field)	Language		
			de	en	mixed
PRPZ-Share	669,745	95,953,036	21%	54%	25%
PRPZ-WebSite	1,753	39,950	19%	51%	30%
Databases	4,048	-	-	-	-

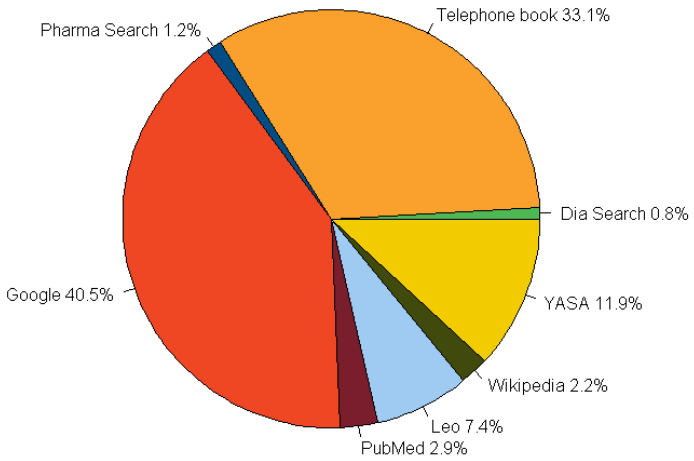
8.1.1 Changes in the search engine usage on the PRPZ-WebSite – A log file analysis

In Chapter 5 we conducted an initial analysis regarding the IR situation in the investigated departments. Amongst others we investigated in 2007 the usage of search engines linked from the local web site of Pharmaceutical Research in Penzberg (Fig. 5-4). Since then, the access profile changed significantly (Fig. 8-1). First of all, the number of search engines linked on the PRPZ-WebSite raised from six search tools in 2007 to eight search tools until the mid of 2008. The additionally linked search engines are the prototype YASA, the online dictionary Leo (English-German), and a search engine targeting the Diagnostics division of Roche. The tool "Google search appliance" is not listed anymore as its license was discontinued after the introduction of YASA.

Google's web search is still the most frequently accessed tool in the investigated departments, followed by the in-house telephone book. While access to the

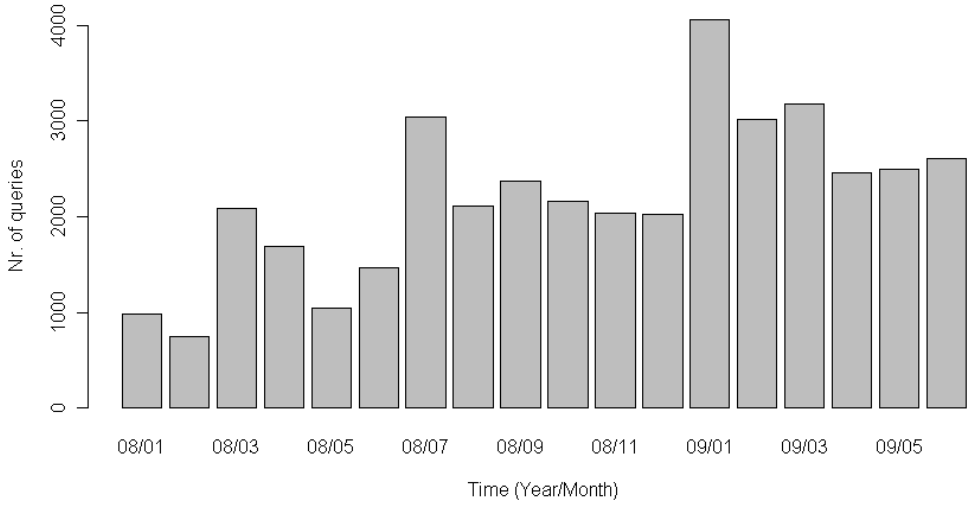
telephone book remained about the same, access to the search engine Google dropped by approximately 10%. The most likely reason is that the search engine Leo is now directly accessible from the PRPZ-WebSite. Prior, workers were used to reach the dictionary by transmitting the navigational query “leo” to Google – as could be observed from the logs in 2008. Further, we assume that another reason might be due to YASA’s integration of DMOZ, which covers parts of the Internet. The usage of the search engines targeting the pharmaceutical intranet web pages and the diagnostics intranet web pages has not changed significantly. Similarly, access to Wikipedia remained about the same. The usage of PubMed however, grew slightly. The most significant change occurred in the search engine which originally targeted the PRPZ-Share. In the past, Google’s Search Appliance had a fraction of only 0.3%. YASA on the contrary, which has replaced Google Search Appliance, has reached a quota of 11.9% – a significant increase.

Fig. 8-1: Usage of search engines linked from the PRPZ-WebSite.
Data logged over a period of 6 month (Jan ‘09 – June ‘09).



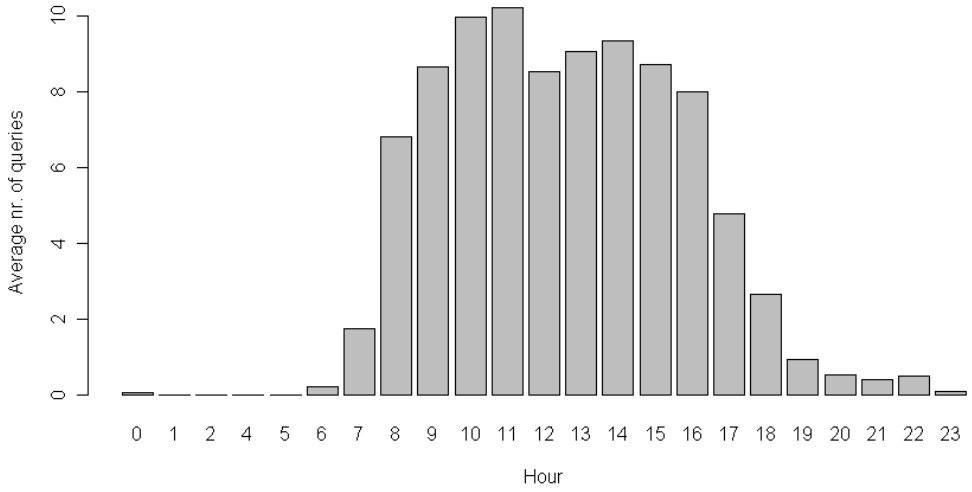
The first version of the prototype YASA was introduced in January 2008. At that time YASA’s functionality was very limited: Search was only possible on the PRPZ-Share and adaptation was merely applied based on the searcher’s departmental background. In March 2008, we introduced the first prototype to approximately 40 scientists from Pharmaceutical Research in Penzberg. As a result the usage number grew significantly in that month (Fig. 8-2). In the subsequent month more and more functionality was integrated. Most notably, in July 2008, we expanded the coverage of the PRPZ-Share to the patents department and in 2009 we indexed all of the PRPZ-Share including secured folders. In January 2009 an increased usage of YASA could be observed. Eventually this is because of the inclusion of the previously not searchable secured folders. Wrapping it up, the usage of YASA grew steadily the more principles were implemented and the more workers heard about the tool.

Fig. 8-2: Number of queries transmitted to YASA per month.
Data logged over a period of 18 month (Jan '08 – June '09).



The diagram displaying the query-per-hour distribution (Fig. 8-3) follows the typical pattern of the workday. At about 8 o'clock a large increase can be observed, which has a local minima at noon due to lunch, before it decays the later the hour gets. The peak usage of about 10 queries per hour is reached at 11 o'clock.

Fig. 8-3: Average number of queries per hour.
Data logged over a period of 18 month (Jan '09 – June '09).

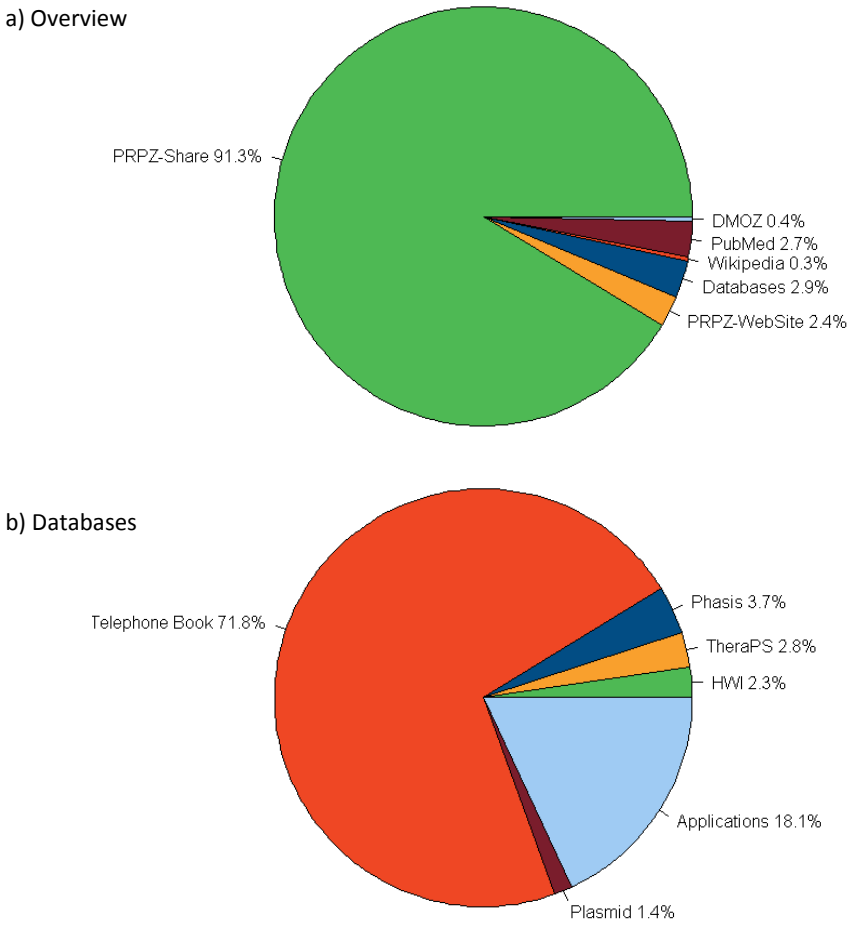


8.1.2 Access of sources within YASA

In order to gather information about access to sources within YASA, we traced source access over a period of three month in 2009 (Fig. 8-4). The vast majority of queries (91.3%) target the PRPZ-Share. The sources PRPZ-WebSite and PubMed have a similar access of 2.4% and 2.7%, respectively. DMOZ and Wikipedia are barely queried as is reflected by their access frequency of about 0.4%. The database sector has a fraction of 2.9%. Here, most requests target the telephone book (71.8%) – a similar observation to that in Fig. 8-1. The software application repository is the second most frequently accessed database having a fraction of 18.1%. The databases

Phasis (Pharmaceutics animal study information system), TheraPS (workflow database), HWI (hardware inventory), and Plasmid are barely accessed by means of YASA. Their low usage was expected. The telephone database and the application database contain information which is freely accessible by all searchers and which is relevant to most employees. In contrast, access to the other databases is often restricted and they contain very specific information which is only relevant to a minority of the staff. Hence, our observation reflects well the access distribution to the sources.

Fig. 8-4: Query distribution on sources within YASA.
Data logged over a period of 3 month (Apr '09 – June '09)

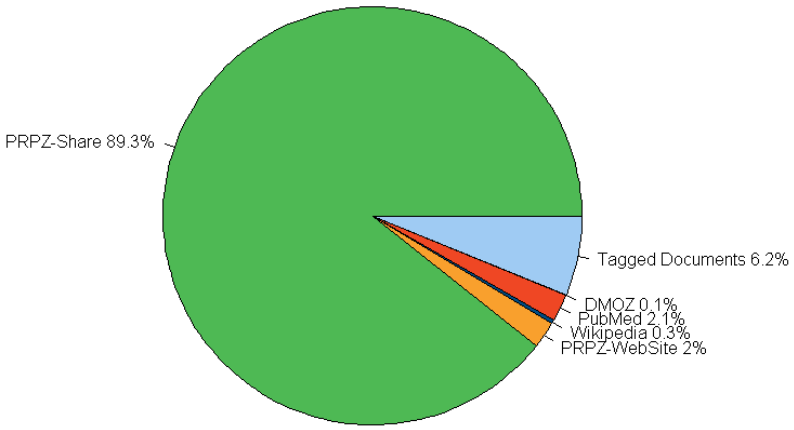


In addition to the queries we also logged the clicks, i.e. the document access frequency, for some sources (Fig. 8-5). We excluded databases from the click statistics because only few databases offer clickable links. Even though, the database "Applications" does usually offer clickable links to the intranet homepage of the listed application, there are still some entries which do not have a homepage associated. Similarly, for the databases "Telephone Book", "Phasis", "TheraPS", "HWI", and "Plasmid" there are no clickable links available. Instead, all relevant information is displayed with the result item. Indeed, offering links would be a convenient feature especially for databases which are highly related to applications such as "Phasis", "TheraPS", and "Plasmid". A hyperlink which would open the

selected item in context of the application would be a convenient feature. However, the respective applications lack this linkage functionality.

The statistics for the selected sources (i.e. the non-database sources) shows that most document requests (89.3%) target the PRPZ-Share. The remaining 11.7% percent are partitioned amongst Tagged Documents, DMOZ, PubMed, Wikipedia, and PRPZ-WebSite. The source "Tagged Documents" is the leader of this minority having a frequency of 6.2%. Following that, PubMed as well as PRPZ-WebSite have an access frequency of about 2%. The least targeted items are results from Wikipedia (0.3%) and DMOZ (0.1%).

Fig. 8-5: Click distribution of sources within YASA.
Data logged over a period of 6 month (Jan '09 – June '09).



8.1.3 Statistics concerning individual queries

During the log period in 2009 we recorded a total of 6,897 distinct queries which have been transmitted to YASA. The average query has 1.69 terms. Most queries have either one term (56%) or two terms (29%). Hence, the number of terms used in a query is usually small, even though occasionally queries with up to 30 terms could be observed (Table 8-2).

Table 8-2: Number of terms per query.

Queries having 1 term	56.0%
Queries having 2 terms	29.4%
Queries having 3 terms	9.9%
Queries having >3 terms	4.7%
Max terms per query	30
Avg. terms per query	1.69

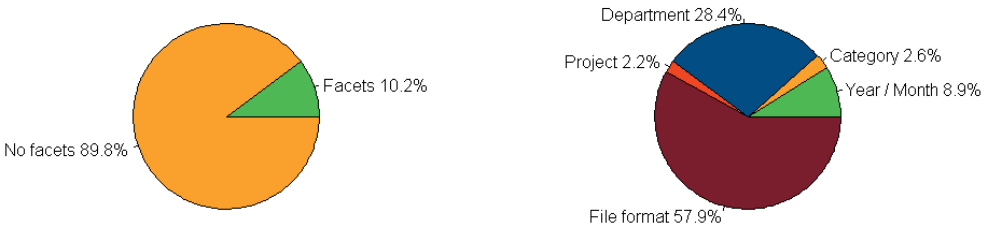
Concerning the Boolean operators (AND, OR, and NOT) we could observe that these are merely used in 0.45% of the queries (Table 8-3). Further, we could observe that if operators are applied, most searchers use only one operator, very rarely two, and never more than two operators. This result is in contrast to the empirical study conducted by Mühlbacher, which suggests that Boolean operators are considered quite important by scientists at Roche (Chapter 5.1.5).

Table 8-3: Number of Boolean operators per query.

0 operators in query	99.55%
1 operator in query	0.43%
2 operators in query	0.02%
>2 operators in query	0%
Max operators in query	2
Avg. operators per query	0.0045

Facets on the other hand are used considerably more often. Indeed, users are refining search results by means of guided navigation in 10.2% of the searches (Fig. 8-6). The "File format" facet is the most used aspect by which documents are filtered (57.9%). Indeed, the file format is a very good tool to filter results by spreadsheets (presumably experimental results), presentations, and full-text documents. The second most frequent facet is "Department" (28.4%). The "Year / Month" facet is the third most prevalent aspect (8.9%) by which results are filtered. The "Project" and "Category" facets are used least with a frequency of 2.2% and 2.6%, respectively. The most often accessed topic is "Minutes", followed by "Agenda", "Portfolio" and "Study Report". Notice, that the facet "Category" was introduced only recently in June. We assume that its usage is biased due to many people who have tested this new functionality. Therefore, the given statistics might not reflect well the facet's future usage statistics.

Fig. 8-6: Usage of facets. Data logged over a period of 6 month (Jan 09 – June 09).



8.1.4 Statistics concerning query duplicates

Query duplicates are queries which are transmitted more than once to YASA (Table 8-4). The top 20 queries submitted to YASA are a mix of project names, person names, and technical terms. The top queries are not listed in this thesis due to confidentiality reasons. On average, each query is transmitted 1.32 times to the system. The top query was executed 59 times. The vast majority of queries (86.4%) was transmitted only once to the system. The remaining 13.7% of the queries are transmitted more than once. Queries which are transmitted more than three times are quite rare (3.4%).

Table 8-4: Statistics of query duplicates.

Query occurs 1 time	86.4%
Query occurs 2 times	7.7%
Query occurs 3 times	2.5%
Query occurs >3 times	3.4%
Max query frequency	59
Avg. query frequency	1.32

8.1.5 Statistics concerning query sessions

We define a query session as a sequence of queries in which two consecutive queries occur within a maximal distance of 10 minutes. The majority of query sessions (72.3%) consist of only one query (Table 8-5). Sessions having two queries are encountered in 14.8% of the cases. Sessions having more than two queries are less frequently. A session contains on average 1.64 queries. These results suggest that query sessions are simple. The most complex session consists of 43 queries.

Table 8-5: Queries per session.

1 query per session	72.3%
2 queries per session	14.8%
3 queries per session	5.8%
>3 queries per session	7.1%
Max queries per session	43
Avg. queries per session	1.64

Another session statistics we consider, concerns the number of result pages viewed per session (Table 8-6). In YASA, 10 result items are maximally displayed on the screen. In order to view more, the page navigation has to be used. The results show that on average a searcher looks at 1.18 result pages per session. Searchers view only one result page for the vast majority of the queries (91.8%). Observations of query sessions in which two, three, or more pages are viewed, are considerably less frequently. The maximum number of result pages viewed in a query was 31.

Table 8-6: Result pages of queries viewed per session.

1 result page per query	91.8%
2 result pages per query	4.6%
3 result pages per query	1.6
>3 result pages per query	2.0%
Max result pages per query	31
Avg. result pages per query	1.18

We also examine how terms are modified within a query session (Table 8-7). The following types of modifications are distinguished: deletion, insertion, complete change, and other modifications. We count a “deletion” if from one query to the subsequent query no other changes could be observed except the deletion of terms. In just the same way we count an “insertion”. A modification of the type “complete change” is counted if none of the original query terms can be observed. Last, the type “other modifications” covers all other cases such as queries in which terms are deleted and added in one step. Given these types, we observed that in most sessions, the queries are completely modified (69.8%). Only in 12.4% (1.5%) of the queries are new terms (facets) inserted. A generalization of queries (i.e. deletion of term) occurred in 3% of the query sessions. 14.8% of the modifications could not be classified into any of the given types.

Table 8-7: Term modifications per query session.

Terms deleted	3.0%
Terms inserted (Facets inserted)	12.4% (1.5%)
Terms completely changed	69.8%
Other modifications	14.8%

8.1.6 Statistics concerning click statistics

Click statistics are about the amount of accessed documents per query (i.e. the number of mouse clicks). Because some sources do not require to open a document after the query is transmitted (e.g. in databases the result is displayed on the screen), we restrict for the following statistics to data from the PRPZ-Share. Further, we had to remove the baseline of queries which were intended for other sources. The problem is that per default the initial query is transmitted to the PRPZ-Share even though the searcher might be interested in results from another source. Hence, eliminating the baseline was necessary. The processed results read as follows (Table 8-8). In 57.7% of the queries, no item is selected. In 25% of the queries only one document is accessed. Two documents are accessed in 8.4% of the queries, and three documents only in about 4% of the queries. The maximal observed clicks for a query is 42. On average a searcher clicks 0.87 times per query.

Table 8-8: Clicks per query.

0 clicks per query	57.7%
1 click per query	25.0%
2 clicks per query	8.4%
3 clicks per query	3.9%
>3 clicks per query	5.0
Max clicks per query	42
Avg. clicks per query	0.87

8.1.7 Discussion

Several of the previously shown statistics have also been conducted in a study which targeted a large query log of the search engine AltaVista [Silverstein et al. 1999]. The analysis was conducted about 10 years ago in 1999. Arguably, this is a long time period so that search behavior might have changed significantly: First, search algorithms have improved since then; Second, much more people are using the Internet since then. Nevertheless, we compare the results obtained within the Roche intranet with the results published by Silverstein et al., because we have not found a similar and more recent study.

According to Silverstein's analysis, operators are used in approximately 20% of the searches. This is in contrast to our results, in which operators are de facto not used. Regarding query duplicates, Silverstein reports that 27% of the queries transmitted to AltaVista are duplicates. In our case, only 14% of the queries transmitted to YASA are duplicates. This suggests that the searchers at Roche do often search for previously unseen information (informational queries). This result reflects well the fact that YASA is deployed in a research department: Research is a highly dynamic field in which the interests might change fast. The results of the session statistics are quite similar. In Silverstein, 2.02 queries are transmitted on average per session.

These are a bit more queries compared to the average number of 1.64 transmitted queries in YASA. Further, the average number of viewed result pages is slightly lower in YASA (1.18) compared to AltaVista (1.39). Regarding the query modifications, the results of the two studies vary significantly. Most notable is the fact, that a complete query change is detected about double the frequent in YASA as in AltaVista.

The statistics related to the indexed databases have to be interpreted with care because very few queries have been transmitted to these. Indeed, database integration is done technically as a proof of concept, but it lacks important features which attract people to search these sources. In particular, a link to the related application would be convenient. Further, an automatic generation of statistics (displayed in histograms, graphs, etc.) based on the retrieved results would generate an additional value for the searcher.

8.2 Text categorization performance

This section compares the text categorization performance of the KE (knowledge engineering) and ML (machine learning) approach (Chapter 6.3). In order to manually identify classification rules for the KE approach as well as to automatically learn a ML model, a training-set is defined. The performance of each method is determined by measuring precision and recall on a pre-defined test-set.

In case of the ML method we also examine which text representation (binary text frequency vs. tf-measure, feature-expansion vs. no feature-expansion, etc.) works best, i.e. has the highest precision and recall, for the investigated document corpus.

8.2.1 Training and test set

The available data – consisting of an input vector (documents) and an answer vector (class) – is separated into two sets of examples: a training set and a test set. The training set is used to form the learned hypothesis (determine its parameters) while the test set is used to evaluate the accuracy of the hypothesis (holding the parameters constant) [Mitchell 1997]. Performance on the training set tells us only that the learning algorithm is able to memorize the given examples. It is not an indicator for the performance on unseen data. Therefore, the separate test set is supplied.

The training and test set are stratified, i.e. sampling is done so that the relative proportions of each class are about the same in both sets (Table 8-9). The training set of step 1 (noise filtering) consists of 2,549 manually annotated documents and the test set consists of 179 documents. In step 2 (topic classification) we have a total of 1,963 documents in the training set and 100 documents in the test set. Arguably, the test set of step 2 is with only 100 samples at its minimum. We decided not to build a larger test set due to the small amount of portfolio documents which are available in the category “Portfolio”.

Table 8-9: Training- and test-set.

	Category	Training-Set	Test-Set
Step 1	Signal	1,835	130
	Noise	714	49
		2,549	179
Step 2	Agenda	625	35
	Minutes	275	13
	Memo	321	12
	Portfolio	46	6
	SOP	243	14
	Pre-Clinical Study Report	453	20
		1,963	100

8.2.2 Optimization of the ML text classifier

The aim of this section is to determine which text representation works best for the chosen SVM learning approach (Chapter 6.3.2; cf. Fig. 6-8). The initial step of the text processing pipeline is tokenization which splits the text into individual tokens using a whitespace tokenizer. Next, a lower case filter is applied so that complexity is reduced while almost no significant information is lost. Tokenization and lower case filtering are always applied, i.e. we focus on testing a) which term weighting schema (binary, term frequency, and inverse document frequency) gives the best results, b) whether eliminating stop words and stemming improves performance, and c) whether feature expansion with named entities brings any improvement.

The performance is measured on the training set of step 2 (topic classification) using a stratified 10-fold cross validation. Notice that optimization creates a bias on the training set. Therefore, we ignore the test set during optimization so that an unbiased performance estimate is received afterwards.

In order to evaluate the performance we consider the following statistics: precision and recall (Chapter 2.7), as well as the kappa statistics. The kappa statistics [Agresti 2002] measures the strength of agreement between two observers. In our case it is the agreement between the SVM's model output and the expected answer as supplied in the training set. The value of kappa is the higher the stronger the agreement is. Kappa equals 1.0 when there is perfect agreement and it equals 0 when the agreement equals that expected by chance. Negative values occur when agreement is weaker than expected by chance – but this rarely happens. The kappa statistics thus shows the performance in one number. Alternatively one could use the F-measure, which calculates the arithmetic mean of precision and recall.

The results are summarized in Table 8-10. Using solely a binary word weight (i.e. a value of 1 if the word occurs in the text and a value of 0 if the word does not occur) delivers already good results. The classification accuracy is improved by approximately 1% if the term frequency measure is used. In particular, the kappa statistics improves from 0.92 to 0.93 as does the precision from 93.93% to 94.48%.

Combining the term frequency with the inverse-document-frequency does not significantly improve performance. Therefore, we decided to use only the term frequency for calculating the term weights.

Eliminating stop words from the text also improved classification performance by about 1%. The amount of correctly classified instances rose from 94.84% to 95.74%.

Stemming on the other hand had a negative impact on performance as the performance dropped by about 1%.

The incorporation of feature expansion improved performance only slightly. The percentage of correctly classified instances grew by 0.06% while the kappa statistics did not change.

We also examined how text classification performs if just the extracted features are considered. Interestingly, it performs quite well, suggesting that the KE approach which is based only on features could compete with the ML approach.

Table 8-10: Text categorization performance for different text processing pipelines (whitespace tokenization and lower case filtering was always applied). Optimization was conducted on the training set using a stratified 10-fold cross validation so that no bias on the test set evaluation is introduced.

	Binary word count	TF	TF*IDF	TF +Stop words elimination	TF +Stop words eliminiation +Stemming	TF +Stop words elimination +Features	Just features using TF
Correctly Classified Instances	93.93%	94.84%	94.89%	95.74%	94.89%	95.80%	90.35%
Incorrectly classified instances	6.07%	5.16%	5.11%	4.26%	5.11%	4.20%	9.65%
Kappa statistics	0.92	0.93	0.93	0.95	0.93	0.95	0.88

Wrapping it up, the applied text processing pipeline consists of a whitespace tokenizer, a lower case filter, stop word elimination, and feature expansion. Term weights are calculated using the term frequency measure. This pipeline is also applied for the step 1 (noise filtering) classification task.

8.2.3 Knowledge engineering vs. machine learning

From now on the test set is used for evaluation.

The ML classifier performs very well in both classification tasks (Table 8-11). The percentage of correctly classified instances is 96.65% for step 1 and 95% for step 2. The kappa statistics is 0.92 for step 1 and 0.94 for step 2.

Table 8-11: ML Classification performance of step 1 and step 2.

	Step 1	**Step 2**
Correctly Classified Instances	96.65%	95.0%
Incorrectly classified instances	3.35%	5.0%
Kappa statistics	0.92	0.94

The confusion matrix shows that the classifier of step 1 misclassified signal as noise in five cases (Table 8-12). The reverse relation is much less prevalent as only one such case could be observed during validation.

Table 8-12: Confusion matrix of ML step 1.

classified as ->	**a**	**b**
a = Noise	48	1
b = Signal	5	125

In case of step 2, the confusion matrix shows that most misclassifications revolve around documents of the type Minutes (Table 8-13). This was expected, because even for a human annotator the classes Agenda, Memo, and Minutes are not always clearly separable. The reason is their similar structure. Further, people sometimes use the term “agenda” when referring to a document of type Minutes and vice versa.

Table 8-13: Confusion matrix of ML step 2.

classified as ->	**a**	**b**	**c**	**d**	**e**	**f**
a = Agenda	34	0	0	0	1	0
b = Memo	1	11	0	0	0	0
c = Minutes	2	1	10	0	0	0
d = Portfolio	0	0	0	6	0	0
e = SOP	0	0	0	0	14	0
f = Pre-clinical study report	0	0	0	0	0	20

Now that we have shown the performance of ML, we next compare its performance to the KE approach. The classification performance of the KE approach and the ML approach are summarized in Table 8-14. Regarding step 1 (the separation of signal and noise), our results show, that both, the KE approach and the ML approach, have a precision of about 1.0 for classifying “Signal”. The recall however, is quite different as the KE approach has a recall level of 0.72 while the machine learning approach has a recall value of 0.96. Hence, ML outperforms the KE approach in step 1. Similarly, in step 2 the ML approach performs better on average than the KE approach. In detail, for documents of type Agenda the ML method should be chosen as it provides the better harmonic mean of precision and recall. In contrast, documents of type Memo are better categorized by the KE approach. In just the opposite manner, the ML approach provides a better performance for documents of the type Minutes. Portfolio documents are classified perfectly by both approaches. The last two categories, SOP and pre-clinical study report, can only be classified by the ML method as no corresponding rules have been provided.

The main reason why the ML method outperforms the KB method is that the latter provides a lower recall level. Nonetheless, this result was expectable because the

rules have been engineered very strictly so that no false positive hits are returned. It is also due to the fact that we can not express per se fuzziness in the rules. In addition it would also be quite awkward to express rules such as “The term ‘project’ must occur with a frequency of 0.23 in the document”, leading the idea of human readable and interpretable domain knowledge ad absurdum.

Table 8-14: Performance comparison between KB and ML.

		KB		ML	
		Precision	**Recall**	**Precision**	**Recall**
Step 1	Signal	1.0	0.72	0.99	0.96
	Noise	0.58	1.0	0.91	0.98
Step 2	Agenda	1.0	0.72	0.92	0.97
	Memo	0.92	1.0	0.92	0.92
	Minutes	1.0	0.62	1.0	0.77
	Portfolio	1.0	1.0	1.0	1.0
	SOP	-	-	0.93	1.0
	Pre-Clinical Study Report	-	-	1.0	1.0

8.2.4 Discussion

The results show that both methods perform quite well on the considered document categories (Chapter 6.3.1). However, the ML approach performs on average better than the KE approach. Further, KE is only feasible for documents which have a precise structure. Otherwise, defining the classification rules becomes a tedious task for the human domain expert so that ML should be preferred. Another reason which speaks in favor of ML is the faster execution time – in our case it was the factor 10 faster than KE. Nonetheless, the applied SVM algorithm lacks the convenient explanation feature a rule-based approach offers: it is difficult to understand why a SVM decides to classify a document into a specific class.

Because the considered documents could be classified accurately, we assume that the other categories of the classification ontology can be classified with a similar performance. Defining a training- and test set for the other categories is the task of future work. In addition, the ML parameters, i.e. the SVM kernel, could be further optimized.

Not yet implemented, but arguably an important feature, would be the incorporation of feedback into the classification process. In principle one could offer the users two buttons in YASA by which they could vote if the document was classified correctly or not. We could even offer the possibility to label unclassified documents so that the training set is expanded. In both cases, the votes should be treated with care. Contradictions such as the re-classifications of documents relevant for the support vectors could lead to a significant impact on the classification performance. Regarding the current usage of YASA, we suggest that feedback votes are collected and manually asserted. In case the usage grows further, we suggest collecting votes and automatically applying the feedback as soon as a certain threshold of feedback has been reached.

8.3 Retrieval performance evaluation using click-through data

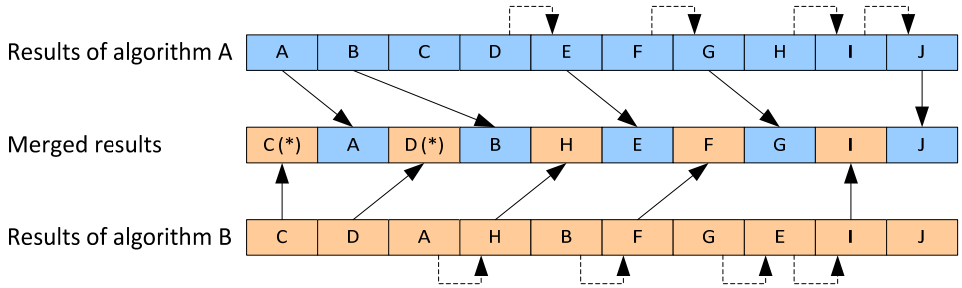
The goal of this evaluation is to estimate the retrieval performance of the ranking algorithms (Chapter 6.4) supplied in the prototype. First, we analyze if incorporating knowledge-based adaptation (Chapter 6.4.1) improves retrieval performance over Lucene's baseline ranking. Second, we test whether log-based adaptation (Chapter 6.4.3) brings any performance improvement over the baseline ranking. In the following, we denote Lucene's baseline ranking algorithm as (B; Baseline). The knowledge-based adaptation approach is denoted by (BC; Baseline with context), and the log-based adaptation method is denoted by (BF; Baseline with log-based feedback). Notice that Lucene's baseline ranking (B) is based on the vector space model introduced in Chapter 2.2.2.

Traditionally, retrieval performance is evaluated by expert judgment using precision and recall. While evaluation by experts provides a great quality it has the major drawback that the availability of experts does not scale well with large and dynamic repositories. First, the more data is indexed, the more data must be evaluated. Second, the more a repository fluctuates (page modifications, deletions and insertions) the more frequently evaluations must be conducted. A common approach to deal with these issues is to focus manual assessment on the top documents, as these are considered to contain the most important hits.

Alternatively, evaluation of retrieval performance can be conducted by means of log data, as is for example described in [Joachims 2003]. The idea is to use implicit feedback gathered by observing clicking behavior. Such click-through data is collected as follows: A user types a query into a search interface and the query is sent to the search engine. Then, a ranking is returned and the user selects the item of interest. The selection, i.e. the association between a query and a clicked item is recorded. This principle is used by [Joachims 2003] to compare ranking of results.

Two rankings are merged similar to a zipper (Fig. 8-7). If a user types a query into a search interface, the query is sent to both search engines. Then, the ranked results of both search engines are merged and the results are returned to the user. Finally, a result is selected and recorded. From the logs we can see which ranking algorithm provided the hit and thus conclude a preference towards one or the other algorithm.

Fig. 8-7: Merging the results of two different retrieval functions.
Clicked items are marked with a star.



In order to prevent a bias towards any ranking algorithm, the first item is selected at random. Thus, in one case the zipping begins with algorithm A and in the other case it begins with algorithm B. In case of Fig. 8-7, the zipping begins with the results from algorithm B. In case duplicates are encountered at any position, the merging continues with the next item of the current ranking list. Notice that if two different ranking algorithms are applied on the same index, it is likely that the result set is identical and that only the order of the items differs. Therefore, when recording the clicks on the merged result list, the items' position in the original rankings must also be tracked. In the given example, the searcher clicked item C on position 1 and item D on position 3. Because of the assumption, that a user scans the results from top to bottom, we can conclude that he has seen item A and item B of algorithm A as well as item C and item D of algorithm B. The number of items which a user would have seen in the original list is determined by taking the position of the last clicked item, dividing the value by two and rounding it up. In our case, the last position on the merged set was 3, and thus the number of items viewed in the original ranking was 2. Because item C as well as item B occurs within the top two of algorithm B but not of algorithm A we count two votes for algorithm B and zero votes for algorithm A.

Joachims also conducted an evaluation in which he compared the click-through approach with the judgment of experts. The results of the click-through data were found to closely follow the relevance judgments [Joachims 2003]. Thus, according to this study, the click-through approach gives meaningful information about the quality of two retrieval functions.

8.3.1 Design

The method's idea of using implicit feedback for evaluation enabled us to conduct the experiment online without the user's awareness. We implemented a merging algorithm as described in [Joachims 2003] into YASA. The user interface remained unchanged – only the ranking algorithm was replaced by the merged version. Therefore, users were not able to deduce by which algorithm an item was retrieved.

The data collected in our experimental setup is discrete and ordinal: for each query we have the total numbers of clicks on items from algorithm A and B. Therefore, the popular Student's t-test [Gosset 1908] should not be applied as it is designed for continuous data having a normal distribution. A suitable test is the Wilcoxon signed-rank test [Sidney 1957; Wilcoxon 1945] as it involves the comparison of differences between measurements. In contrast to the t-test it does not require an assumption about the distribution.

Further, we also conduct McNemar's test [McNemar 1947] which requires even less assumptions than the Wilcoxon signed rank test.

The null hypothesis H_0 is defined as "no preference towards a ranking algorithm". The alternative hypothesis H_1 is defined as "there is a preference towards one of the ranking algorithms". Further, we use a 95% confidence level for verifying / falsifying a hypothesis.

8.3.2 Baseline ranking vs. baseline ranking with feedback

The logs supplied to the (BF) algorithm were collected since the introduction of the prototype, i.e. over a period of about 18 months. The log repository of the PRPZ-Share contained in July '09 feedback data for 11,664 documents. Nonetheless, not all data could be used during the evaluation period. Log-data is coupled to a document by a mere URL. Therefore, if the URL changes (deletion, movement to another folder, or renaming), then the log data becomes worthless. After removing all dead links a total of 7,481 documents containing feedback data remained (Table 8-15). Arguably, 35% of dead links is a considerable amount. For the future it would thus be wise to track documents not only by a URL but also by means of a hash key so that document movement, renaming, and modification can be tracked.

The average number of clicks per feedback document is 1.17. In contrast, the average number of clicks per document (on the entire corpus of the PRPZ-Share) is 0.013. Similarly, the log data comprises only 3,598 different terms – this is only 0.004% of the total PRPZ-Share vocabulary size. The amount of feedback gathered for the PRPZ-Share is thus quite small, making the application of the (BF) ranking approach a challenge.

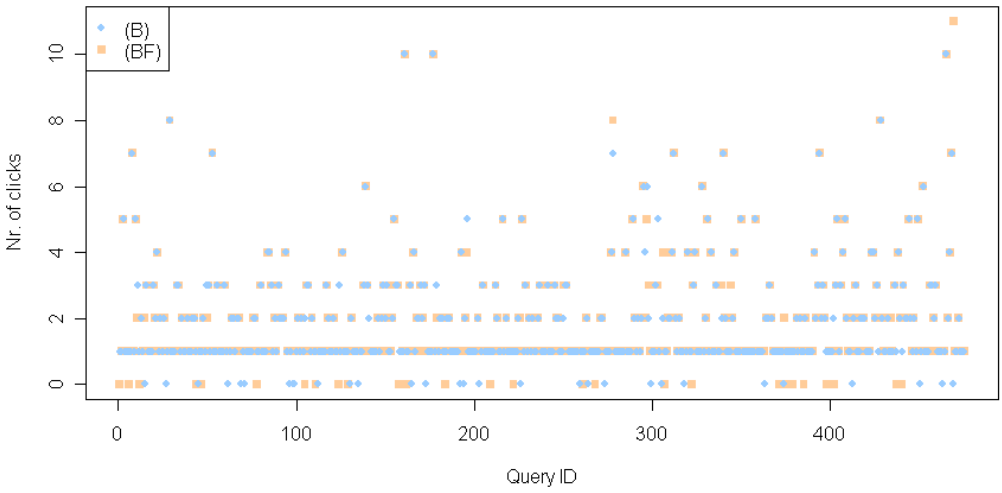
Table 8-15: Size of the PRPZ-Share logs at the time the analysis was conducted.

Number of documents with feedback data	Total number of clicks	Avg. clicks per document (feedback corpus)	Avg. clicks per document (entire PRPZ-Share corpus)	Size of feedback vocabulary (nr. of terms)
7,481	8,781	1.17	0.013	3,598

The experiment (comparing the baseline algorithm with the feedback enhanced baseline algorithm) was conducted over a period of one month in July '09. During this period a total of 940 click-through entries originating from 475 queries have been recorded for evaluation. Queries which have no clicks are ignored so that an average number of 1.97 clicks per query are counted. As already outlined, the available feedback data is sparse. Nonetheless, we counted 75 queries (or 15%) for which feedback data was available. The other 400 queries contained no feedback data so that (B) and (BF) delivered the same ranking of results.

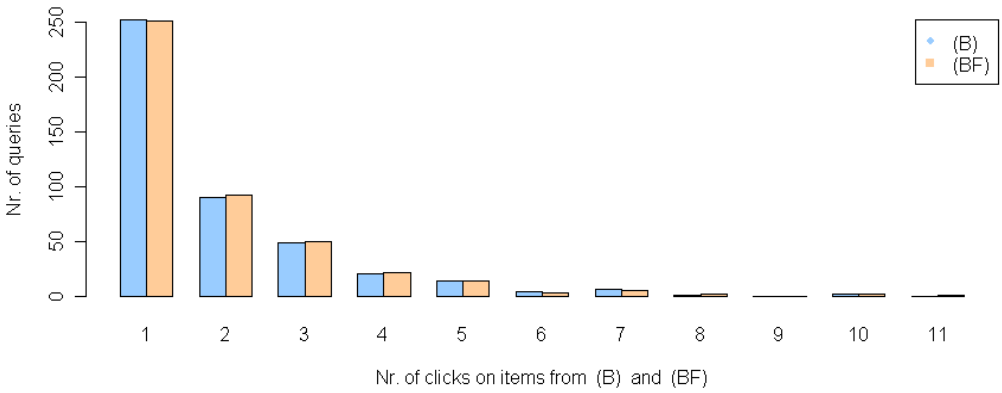
An illustration of the collected data is shown in Fig. 8-8. The mere eye can't see any significant preference towards one or the other ranking algorithm.

Fig. 8-8: Click distribution of (B) and (BF)



A comparison of the frequency distribution of (B) and (BF) does also not provide any visual evidence towards a preference of a ranking algorithm (Fig. 8-9). The figure shows the amount of clicks on items from (B) and (BF) grouped by the total number of clicks per query. The frequencies are almost identical for (B) and (BF).

Fig. 8-9: Frequency of clicks per query of (B) and (BF)



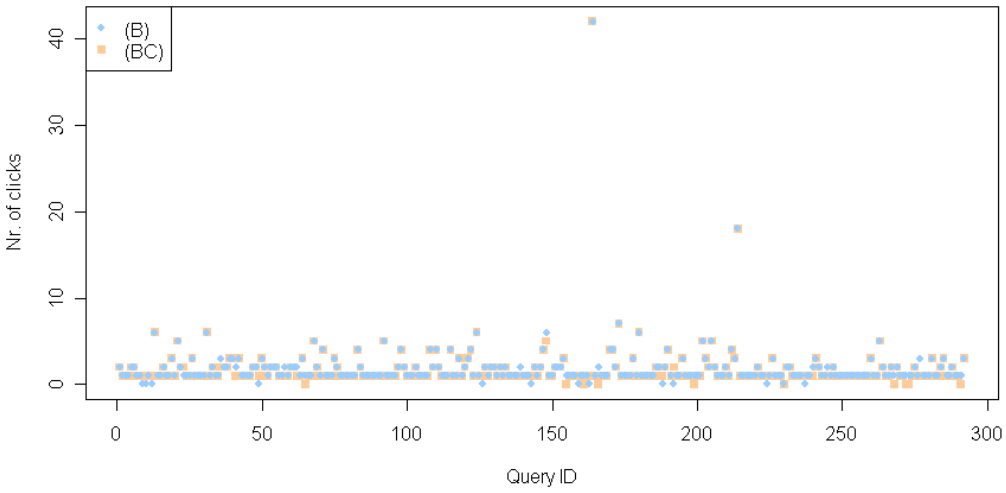
The Wilcoxon signed-rank test returns a p-value of 0.382. Thus, given a confidence level of 0.95 we can not decline H_0. Similarly, the McNemar statistics returns a p-value of 1.0, so that H_0 can not be declined. Hence, neither a preference of the searchers towards the baseline ranking, nor towards the baseline ranking with feedback can be detected.

8.3.3 Baseline ranking vs. baseline ranking with context

The analysis conducted here is based on log data which was collected over a period of three weeks in May '09. A total of 568 click-through entries originated from 293 queries are recorded. Thus, we encountered on average 1.93 clicks per query.

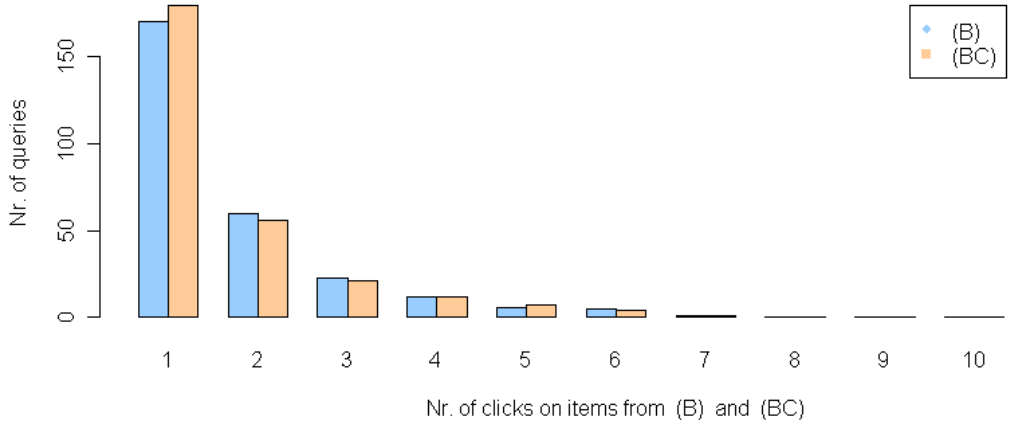
The data used for evaluation is visualized in Fig. 8-10. The mere eye can't detect any preference towards a ranking algorithm when looking at the picture.

Fig. 8-10: Click distribution of (B) and (BC) after removal of duplicates



The frequency distribution of (B) and (BC) (Fig. 8-11) on the other hand shows a slight tendency at least for queries having only one click. In this case, items from (BC) are preferred to items from (B). For queries having two or more clicks, almost no difference can be observed.

Fig. 8-11: Histogram of clicks per query of (B) and (BC) after removal of duplicates



The Wilcoxon signed-rank test returns a p-value of 0.832 and the McNemar test returns a p-value of 0.23. Therefore, at a 0.95 confidence level we can not conclude that H_1 holds and thus H_0 can not be declined. In other words, no statistical significant preference towards any ranking algorithm can be detected.

8.3.4 Discussion

In the first experiment we compared (B) with (BF). The main finding is that no significant preference towards (B) or (BF) could be detected. In fact, they performed almost the same.

A key question is: How much feedback data do we need so that (BF) outperforms (B)? This question can't be answered easily because it depends on the repository. For some repositories, few feedback data could suffice while others might require much more. Actually, [Hawking et al. 2006] investigated exactly this issue. They examined

if the ranking of results can be improved in small scale web-search by feedback data, and how much feedback data is needed to achieve a significant improvement. Their finding was that small amounts of log data can improve performance. However, the amount of log data required is strongly dependent on the query type. The authors discriminate "popular queries" (frequently executed queries), and "sitemap queries" (queries are derived from a website's sitemap; the entries become queries and the links become the corresponding best answer). Given a "stock exchange" repository, consisting of $2.2x10^4$ pages, as few as 4000 clicks (i.e. 0.18 clicks per page) are sufficient to reach an asymptotic limit of improvement. For sitemap queries however, it takes over $1x10^5$ clicks to reach an asymptotic point and the quality is still less than for popular queries.

Sitemap queries can not be applied for the file shares and are thus not considered. Popular queries on the other hand can be applied. In order to verify if an improvement can be achieved, we re-run the experiment (B) vs. (BF) and restricted the evaluation to popular queries. In particular, we considered only queries which have been transmitted at least 5 times. In effect, only 28 queries remained. Despite the restriction to popular queries no significant improvement could be observed, i.e. the hypothesis H_0 can still not be declined.

Hawking et al. demonstrated a substantial improvement over the baseline for popular queries if logs are exploited. Based on their findings, they assumed that the greatest potential gain lies in non-web environments such as the PRPZ-Share. In our study we were not able to detect any significant improvement in non-web environments. The main problem is without a doubt the small click per document ratio having a value of 0.013. It is likely that at least ten times more log data is needed in order to see a significant improvement over the baseline.

Consider the fact that the repository is constantly growing and that during the period of one year, a click per document ratio of only 0.013 has been achieved. Further, assume that the amount of employees stays constant. Then, it is questionable if the ratio will ever reach a level which suffices to see a significant improvement. The question is why do we observe this discrepancy at all? We believe it is due to the hierarchical organizational structure of the company. On the one hand, the majority of data is produced by employees who conduct the experiments (such as lab workers). On the other hand, the majority of data is consumed by the decision makers (i.e. the group and department leaders). Because there are about ten times fewer consumers (department leaders) than producers (lab workers) and because research is a highly dynamic area, the ratio will probably always have a low value. We can thus conclude that log-based feedback, as a method for improving ranking performance, is not well suited for similar professional environments.

Regarding the second experiment, no statistical significant preference towards ranking algorithm (B) or (BC) could be observed. Just the tendency could be detected that people accessing only one document within a query, are slightly more often clicking on items from (BC) than from (B).

Indeed, our assumption that people prefer results from their context than others might be wrong. However, we believe this is unlikely because the adaptation was tuned so that only similar hits are separated (Chapter 6.5.2). Another reason might be that adaptation has only rarely a strong effect on the rankings. This would be the case if the returned results have a similarity curve with a strong decay. Investigation of the query logs could give valuable information about the nature of the similarity profiles. This is to be investigated as part of future work.

Last, we want to stress that a wrong parameter setting might be the reason why no significant performance improvement of (BF) or (BC) over the baseline could be observed. Indeed, even though we fine-tuned the ranking parameters by trial and error it might not be what the average searcher expects. An automatic optimization of the parameters would be possible. For instance, one could do a search over the parameter space and test for which settings the ranking of the clicked items are improved. Conducting such an optimization is also part of future work.

8.4 Controlled experiments

In this section we describe several controlled experiments in order to verify various aspects such as

a) is search for information preferred over browse for information,
b) are facets (i.e. classification) helpful, and
c) is one single entry-point to internal & external information helpful

All these goals revolve around the following hypothesis: “A significant improvement in satisfying the scientists’ information needs is achieved by the approaches implemented in the YASA prototype”. In order to test the hypothesis a task-based evaluation of the YASA prototype is conducted. A task-based evaluation is an evaluation method in which test persons are given a couple of tasks which they have to accomplish. The data collected during task processing is used to conclude whether the hypothesis is valid or not.

Next, the evaluation process and methodology will be described, which are similar to [Franz et al. 2008]. Then, the test persons are introduced. Following that the developed tasks and the test system are described. Finally, the evaluation is conducted and the results are presented.

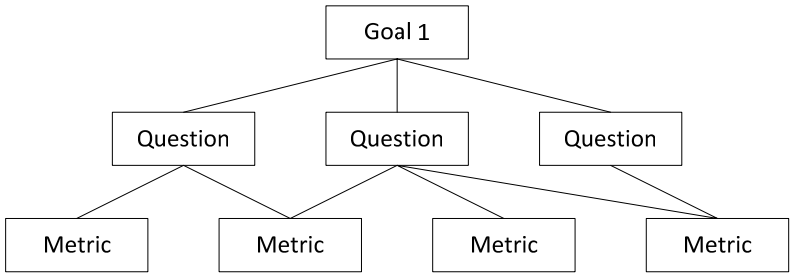
8.4.1 Evaluation process and methodology

The evaluation process consists of the three phases: briefing, observation, and feedback. In the first phase each test person is briefly introduced to the test system and to the evaluation procedure. This step is very important in order to remove any bias caused by (un)experienced users. In the second phase, the users are not given any assistance and we just observe how they solve the given tasks. In the last phase subjective feedback is gathered from the test persons in form of questionnaires.

In order to evaluate the prototype we use the goal question metric (GQM) paradigm [Basili et al. 1994]. The GQM paradigm is a top-down approach (Fig. 8-12) which allows evaluating the quality of specific processes and products of a software system. First, a goal is defined for an object which can be product deliverables,

product specifications, processes, or resources. Then, a set of questions are formulated, which characterize the object of measurement. Finally, a metric is associated with every question in order to get a quantitative measurement. The data on which the metrics are applied can be objective or subjective.

Fig. 8-12: Hierarchical structure of the GQM model



In our case, the goal is formulated as “a significant improvement in satisfaction of the scientists’ information need”. The questions revolving around this goal are about effectiveness (how many tasks are completed?), about efficiency (how fast are the tasks completed?), and about satisfaction (how pleased are the users with the prototype?). In Table 8-16 an overview of the used goal, questions, methods, and metrics is given.

Table 8-16: Goal, questions, methods and metrics

Goal	**Question**	**Method**	**Metric**
A significant improvement in satisfying the scientists’ information needs.	How effective can users complete the tasks?	Objective	Success Rate
	How efficient can users complete the tasks?	Objective	Execution time Mouse movement
	How satisfied are users with the systems?	Subjective	Questionnaire Interview

The metrics can be distinguished in objective and subjective methods. Effectiveness and efficiency are evaluated using objective methods, while user satisfaction is evaluated using subjective questionnaires and interviews.

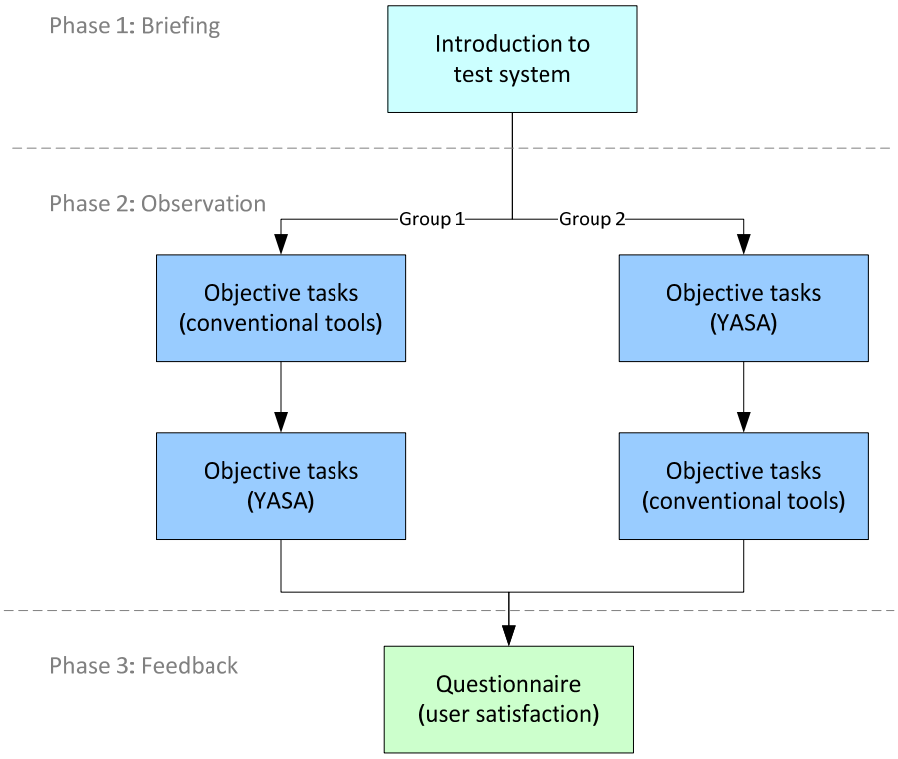
The evaluation process consists of three phases (Fig. 8-13). In the first phase (briefing) the test person is introduced to the test system.

In the second phase (observation) we determine whether the participants solve the objective tasks more efficiently using conventional tools or YASA. Because we expect only few employees to volunteer for the evaluation, we let each test person conduct the objective tasks twice: once using conventional tools and once using YASA. The test persons were aware about this procedure. In order to exclude a bias in the sequence of used tools we randomly split the test persons into two groups. The first group must begin with conventional tools (any state of the art tool except YASA) and then YASA, while the second group uses the tools in reverse order. By letting two groups solve the tasks using the opposite tool order, we might not only exclude any potential bias but also detect interesting relationships. For instance it might be that

if the objective tasks are first conducted with YASA, the tasks are solved much faster with conventional tools than vice versa.

In the third phase (feedback), the user is given a questionnaire whose purpose is to capture the subjective opinion about the prototype's applied principles.

Fig. 8-13: Evaluation process overview.



Typically, a scientist at Roche has only little time. For that reason, a test person should finish the evaluation process in an average time of 30 minutes. The tasks and questionnaire (Chapter 8.4.3) will therefore be designed so that the three phases stick to the following time slots: The introduction to the test system in phase 1 is done in 5 minutes; the objective tasks of phase 2 should be finished in 20 minutes; and the questionnaire of phase 3 should be done in 5 minutes.

8.4.2 Test persons

We asked 20 persons of the Roche Pharma Research department of Penzberg to participate in our evaluation. The persons were selected so that a representative distribution is achieved. We asked employees from various departments for their participation in the user study. In particular, the participants work in departments such as "Protein Engineering", "Biological Screening", "Oncology Discovery", "Cell Biology", "Molecular Biology", "Patent Division", "Statistics", "Bioinformatics", and "Informatics".

Among the 20 participants, there were 19 employees (including 6 group leaders) and one student. The highest academic degrees distribute as follows among them (Table

8-17): a bachelor degree is owned by 1 test person, a diploma by 3 persons, and a Ph.D. by 15 participants.

Regarding the participants' profession (Table 8-18), we counted 6 biologists and 4 chemists. The areas bioinformatics, informatics, and statistics were represented by three participants in each case.

The majority of the participants, i.e. 16 persons, had used YASA before while the others used YASA during the evaluation for the first time.

Table 8-17: Educational background of participants.

Education	Amount
Bioinformatics	3
Biologist	6
Chemist	4
Informatics	3
Statistics	3
Student	1

Table 8-18: Academic degree

Highest academic degree	Amount
Bachelor	1
Diploma	3
Ph.D.	15
Other	1

8.4.3 Designing the tasks

We created a total of 6 different tasks for the observation phase (Table 8-19). The developed tasks are retrieval tasks, i.e. the test persons have to look up some information. Depending on the task one or more answers can be valid. The main focus was to create a realistic set of heterogeneous tasks which reflects well typical queries on the present corpus. We decided to accomplish this by careful examining 3 month of log data. In a next step we slightly modified the observed queries while maintaining the query type. For instance, we noticed that people often look up project related information. As a result we created task number 5 in which project related information must be gathered.

Table 8-19: List of tasks. *: Baseline task. #: A gene name not to be disclosed.

Task ID	Description
(*) 1	Find a Wikipedia article about Herceptin
2	Find the location and phone number of the company's medical doctor at Roche in Penzberg
3	Get the full text of a publication named: A Breast Cancer Risk Haplotype in the Caspase-8 Gene
4	Find a location with literature (publications, presentations, posters) about YASA (Your Adaptive Search Agent)
5	Find the main folder on the PRPZ-Share where literature (studies, reviews, etc.) about MyGene# (also known as ---) is consolidated
6	Find the intranet homepage of the application "Prous Integrity"

In addition to the 6 objective tasks we also ask the test persons to participate in a questionnaire. While the tasks were used to determine the performance of YASA compared to conventional tools, the questionnaire aimed at determining quality characteristics which can't be captured well by mere numbers. Therefore, we formulated 39 questions (Table 8-20) which cover the quality of the objective tasks, various aspects of search, and several details about YASA – especially faceted navigation and source integration. Further, we acquired some demographic data of the test persons.

Table 8-20: List of questions used in the questionnaire.

Question Category	**ID**	**Question**	**Rating Scale**
Objective Tasks	1	Working on the previous tasks was tedious / cumbersome	IsoMetrics
	2	The previous tasks correspond to task types that I also need to do for my work	IsoMetrics
General	3	Which tool would you use first to access internal information?	Pharma Search; Dia Search; Windows Explorer; TagIt; YASA; Other
	4	Which search engine would you use first to find external information?	Google; Yahoo; Bing!; PubMed; TagIt; Other
	5	Why would you choose a search engine?	Just a habbit, I like the layout; Size of the repository; Quality of search results; Speed; Other
YASA General	6	Have you used YASA before?	yes; no
	7	How often do you use YASA per week?	9+, 6-8, 3-5, 1-2,0
	8	Search results in YASA are relevant to my query	IsoMetrics
	9	I prefer search results from my department	IsoMetrics
	10	I often use YASA to find documents of other groups or departments in Pharma Research	IsoMetrics
	11	The time it takes for the search engine to return its information is too long	IsoMetrics
	12	The user interface of YASA is intuitive	IsoMetrics
	13	I immediately find what I am looking for	IsoMetrics
YASA Facets	14	The refine search options (facets) help me to find the information I need	IsoMetrics
	15	The *my files* facet is helpful	IsoMetrics
	16	The *file format* facet is helpful	IsoMetrics
	17	The *year* and *month* facets are helpful	IsoMetrics
	18	The *project* facets are helpful	IsoMetrics
	19	The *department* facet is helpful	IsoMetrics
	20	The *category* facet is helpful	IsoMetrics
	21	I would like to have more refine search options	IsoMetrics
	22	I do not find information faster using facets	IsoMetrics

YASA Integration	23	It is useful to have access to internal & external information from one search tool	IsoMetrics
	24	The integration of Tagged Documents into YASA is useful	IsoMetrics
	25	I actively use TagIt for my personal information management	IsoMetrics
	26	The integration of PubMed into YASA is useful	IsoMetrics
	27	The integration of Applications into YASA is useful	IsoMetrics
	28	The integration of Wikipedia into YASA is useful	IsoMetrics
	29	The integration of the Internet catalogue is useful	IsoMetrics
	30	The integration of the Telephone Book is useful	IsoMetrics
	31	The integration of the PRPZ-WebSite is useful	IsoMetrics
	32	The integration of databases is useful	IsoMetrics
	33	The integration of the PRPZ-Share is useful	IsoMetrics
	34	Would you like to be able to search your personal U-Drive with YASA as well?	IsoMetrics
	35	Would you like to be able to search other areas such as Pharma Technical Development as well?	IsoMetrics
User	36	What is your profession?	Biologist; Chemist; Informatics; Bioinformatics; Statistics; Student; Other
	37	What is your highest academic degree?	Bachelor; Master; Diploma; Ph.D.; Other
	38	Is finding information part of your daily work?	yes; no
	39	How experienced do you consider yourself regarding search for information?	Novice; Advanced beginner; Competent; Proficient; The Expert

The questionnaire is mainly conducted using the IsoMetrics [Gediga & Hamborg 1999] usability inventory which is a summative as well as formative approach in software evaluation. Each question is assessed using a 5 point rating scale, starting from 1 (“predominantly disagree”) to 5 (“predominantly agree”). In addition a “no opinion” field is supplied to reduce arbitrary answers. Beside the IsoMetrics, we make use of various multiple-choice answer lists, such as yes / no.

Finally, the users were allowed to make free comments about the prototype and the principles used.

8.4.4 Test system

At Roche, desktop computers and laptops are supplied by a single vendor. Further, a common office and desktop environment is employed. This setup has the benefit, that hardware and software systems are very homogenous. Workers can thus easily switch machines without the need to get accommodated to a new system. We recognize this fact and decided to build up our test system on top of a regular Roche desktop computer. The only difference to an out-of-the-box system is that some extra software is installed for conducting the evaluation and monitoring purposes.

This extra software is called “EasyEval”. The purpose of EasyEval is twofold. On the one hand side it provides the objective tasks of the second phase (observation) to the user. On the other hand it monitors the user’s activities.

The tasks are presented sequentially to the current user. He has to press start when he starts processing a task and he has to press stop when he has finished a task. Once a task has been completed, the user can’t go back, i.e. the EasyEval application is unidirectional.

The elapsed time for each task to finish is measured and recorded. In addition we measure the mouse movement in pixels. We did not find any other useful metrics to be considered. Measuring the number of “windows switches” for instance does not make sense because in one case (conventional tools) the participants can use any tool and could thus freely switch between applications while in the other case they are forced to use only one tool (YASA).

8.4.5 Conducting the evaluation

At the beginning of an experiment session the user is briefly introduced to the evaluation procedure. In order to make the user feel comfortable with the test system we start EasyEval with a demo of two tasks so that the participant can see how the actual test will proceed. During this demo they are also introduced to YASA regardless of their prior knowledge about it. Regarding the conventional tools, we briefly point out which options exist but do not further introduce all of them as we consider those as a part of their daily work. Conventional tools are e.g. the windows file explorer, mail, Google, the PRPZ-WebSite, other intranet search engines, etc.

In addition, we also point out that we do not set an explicit time limit for the objective tasks. Hence, participants have to decide on their own how much time they are willing to invest for answering a query.

After the introduction, we randomly assign a user to a group (cf. Fig. 8-13). During the observation phase, the user is not given any assistance so that any bias is avoided. Once the observation phase is finished, we conduct the questionnaire with the participant. During the questionnaire we give feedback in case the participant does not understand the meaning of a question.

8.4.6 Results of the observation phase

In the observation phase, the test persons have to provide an answer to the objective tasks. The given answers might be wrong and must therefore be handled accordingly. We decided to penalize wrong or missing answers by introducing a threshold value. Arguably, the choice of a threshold value could influence the results. It should thus not be set to high as otherwise wrong answers might bias the results significantly.

In order to find a good threshold, we examined the distribution of the time metrics and the mouse movement metric, respectively. We observed that the maximum time to successfully finish a task was 540 seconds. In fact only one event could be

observed in which test persons needed longer than 500 seconds to provide the correct answer to a task. The second slowest time was 440 seconds. Therefore, we set the threshold value to 500 seconds, i.e. wrong answers as well as correct answers which needed more time than the given threshold are set to 500 seconds. Similarly, the maximal mouse movement of a correct answer was about 900,000 pixels. Hence, we set the threshold value to 1 million pixels.

The percentages of wrong answers per task are summarized in Table 8-21. Task 1 "Find a Wikipedia article…" and task 3 "Get the full text of a publication named…" can be considered as easy tasks because every participant solved them correctly. Similarly, task 6 "Find the intranet homepage of the application…" is an easy task as almost all participants provided the correct answer. Task 2 "Find the location and phone number of the company's medical doctor…" is of medium difficulty because 15% of the test persons were not able to solve it using conventional tools and using YASA. Task 4 "Find a location with literature […] about YASA…" and task 5 "Find the main folder […] where literature […] about MyGene is consolidated" are arguably difficult tasks. In case of task 4, 35% of the participants were not able to find the correct answer using conventional tools. With YASA the fraction of wrong answers is 30%. Regarding task 5, 60% of the test persons provided a wrong answer if conventional tools were used. However, only 20% of the answers were wrong in case YASA was applied.

Table 8-21: Fraction of wrong answers per task.

Task ID	**Wrong answers using conventional tools**		**Wrong answers using YASA**	
	relative	absolute	relative	absolute
1	-	-	-	-
2	15%	3	15%	3
3	-	-	-	-
4	35%	7	30%	6
5	60%	12	20%	4
6	5%	1	-	-

Next, we investigate whether the tasks can be answered faster (time metric) after they have been solved once. In case this effect can be observed, we are also interested whether it occurs for both, YASA and conventional tools.

The results show that the tasks are finished considerably faster with YASA if they were previously answered with conventional tools (Group 1; Table 8-22). The reverse effect can not be observed (Group 2). We can thus conclude that if results have been found with conventional tools first they can be found much faster with a search engine. On the other hand, finding an answer with YASA does not help to find the same answer faster with conventional tools.

The total performance result reads as follows: On average (median) a participant needs 117 (64) seconds to answer a question with YASA and 173 (85) seconds to answer a question with conventional tools (Table 8-22).

Table 8-22: Influence of the order (Group 1 vs. Group 2) on the performance.

	Group 1 (Conventional ->YASA)		**Group 2 (YASA -> Conventional)**		**Total**	
	Conventional	**YASA**	**YASA**	**Conventional**	**Conventional**	**YASA**
Average	169.42	90.38	143.87	177.20	173.31	117.13
Median	85.0	59.50	76.50	85.0	85.0	63.50

The observed performance data (time to finish a task; mouse movement) can be thought of as a random sample of a larger population. Our observation is thus a reflection of the unobservable underlying probability density function, according to which a large population is distributed. Using statistics we are able to estimate the density function based on the observed data, so that we can plot the probability of obtaining a certain performance value. Next, we provide the estimated density plot for the recorded time performance as well as mouse movement performance.

Fig. 8-14: Density plot of the execution time.
The red curve represents conventional tools and the blue curve represents YASA.

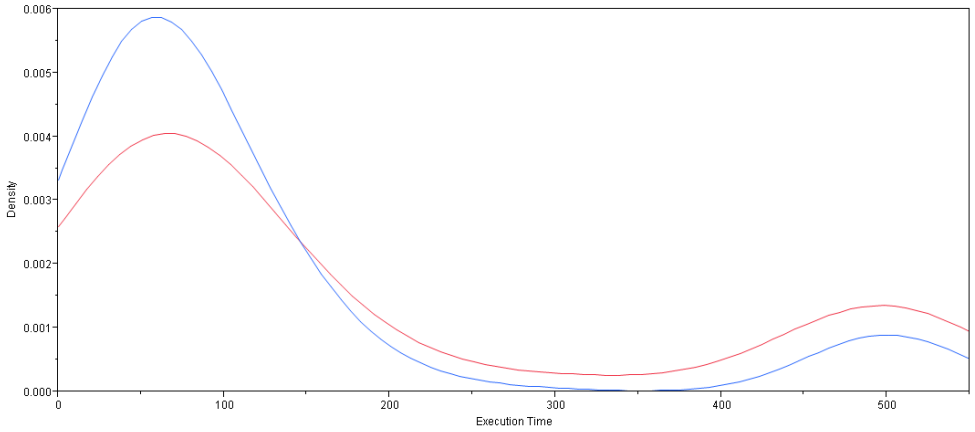
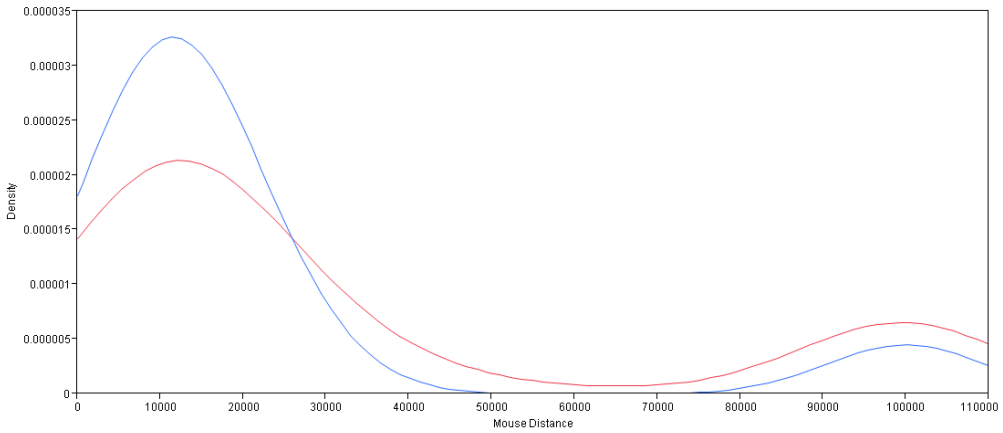


Fig. 8-15: Density plot of the mouse movement.
The red curve represents conventional tools and the blue curve represents YASA.



The estimated density plot displayed in Fig. 8-14 illustrates the performance advantage of YASA over conventional tools. There are considerably more tasks which

are answered faster with YASA than with conventional tools. At about 500 seconds, there is a small peak due to the wrong answers which pile-up. The estimated density plot of the mouse movement (Fig. 8-15) shows a similar distribution, indicating a better performance of YASA compared to conventional tools.

We were also interested in the effects of various unknown variables like: “Does the used tool significantly influence the results?”, “Does the task have an influence on the performance?”, or “Is the performance of the applied tool dependent on the considered task?”.

In order to answer these questions, we set up a linear model with the factors “TaskID” (objective task identifier), “Tool” (YASA or conventional), “Person” and the interactions “TaskID*Tool”, “Person*Tool”, and “Person*TaskID”. The factor “Person” and its interactions are defined as random effects on the performance, i.e. we are not interested in the performance of individual persons. Rather, we are interested in the influence of the task, the influence of the tool, and in the influence of the task on a specific tool. In a mathematical formula, the model could be expressed as follows (random effects are not marked):

$$\begin{aligned}\text{Model} = {} & \alpha\text{Tool} + \beta\text{TaskID} + \gamma\text{Person} + \\ & \phi\text{Person} * \text{Tool} + \varphi\text{Person} * \text{TaskID} + \lambda\text{Tool} * \text{TaskID} + \varepsilon\end{aligned}$$

The parameters α, β, γ, ϕ, ψ, λ, and ε are determined by fitting the mixed model using the popular REML (restricted maximum likelihood) method [Harville 1977]. The calculated fit has a root mean squared error of 98.12 seconds. In light of the model’s simplicity and purpose we accept this size of noise.

The results of the fixed effect tests are summarized in Table 8-23. The task, the tool, as well as their combination have a statistically significant effect on the performance even on the 99.9% confidence level.

Table 8-23: Fixed Effect Tests using the F-test. Statistical significant values are marked by *.

Source	Degree of freedom	Probability > F
Task ID	5	<.0001*
Tool	1	0.0014*
Task ID * Tool	5	<.0001*

Table 8-24 gives the detailed results for each task. Consider for instance the influence of the chosen tool used to answer the tasks. In case conventional tools are used, the average time to solve a task is 28.1 seconds slower as the mean. Hence, if YASA is used, the average time to solve a task is 28.1 seconds faster. Summing it up, the difference between YASA and conventional tools is on average 56.2 seconds.

Table 8-24: Parameter estimates. The estimates tell how much slower (positive values) or faster (negative values) the performance is on average depending on the task, tool, or task*tool combination. Significance is determined using the t-test. Statistical significant effects are marked by *.

Term	Estimate	Std Error	Probability > |t|
Task ID[1]	-104.17	21.21	< .0001*
Task ID[2]	-1.82	21.21	0.9319

Task ID[3]	-59.7	21.21	0.0059*
Task ID[4]	99.29	21.21	< .0001*
Task ID[5]	137.41	21.21	< .0001*
Task ID[6]	-71.01	21.21	0.0012*
Tool[Conventional]	28.1	7.5	0.0014*
Tool[YASA]	-28.1	7.5	0.0014*
Task ID[1]*Tool[Conventional]	-29.6	14.16	0.0393*
Task ID[2]*Tool[Conventional]	-28.84	14.16	0.0445*
Task ID[3]*Tool[Conventional]	-37.47	14.16	0.0095*
Task ID[4]*Tool[Conventional]	10.61	14.16	0.4557
Task ID[5]*Tool[Conventional]	93.33	14.16	< .0001*
Task ID[6]*Tool[Conventional]	-8.03	14.16	0.572
Task ID[1]*Tool[YASA]	29.6	14.16	0.0393*
Task ID[2]*Tool[YASA]	28.84	14.16	0.0445*
Task ID[3]*Tool[YASA]	37.47	14.16	0.0095*
Task ID[4]*Tool[YASA]	-10.61	14.16	0.4557
Task ID[5]*Tool[YASA]	-93.33	14.16	< .0001*
Task ID[6]*Tool[YASA]	8.03	14.16	0.572

Task number 1 is a baseline task and the estimated model confirms this. In case YASA is used to answer the question, the average time to answer the question is 2*(-28.1+29.6)=3 seconds slower than conventional tools. This difference is insignificant so that we can say that YASA as well as conventional tools deliver the same performance for task number 1.

The strongest effect of the chosen tool is observed for the difficult task number 5: In case YASA is applied to solve this task, the average time to provide an answer is reduced by approximately 242.86 seconds (Fig. 8-16 and Fig. 8-17). This estimate is highly significant because the p-value is less than 0.0001.

Fig. 8-16: Prediction profile. The left picture shows the estimated average execution time of a task in case conventional tools are used. The right picture shows the average execution time for task number 5 and its dependence on the used tool.

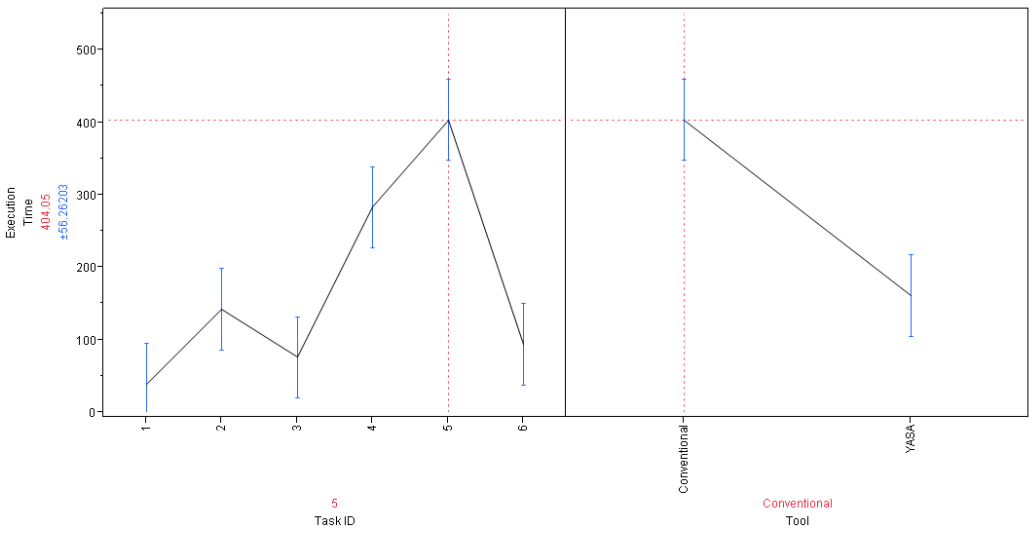
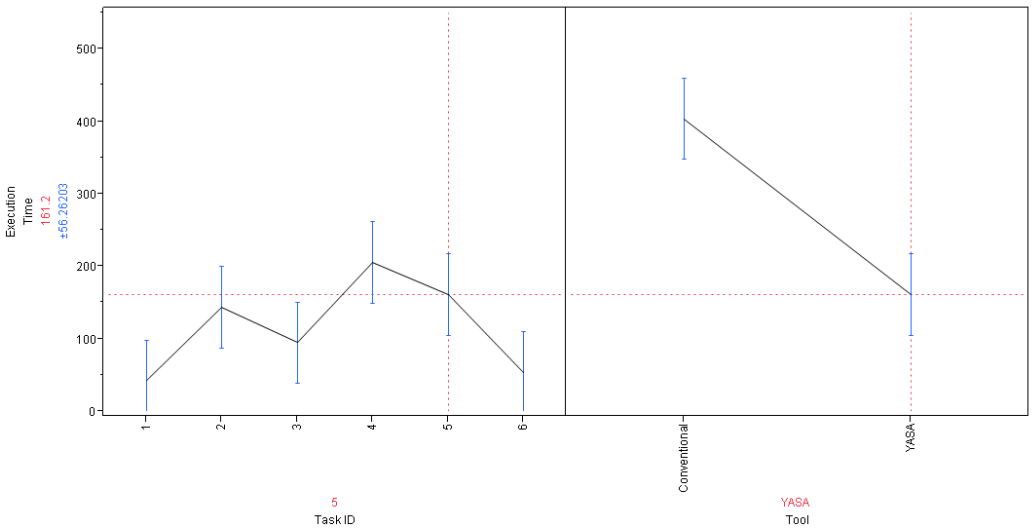


Fig. 8-17: Prediction profile. The left picture shows the estimated average execution time of a task in case YASA is used. The right picture shows the average execution time for task number 5 and its dependence on the used tool.



8.4.7 Results of the feedback phase

We first show the results of the questions which were quantified by IsoMetrics. Then, we show the results of the questions which were quantified by other metrics. The reason is that the IsoMetrics are numeric and can thus be processed differently than the other metrics using categories which can not be mapped to numeric values.

Fig. 8-18 summarizes the results of questions using the IsoMetrics. Notice that in case “No opinion” was selected by a user, the respective question is ignored for the following statistics.

Working on the objective tasks in the observation phase was on average not considered as tedious (QID 1; short for Question ID 1). Further, participants agree that the tasks correspond to task types that they typically do in their work (QID 2).

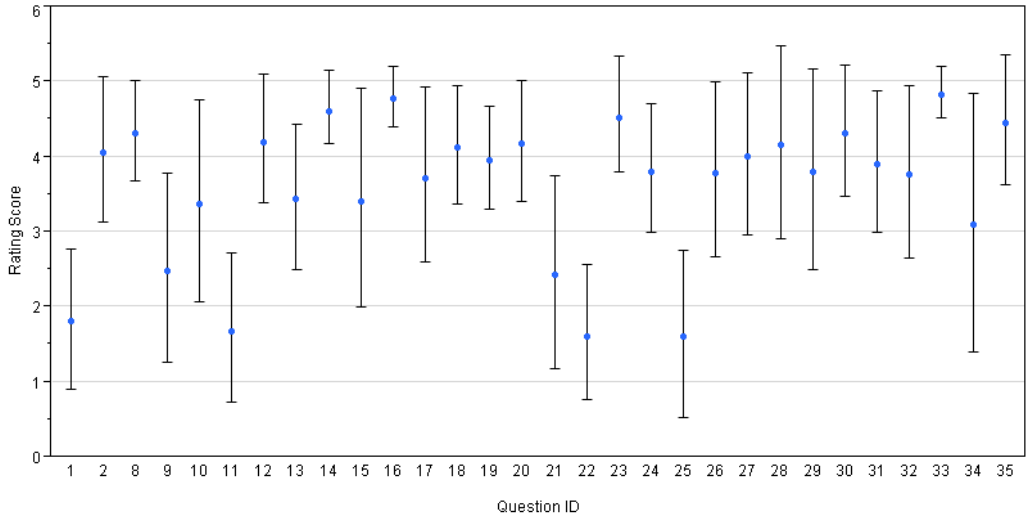
Regarding the quality of search results in YASA, the participants strongly agree that search results are relevant to their query (QID 8). QID 9 and QID 10 are coupled as the first one asks whether the user prefers search results from the department he is working in and the second one asks whether he often searches for information in other departments. The average answers for both questions suggest that people slightly prefer to search in other departments than in their own. However, the standard deviation is rather high suggesting a controversial point of view. The answers to questions 11 (notice: negated question), 12, and 13 show that people are satisfied with the speed, the user interface, and the search result quality of YASA.

Considering the usage of facets, the analysis shows that people strongly agree that facets help them to find information faster with YASA (QID 14) – this result is supported by the control QID 22. The importance of the individual facet is disputed. The “my files” facet (restricts results to documents the searcher has created) for instance is mediocre and has a high standard deviation (QID 15). The “file format”

facet (restricts results to a certain document format) is considered as highly relevant (QID 16). The “year” and “month” facets (restrict search results by date) have a similar average value and standard deviation as the “my files” facet – it is thus only slightly agreed that they are important (QID 17). The “project”, “department”, and “category” facets are all agreed to be important having a relatively low standard deviation (QID 18, 19, and 20). Whether more facets are needed is disputed by the participants (QID 21): the average rating shows a tendency towards disagreement – however, the standard deviation is high.

The integration of internal and external information in one search tool is strongly agreed to be helpful (QID 23). The integration of Tagged Documents into YASA is also considered relevant by most participants (QID 24). Interestingly, most test persons state that they do not use TagIt actively but rather consume existing tags passively (QID 25). The integration of PubMed, Applications, Wikipedia, DMOZ, Telephone Book, PRPZ-WebSite, and databases is on average agreed to be important (QID 26, 27, 28, 29, 30, 31, and 32). The integration of the PRPZ-Share is strongly considered to be important as indicated by a high average score and a very small standard deviation (QID 33). Interestingly, the integration of the U-Drive – the private document repository of a user – is highly disputed: the average score shows a slight preference to disagree and the standard deviation is very large (QID 34). The demand to be able to search other areas within the corporate environment beside Pharma Research is very strong (QID 35).

Fig. 8-18: Questionnaire results for questions using IsoMetrics (scale from 1 to 5). The blue points represent the average value and the black error lines represent one standard deviation.

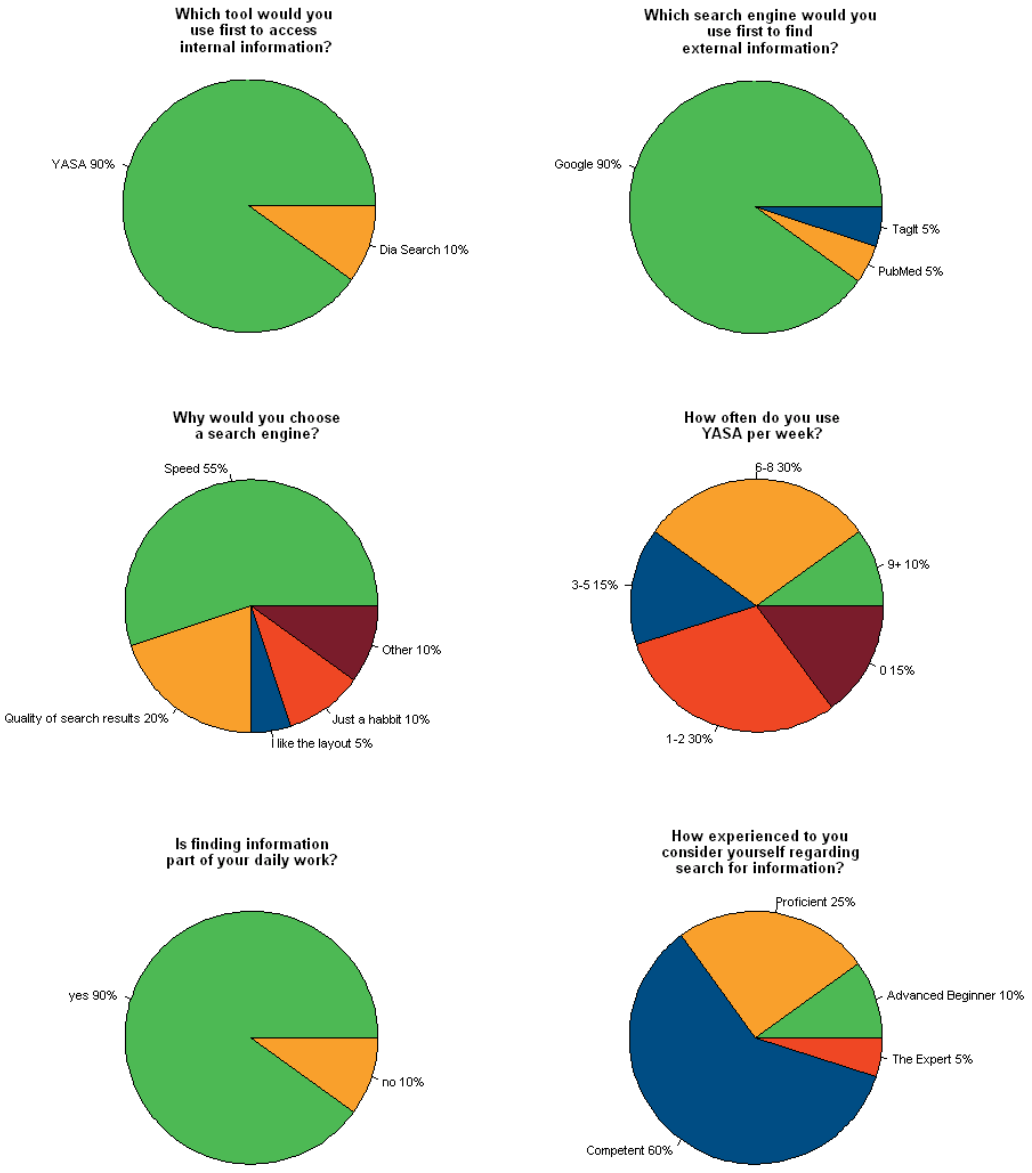


Next, we present the results of the questions which used a metric different to IsoMetrics. Fig. 8-19 displays a summary of the respective results. The majority of the participants (90%) agree that YASA is the first tool they would use to access internal information (QID 3) followed by the Diagnostics search engine in the second place (10%). For external information (QID 4) the majority (90%) would first use Google, followed by PubMed and TagIt on second place (5% each). Being asked why they would choose a search engine (QID 5), approximately half of the test persons

(55%) selected speed as the most important point. Quality of search results is considered most relevant by a fifth of the persons (20%). Regarding the usage of YASA per week (QID 7) the picture is twofold: Approximately 40% use YASA six or more times per week, while approximately 45% use it up to five times per week – 15% stated not to use YASA.

90% of the participants stated that finding information is part of their daily work (QID 38). Regarding their search experience, 60% feel they are competent, 25% proficient, 10% advanced beginner, and 5% to be the expert.

Fig. 8-19: Results for questions 3, 4, 5, 7, 38, and 39.



We finish this section with a discussion of the comments which have been given at the end of the questionnaire. Three participants were quite enthusiastic about the possibilities YASA offers and they wrote the following comments:

*"Thank you very much for the great work –
without YASA I have had lost a lot of time of my life!"*

*"Now that I have seen how easy YASA is to use and how user friendly it is;
I found it to be much easier to find internal information."*

"Learned about YASA and find out advantages for my daily life"

Critics have been mentioned especially concerning the tagged documents section (which integrates results from TagIt). People complained about dead links, slow result fetching, and the quality of the tags. The first issue could be handled by automatically checking whether the respective link is not reachable over a certain period of time. The second issue could also be solved by optimizing the respective service. The last issue is due to the fact that tags are mainly provided by people from the diagnostics division rather than workers from pharmaceutical research.

People also complained about a missing auto correction feature for queries. Another source of complain was the integration of the telephone book. In particular, people are missing navigation support in order to easily browse the telephone results by department.

8.4.8 Results interpretation

The user study shows that on average, YASA outperforms conventional tools for the given objective tasks – which according to the participants represent well the task types they need to do for their daily work.

Simple queries such as the lookup query of a Wikipedia article or a specific publication are answered similarly fast with YASA as with conventional tools. Indeed, the integration of sources such as Wikipedia, PubMed, or the company's telephone book does not necessarily imply that search requests can be answered faster compared to the direct access of the tools. Rather, the benefit is the ability to access all sources from one user interface. A second advantage is that people must know only the address of YASA and not of all specialized tools – this is especially useful once more database sources are integrated.

We could also observe that people sometimes needed more time to solve the tasks with YASA because they expected to receive a merged result list. It thus took a while until they understood that the appropriate source, such as the telephone book, needs to be selected prior to receiving the respective results. This suggests on the one hand, that the current layout might not be optimal. On the other hand, it suggests that a merged result list would be better suited, because it frees the user of manually selecting the appropriate source. However, the difficulty is to merge the different result rankings properly and to combine distinct types of results in one user interface (database results for instance are rendered differently than results from text documents).

In contrast to simple queries, YASA outperforms conventional tools significantly in case difficult queries need to be answered such as task number 5. Here, searchers clearly benefit from the automatic categorization of documents into facets like departments, projects, and categories. The questionnaire also outlines that the facets offered by YASA are helpful to the users and that they cover the most important aspects for the daily work. Nonetheless, we assume that a refinement of the document category facet beyond Agenda, Minutes, Memo, Portfolio, etc. could be beneficial a) for extending the adaptation of search results and b) for persons who are not well aware of the present information landscape.

Our current configuration of "role-based adaptation of search results" seems to be suboptimal according to the average ratings of QID 9 and QID 10. These indicate that some people prefer results from their department while others do not. In order to tackle this problem, we could expose the user model to the searcher, so that he can adjust the preferences as he wishes. Or alternatively, we could automatically learn a user's profile from past searches. However, this requires personal log-data to be stored (a privacy issue in corporations; cf. [Kobsa 2007]) and it requires a minimum search activity in order to be applicable.

We were a bit surprised that the choice of tool has no significant influence on task 4 ("Find […] literature […] about YASA") and interpret the result as follows. We could observe that the test persons often provided the intranet URL of YASA and the web help page of YASA as an answer. This indicates that they were not fully aware about the question's intention: Namely, to provide a folder on the PRPZ-Share or an Internet URL, where documents about YASA are bundled. We further assume this is due to the fact, that the respective PRPZ-Share folder is not categorized as a project. Indeed, we categorize only medicine projects from Pharma Research and not any IT-related projects. The incorporation of non-medicine projects is thus the target of future work.

8.5 Discussion

In this chapter we evaluated the prototype YASA using online and offline evaluations. The data used to conduct the evaluation originated from query logs, monitoring logs, and from questionnaires. Logs were used to determine the usage characteristics of YASA as well as of other tools, and also to verify YASA's ranking performance. The monitoring logs obtained from the user study were used to evaluate the performance of YASA compared to conventional tools. Finally, the questionnaire allowed us to capture the user opinion about general search aspects and about the principles applied in YASA.

One of the first results we obtained was that YASA is one of the most frequently used tools in Pharmaceutical Research in Penzberg, to access internal information. Further, it was shown that YASA's usage is still growing.

An analysis of YASA's query logs showed that the vast majority of queries (89%) target the PRPZ-Share. Similarly, the results of the questionnaire also point out that the PRPZ-Share is the most important source. This suggests that enough incentives

(e.g. quality of search results, faceted navigation, etc.) are given in order to access the PRPZ-Share by means of YASA.

Regarding the other sources however, the results of the query log analysis and the questionnaire do not agree well. While the query logs indicate that these are barely used, the answers to the questionnaire indicate that their integration is useful.

In case of PubMed we can identify two main reasons: user habit and speed of information retrieval (cf. Fig. 8-19; QID 5). Indeed, people might just be used to access the source directly or via Google. Regarding speed, people might get results faster if they use Google because the results often point directly to the full-text content of the article. Conducting the same search in YASA, people first transmit the query, then select PubMed as the source, and then they select a result item, from where another two clicks are necessary in order to receive the full text (these two clicks are part of the official order procedure for publications and can not be circumvented). Arguably, the workflow could be faster. Further, incentives such as faceted navigation are missing.

Regarding the low usage of databases we already argued in Chapter 8.1.2 and Chapter 8.1.7 that the access restriction and the prototypical implementation are reasons for the relatively low usage.

The investigation of the query logs revealed that queries and query sessions are quite simple: only few terms are used in a query, Boolean operators are de facto not used, and facets are used in about 10% of the searches. We can thus conclude, that experts barely require Boolean operators to find the information they seek. Rather, they use simple keyword queries in combination with faceted navigation in order to slice and dice the data.

Faceted navigation is enabled by the automatic categorization of text documents. In this thesis we compared two classification approaches: KE (knowledge engineering) and ML (machine learning). Even though both approaches delivered good results, on average ML clearly outperformed KE in terms of precision and recall. In particular, we recommend applying KE only if the documents have a simple and clear structure which can easily be described by means of rules. The usage of KE enables domain experts to explicitly encode their domain knowledge and the users benefit from the possibility to receive an automatic explanation of why a certain document was classified into the respective category.

The analysis of the ranking algorithms revealed that neither log-based adaptation nor rule-based adaptation delivered any significant improvement over the baseline.

In case of the log-based approach we were not able to detect a performance difference because of the small click per document ratio of 0.013. It is maybe due to the nature of research (which mostly requires the latest knowledge as was also demonstrated by the fact that 86% of the logged queries were unique) that the click ratio did not rise significantly in the investigated departments. We can thus conclude

that the simple log-based feedback approach we investigated in this thesis is not suited for the investigated context as well as for similar environments.

Rule-based adaptation on the other hand seemed to have failed due to wrong assumptions. The questionnaire results pointed out that people do not have a clear opinion whether they prefer results from their own context or from others (cf. Fig. 8-18; QID 9 and QID 10). They support thus the results from our ranking comparison in which no significant difference to the baseline algorithm could be detected. In effect, we need to revise our approach of a rule-based adaptation of search results. We see two directions which we could follow: a) let the user customize his preferences or b) cluster the users by departments and projects and let the adaptive system automatically adjust preferences by means of feedback (cf. Chapter 6.4.4). In the latter case, we have to determine whether such an approach would yield enough feedback data in the investigated environment.

Part IV

Conclusion

Chapter 9

Summary and future work

"The more we learn about the world, and the deeper our learning, the more conscious, clear, and well-defined will be our knowledge of what we do not know, our knowledge of our ignorance. The main source of our ignorance lies in the fact that our knowledge can only be finite, while our ignorance must necessarily be infinite."

Karl Popper (1902 – 1994)

9.1	Summary	181
9.2	Lessons learned	183
9.3	Future work	184
9.4	Outlook	186

Within a few years search engines have become the method of choice for accessing and retrieving information. This phenomenon is not only observable in the Web but also in large corporate intranet environments. Even though search engines have existed now for several years, recent studies show that search for information, especially in enterprise environments, is still a challenge: 1) information is clustered over multiple sources, 2) search engines apply a "one-size-fits-all" approach ignoring the needs of the individual, 3) users are given no guidance through the information space, 4) general search engines are ignorant to a company's existing knowledge. These are exactly the issues which we have tried to address in this thesis using techniques from information retrieval, adaptive systems, the semantic web, and machine learning. Carefully selected methods which are based on the given technologies have been implemented in the prototype YASA and their performance was measured in the research department of Roche in Penzberg.

9.1 Summary

In the *preliminary part*, we have presented the background of this thesis: the challenges in professional search, the foundation of the broad field of information retrieval, the role of adaptive systems in prevailing information overload, and the promises and current state of semantic technologies.

The chapter about information retrieval started with the historical evolution of the field. Then, the information retrieval (IR) process was introduced focusing on the widely used vector space model. We also discussed text processing as this step is not only important for IR but also for text categorization. Having introduced the basics of

IR as well as Web search, we discussed next the current state of the art in intranet search. At the end of the chapter we focused on evidence for deducing a document's relevancy.

The following chapter introduced the basics of adaptive systems. We focused predominantly on user modeling and listed applications of adaptive systems in the area of information retrieval (i.e. personalized search).

The third and last background chapter surveyed the state of the art in semantic technologies. Special focus was put on the combination of semantic technologies and information retrieval as well as the application of semantic technologies in the health care and life sciences sector.

In the *core part*, we started by giving an analysis of the current situation of search for information in the investigated department at Roche in Penzberg. The review was based on a log-file analysis and two empirical studies. The results point out that the present information landscape is complex and that scientists are not satisfied with the performance of existing search tools. Given this preliminary analysis we then suggested several quality characteristics a professional search tool should follow: One single entry point, role-specific ranking of search results, guided navigation, and exploitation of a priori knowledge.

Given these quality characteristics we described a concept of an ontology-based information retrieval approach which was implemented in the prototype YASA (Your Adaptive Search Agent). YASA enabled us to conduct several empirical studies in order to conclude which of the characteristics are most important for a professional search tool.

Ontologies are an integral part of our concept: They are used to capture a document's annotations such as entities occurring in the text, the document's overall topic, as well as its organizational embedding; and ontologies are also used to describe the user model used during the adaptation process.

The automatic annotation of documents is a key task because the additional metadata is crucial for the adaptation of search results as well as for guided navigation. Therefore, we investigated two paradigms of augmenting documents with metadata: knowledge engineering and machine learning. Despite the fact, that machine learning is nowadays the most used method for text categorization, we had a special interest in comparing it to the knowledge engineering approach because the latter enabled us to explicitly encode domain knowledge and because it seamlessly integrates with the applied semantic web framework.

General search tools mostly use a "one-size-fits-all" approach which does not deliver optimal results in a professional environment. Therefore, we apply and compare two methods in YASA for achieving a role-specific ranking of search results: Knowledge-based adaptation and log-based adaptation. In the first case, a priori knowledge about the searcher's context is exploited. The background knowledge is encoded by

rules which complement the ontologies. The rules express what interests to infer and how to weight them. In addition to the rule-based adaptation approach we also apply a log-based approach which is based on past searches. The charm of the latter is its ability to automatically adjust document relevance as new click events are observed. Finally, we pointed out how adaptation of search results could be extended in case the majority of the indexed documents are assigned to a topic.

We have investigated the proposed quality characteristics by deploying YASA in the research environment of a large pharmaceutical company, namely Roche in Penzberg, Germany. The overall results show that YASA – and the integrated principles as a whole – is a success. Even though the usage is already considerable compared to other search engines we can still observe an increase.

The evaluation of the ranking algorithms revealed that a significant improvement over the baseline ranking could neither be achieved by the usage of a log-based feedback method nor by knowledge-based adaptation. A discussion of the reasons why both approaches have failed was given in the respective chapters.

Comparing the two approaches, knowledge engineering and machine learning, for text categorization showed that machine learning is on average better than knowledge engineering – especially if the recall levels are considered. However, the latter approach still performed quite well, so that a final judgment can only be given in light of the context. In case the considered document type has a clear structure and if it is important to explicitly encode the classification rules, then knowledge engineering is the way to go. Otherwise, we suggest using machine learning.

Regarding the exploitation of existing in-house knowledge we could demonstrate that this principle significantly improves a search engine's quality. In one case, namely the role-based adaptation, the application of in-house knowledge failed – maybe because the applied model was too vague. However, in case of text categorization, i.e. the classification of documents into projects and departments, we obtained a different result. The log-usage analysis as well as the user study shows that these aspects of faceted navigation are highly important for the users.

9.2 Lessons learned

Research in the area of adaptation has so far mostly targeted adaptation services for specific types of contents such as news and e-commerce (e.g. online book selling shops). Further, research regarding the personalization of search results mainly focused on implicit feedback methods which exploit click-through log data.

Context-based search (i.e. adaptation of search results based on the working context of a user) as investigated in this thesis, has not yet been well studied in research. The reason is that such investigations require quite complex systems. YASA is such a system. Its major scientific benefit thus is that it made possible research in context-based search in the first place. This thesis is a first step for research in this direction. So what are the more specific findings of this new research enabled by YASA?

At the beginning of this research project we started empty-handed: No formal ontology modeling the knowledge domain of Pharmaceutical Research was available. In fact, there was not even a taxonomy-like structure which described frequently used concepts and their relationships. For that reason, we had to build upon unstructured free-text documents. This is a serious issue, because the ability to extend search by symbolic knowledge representation approaches such as those of the semantic web (RDF/S, OWL, F-Logic, etc.) is limited by the quality of the indexed content. This was the case in this research project.

Interestingly, our results show that this wide-spread limitation can be compensated to a large degree by using automatic metadata extraction techniques like those compiled in Chapter 6.3 of this thesis.

Indeed, we were able to extract large quantities of metadata from free-text documents. Amongst others, we can extract the department to which a document belongs, the project to which it is related, and a range of topics. We are aware that such an automatic extraction does not yield optimal quality: extracted metadata may be erroneous and more importantly, the metadata is not semantically rich.

Indeed, the ontologies describing the metadata are not complex but have large similarities to taxonomies. Despite the “sparse semantics” offered by the ontologies (cf. 6.2), we applied faceted navigation and context-based personalization (cf. 6.4) on top of them. The relevance of semantics for professional search is reflected well by the fact that the integrated facets have a simple semantic structure: Even though simple, it is a success (cf. 8.5). Hence, YASA shows that context-based search benefits greatly from semantics.

We can thus conclude, that already a small bit of semantics goes a much longer way than one would have expected. Further, it seems to be worthwhile to intensify research on including even more semantics.

9.3 Future work

Our study opens many possibilities for future work. In particular, several issues which could not be targeted or were not comprehensively considered in this work can be further investigated. In just the same manner, new or improved approaches can then be analyzed and evaluated. Next, we discuss several areas of future work.

In the previous section we argued that even though we have not implemented a lot of semantics into YASA, it makes already a significant difference. A key question is therefore how to integrate more semantics?

The most apparent solution is to add further metadata by means of automatic classification. However, automatically extracting metadata from unstructured documents is related with errors. We are thus not able to produce 100% accurate metadata by means of automatic approaches. We could simply tolerate these errors as long as the error rate is low or we could enable a human feedback cycle so that any detected errors are eliminated.

Alternatively, the applied semantics could be expanded by extending the knowledge base, i.e. the ontologies. For instance, we could add more relationships to the concepts. These relationships could connect distinct concepts and they could extend search and navigation options at query time.

Another area of future work concerns the integration of internal and external information into one search tool which resulted in an inconclusive picture at the end of this thesis. While the query logs show a low access to various sources, the questionnaire suggests something different (Chapter 8). In case of internet sources the most probable reasons are a) that people access the respective sources slightly faster using Google and b) the searchers' habit of how information is accessed. In case of intranet sources we see the biggest challenges in integrating the databases – which in the current state of the implementation, do not offer a significant value for searchers. We therefore suggest focusing on in-house databases for future work. In particular, there are several databases which co-exist and which are usually only accessible by means of individual applications. In case all these databases would be integrated, connected and if an appropriate visualization of the data would be offered then a huge impact could be achieved in this sector.

Wolfram Alpha[24] and Google squared[25] are examples of such a "database of everything", in which data from distinct sources are connected and integrated in a novel way. For instance, a search for a city within Wolfram Alpha typically causes the tool to gather data about the population from databases, calculate statistics and to render the results in a convenient way. Achieving this "database of everything" implies a lot of work. Not only must all relevant knowledge about the data and their relationships be captured but also definitions of how to render the data, how to rank the data, etc. While a general methodology for achieving this goal might be found, the concrete implementation would be different for each domain. Further, issues such as data curation (how to correct errors in databases?) and citation (how to cite data) must also be addressed in this context.

Related to the integration of information sources into YASA is the question of how well YASA scales if it is deployed to other areas. Indeed, users would benefit significantly if they were able to search other areas as well. In fact, there is already ongoing work in order to deploy YASA also in the division of Pharmaceutical Technical Development – a division closely related to Pharmaceutical Research in Penzberg. Once this is accomplished and in case it is a success, it would be appealing to extend YASA to a global scale, i.e. to integrate data from other sites as well. In this respect it would also be interesting to verify how role-based adaptation performs on a global scale rather than on a local scale.

Another area of future work is the extension of text categorization to further topics. On the one hand, we would be able to improve log-based feedback by associating

[24] http://www.wolframalpha.com
[25] http://www.google.com/squared

the feedback data with a document's topic instead of the individual document – the click-through data would thus be generalized to the documents' topics. On the other hand, users would benefit from a richer set of guided navigation topics. In this context it would also be interesting to investigate if YASA's users can be involved into the categorization process. In particular, we could offer a voting button by which users could correct misclassifications or even suggest new topics. The collected feedback could be added as new training samples to the machine learning algorithm (either automatically or after a manual review-process) so that the model is adjusted accordingly.

The current knowledge-based adaptation approach did not deliver the expected results, i.e. it did not outperform the baseline ranking. Prior to applying major changes to the rules, we suggest to first leave them as is and to only conduct an automatic fine-tuning of the parameters (the weights which control a rule's influence on the ranking of search results). The fine-tuning could be done by optimizing on past query logs. In case this procedure fails one could still leave to the user the choice of setting the search context. Alternatively, a group-based adaptation approach could be applied in which the log data is associated with the searcher's role. However, doing so requires the storage of person-related logs. This must first be approved by the company's work council.

Even though YASA was fine-tuned for Roche, we believe that the applied principles as well as the main findings are transferable to other domains. We can not prove this statement without conducting an explicit evaluation within a different context. However, the statement is based on the following assumptions: other large corporations have a similar complex intranet environment; effective search of in-house information is also an issue in other companies; in-house knowledge such as administrative databases, project lists, and controlled vocabularies is also available in other companies. Therefore, we would only need to adjust the company specific parts, i.e. the topics of the classification ontology, the classification models, and the rules. Regarding feedback-based adaptation we expect the results to be similar for environments in which a similar user-to-document-ratio exists as in our case. However, if this ratio shifts towards more users vs. fewer documents, then we expect the feedback-based adaptation to outperform the baseline. Transferring the performance results of the role-based adaptation to other companies is not possible because other companies will have a different set of adaptation rules. However, we expect guided navigation as well as the incorporation of in-house knowledge into the classification processes to play a role of similar relevance in other companies.

9.4 Outlook

A final outlook and personal opinion might be appropriate at the end of this thesis.

Today's enterprise environments are characterized by an information space in which large quantities of data are stored in form of unstructured text. I don't see this to change fundamentally in the future because written text is the way to preserve knowledge.

Considering this statement in light of the steadily growing information quantities, the need for semantic search tools and personalization services becomes eminent. These approaches have the ability to improve a machine's understanding of the user's information need (which is such a highly fuzzy notion) by incorporating semantics and context. Indeed, the relevance of a document not only depends on the query and the document corpus but also on the context. The rising question is thus, how to permanently enrich unstructured text as well as the search process with semantics so that context-based search is made possible? Notice, that this question is also relevant for other content types such as images which are e.g. often used to sketch patents or to render small molecules. Being able to search these effectively is thus similar important.

This work made use of statistics, logical inference, and heuristics in order to classify unstructured text. However, these methods share a common drawback. They do not automatically adjust to changing environments as encountered in companies: projects change, departmental hierarchies are undergoing steady adjustments due to internal re-structuring, existing document topics shift their focus, new topics are added due to new trends in research, etc. In effect, the characteristics of each category undergo steady changes so that the classification models must be adjusted. The required adjustments may vary greatly depending on the considered class. In case of departments for instance, it would suffice to simply update the organizational hierarchy in the ontology. In case of topic classifications on the other hand, new training data must be provided potentially causing a considerable effort.

Without a doubt, this poses a serious challenge if context-based search is to be applied on the corpus of large corporations. This is particularly true for resources drawn from the Internet. It remains to be seen if other approaches, possibly based on human computation [Von Ahn & Dabbish 2004] in which the computational ability of humans solve problems computers cannot, could mitigate the issue. For the time being we have enough to do by coping with those problems for which approaches exist, that at least look promising. Hopefully, this thesis is a useful contribution towards solving some of these issues.

Bibliography

Agresti, A., 2002, *Categorical Data Analysis,* Second Edition ed., John Wiley & Sons, Hoboken, NJ, USA.

Allemang, D. & Hendler, J., 2008, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL,* Morgan Kaufmann, Burlington, MA, USA.

Anderson, C., 2006, *The long tail: Why the future of business is selling less of more,* Hyperion, New York, NY, USA.

Andrews, W., *Gartner Magic Quadrant for Information Access,* Gartner Inc. Retrieved 7 October, 9 A.D., from http://mediaproducts.gartner.com/reprints/microsoft/vol7/article2/article2.html.

Antoniou, G. & Van Harmelen, F., 2008, *A Semantic Web Primer,* 2nd Edition ed., The MIT Press, Cambridge, MA, USA.

Asnicar, F. A. & Tasso, C., 1997, 'ifWeb: A prototype of user modelbased intelligent agent for document filtering and navigation in the world wide web', in *Proceedings of the Workshop on Adaptive Systems and User Modeling on the World Wide Web (UM97), Sardinia, Italy*, pp. 3-12.

Avery, C. & Zeckhauser, R., 1997, 'Recommender systems for evaluating computer messages', *Communications of the ACM*, 40, (3) pp. 88-89.

Baeza-Yates, R. & Ribeiro-Neto, B., 1999, *Modern information retrieval,* Addison-Wesley, Harlow, England.

Bailey, J., Zhang, C., Budgen, D., Charters, S., & Turner, M., 2007, 'Search Engine Overlaps: Do they agree or disagree?', in *Proceedings of the Second International Workshop on Realising Evidence-Based Software Engineering, Minneapolis, MN, USA*, IEEE Computer Society, Washington, DC, USA, p. 2.

Basili, V., Caldiera, G., & Rombach, H. D., 1994, 'The Goal Question Metric Paradigm, Encyclopedia of Software Engineering', in *Encyclopedia of Software Engineering*, vol. Volume 1 J. J. Marciniak, ed., John Wiley & Sons, New York, NY, USA, pp. 528-532.

Becket, D., *RDF Test Cases,* W3C. Retrieved 10 March, 2009a, from http://www.w3.org/TR/rdf-testcases/#ntriples.

Becket, D., *RDF/XML Syntax Specification,* W3C. Retrieved 10 March, 2009b, from http://www.w3.org/TR/rdf-syntax-grammar/.

Becket, D., *Turtle - Terse RDF Triple Language,* Retrieved 10 March, 2009, from http://www.dajobe.org/2004/01/turtle/.

Berners-Lee, T., *A readable language for data on the Web,* W3C. Retrieved 10 March, 2009, from http://www.w3.org/DesignIssues/Notation3.

Berners-Lee, T. & Fischetti, M., 1999, *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor,* HarperCollins, New York, NY, USA.

Berners-Lee, T., Hendler, J., & Lassila, O., 2001, 'The semantic web', *Scientific American*, 284, (5) p. 35.

Bharat, K., Kamba, T., & Albers, M., 1998, 'Personalized, interactive news on the web', *Multimedia Systems*, 6, (5) pp. 349-358.

Boolean algebra, *Encyclopædia Britannica,* Encyclopædia Britannica Online. Retrieved 23 October, 2008, from http://www.britannica.com/EBchecked/topic/73621/Boolean-algebra.

Brank, J. & Grobelnik, M., 2002, 'Interaction of feature selection methods and linear classification models', in *Proceedings of the ICML-02 Workshop on Text Learning*, *Sydney, Australia*, Forthcoming.

Breese, J. H., Heckerman, D., & Kadie, C., 1998, 'Empirical Analysis of Predictive Algorithms for Collaborative Filtering', in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, *Madison, WI*, Morgen Kaufmann, San Francisco, CA, USA.

Brickley, D. & Guha, R. V., *RDF Vocabulary Description Language 1.0: RDF Schema,* W3C. Retrieved 10 March, 2009, from http://www.w3.org/TR/rdf-schema.

Brin, S. & Page, L., 1998, 'The anatomy of a large-scale hypertextual Web search engine', in *Computer Networks and ISDN Systems*, Elsevier, Amsterdam, pp. 107-117.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., & Wiener, J., 2000, 'Graph structure in the Web', *Computer Networks*, 33, (1-6) pp. 309-320.

Budzik, J., Hammond, K.J., & Birnbaum, L., 2001, 'Information access in context', *Knowledge-Based Systems*, 14, (1-2) pp. 37-53.

Burke, R., 2002, 'Hybrid Recommender Systems: Survey and Experiments', *User Modeling and User-Adapted Interaction*, 12, (4) pp. 331-370.

Castells, P., Fernandez, M., & Vallet, D., 2007, 'An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval', *IEEE Transactions on Knowledge and Data Engineering*, 19, (2) pp. 261-272.

Cheung, K.H., Prud'hommeaux, E., Wang, Y., & Stephens, S., 2009, 'Semantic Web for Health Care and Life Sciences: a review of the state of the art', *Briefings in Bioinformatics*, 10, (2) pp. 111-113.

Codd, E.F., 1970, 'A relational model of data for large shared data banks', *Communications of the ACM*, 13, (6) pp. 377-387.

Cohen, W. W. & Singer, Y., 1999, 'A simple, fast, and effective rule learner', in *Proceedings of the sixteenth national conference on Artificial intelligence*, American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 335-342.

Cortes, C. & Vapnik, V., 1995, 'Support-Vector Networks', *Machine Learning*, 20, (3) pp. 273-297.

Craswell, N., Hawking, D., & Robertson, S., 2001, 'Effective site finding using link anchor information', in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, *New Orleans, LA, USA*, ACM, New York, NY, USA, pp. 250-257.

Craswell, N., Robertson, S., Zaragoza, H., & Taylor, M., 2005, 'Relevance weighting for query independent evidence', in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, *Salvador, Brazil*, ACM, New York, NY, USA, pp. 416-423.

Dasarathy, B.V., 1990, *Nearest neighbor (NN) norms: NN pattern classification techniques,* IEEE Computer Society Press, Los Alamitos, CA.

Domingos, P. & Pazzani, M., 1997, 'On the optimality of the simple Bayesian classifier under zero-one loss', *Machine Learning*, 29, (2) pp. 103-130.

Eirinaki, M. & Vazirgiannis, M., 2003, 'Web mining for web personalization', *ACM Transactions on Internet Technology*, 3, (1) pp. 1-27.

Eiron, N. & McCurley, K. S., 2003, 'Analysis of anchor text for web search', in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, *Toronto, Canada*, ACM, New York, NY, USA, pp. 459-460.

Fagin, R., Kumar, R., McCurley, K. S., Novak, J., Sivakumar, D., Tomlin, J. A., & Williamson, D. P., 2003, 'Searching the workplace web', in *Proceedings of the 12th international conference on World Wide Web*, *Budapest, Hungary*, ACM, New York, NY, USA, pp. 366-375.

Feldman, S. & Sherman, C., 2004, 'The High Cost of Not Finding Information', *KM World*, 13, (3).

Fox, C., 1989, 'A stop list for general text', *ACM SIGIR Forum*, 24, (1-2) pp. 19-21.

Fox, C., 1992, 'Lexical analysis and stoplists', in *Information retrieval: data structures and algorithms*, W. B. Frakes & R. Baeza-Yates, eds., Prentice-Hall, Inc., Upper Saddle River, NJ, USA, pp. 102-130.

Fox, E., Betrabet, S., Koushik, M., & Lee, W., 1992, 'Extended boolean models', in *Information retrieval: data structures and algorithms*, W. B. Frakes & R. Baeza-Yates, eds., Prentice-Hall, Inc., Upper Saddle River, NJ, USA, pp. 393-418.

Frakes, W. B., 1992, 'Stemming algorithms', in *Information retrieval: data structures and algorithms*, W. B. Frakes & R. Baeza-Yates, eds., Prentice-Hall, Inc., Upper Saddle River, NJ, USA, pp. 131-160.

Franz, T., Scherp, A., & Staab, S., *Does a Semantic Desktop Facilitate Your Daily Tasks?,* University of Koblenz-Landau. Retrieved 17 September, 2009, from *http://kola.opus.hbz-nrw.de/volltexte/2008/334/*.

Fujii, A., 2008, 'Modeling anchor text and classifying queries to enhance web document retrieval', in *Proceeding of the 17th international conference on World Wide Web*, *Beijing, China*, ACM, New York, NY, USA, pp. 337-346.

Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., & Lochbaum, K. E., 1988, 'Information retrieval using a singular value decomposition model of latent semantic structure', in *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, *Grenoble, France*, ACM, New York, NY, USA, pp. 465-480.

Gediga, G. & Hamborg, K. C., 1999, 'IsoMetrics: An usability inventory supporting summative and formative evaluation of software systems', in *Proceedings of HCI International 99 on Human-Computer Interaction: Ergonomics and User Interfaces*, *Munich, Germany*, Lawrence Erlbaum Associates Inc, Hillsdale, NJ, USA, pp. 1018-1022.

Gentili, G., Micarelli, A., & Sciarrone, F., 2003, 'Infoweb: An adaptive information filtering system for the cultural heritage domain', *Applied Artifical Intelligence*, 17, (8-9) pp. 715-744.

Gillies, J. & Cailliau, R., 2000, *How the Web Was Born: The Story of the World Wide Web,* Oxford University Press, New York, NY, USA.

Gosset, W.S., 1908, 'The probable error of a mean', *Biometrika*, 6, (1) pp. 1-25 Retrieved 18 September 9 A.D., from http://biomet.oxfordjournals.org/cgi/content/citation/6/1/1.

Gruber, T.R., 1993, 'A translation approach to portable ontology specifications', *Knowledge acquisition*, 5, (2) pp. 199-220.

Harville, D.A., 1977, 'Maximum likelihood approaches to variance component estimation and to related problems', *Journal of the American Statistical Association*, 72, (358) pp. 320-338.

Haveliwala, T.H., 2003, 'Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search', *IEEE Transactions on Knowledge and Data Engineering*, 15, (4) pp. 784-796.

Hawking, D., 2004, 'Challenges in enterprise search', in *Proceedings of the 15th Australasian database conference - Volume 27*, *Dunedin, New Zealand*, Australian Computer Society, Inc., Darlinghurst, Australia, pp. 15-24.

Hawking, D., Rowlands, T., & Adcock, M., 2006, 'Improving rankings in small-scale web search using click-implied descriptions', *Australian Journal of Intelligent Information Processing Systems.ADCS 2006 special issue*, 9, (2) pp. 17-24.

Hayes, P., *RDF Semantics,* W3C. Retrieved 10 March, 2009, from http://www.w3.org/TR/rdf-mt/.

Heckerman, D., 1999, 'A tutorial on learning with Bayesian networks', *Learning in Graphical Models*.

Henze, N., Dolog, P., & Nejdl, W., 2004, 'Reasoning and ontologies for personalized e-learning in the semantic web', *Educational Technology & Society*, 7, (4) pp. 82-97.

Henzinger, M.R., Motwani, R., & Silverstein, C., 2002, 'Challenges in web search engines', *ACM SIGIR Forum*, 36, (2) pp. 11-22.

Herlocker, J.L., Konstan, J.A., Terveen, L.G., & Riedl, J.T., 2004, 'Evaluating collaborative filtering recommender systems', *ACM Transactions on Information Systems*, 22, (1) pp. 5-53.

Heydon, A. & Najork, M., 1999, 'Mercator: A scalable, extensible Web crawler', *World Wide Web*, 2, (4) pp. 219-229.

Horrocks, I., 2005, 'Owl rules, ok?', in *W3C Workshop on Rule Languages for Interoperability*, *Washington, DC, USA*.

Jacso, P., 2008, 'Google Scholar revisited', *Online Information Review*, 32, (1) pp. 102-114.

Jameson, A., 2003, 'Adaptive interfaces and agents', in *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, J. A. Jacko & A. Sears, eds., Lawrence Erlbaum Associates Inc., Hillsdale, NJ, USA, pp. 305-330.

Joachims, T., 2002, 'Optimizing Search Engines using Clickthrough Data', in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, *Edmonton, Alberta, Canada*, ACM, New York, NY, USA, pp. 133-142.

Joachims, T., 2003, 'Evaluating retrieval performance using clickthrough data', in *Text Mining*, J. Franke, G. Nakhaeizadeh, & I. Renz, eds., Physica / Springer, pp. 79-96.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., Radlinski, F., & Gay, G., 2007, 'Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search', *ACM Transactions on Information Systems*, 25, (2).

Joachims, T., Nedellec, C., & Rouveirol, C., 1998, 'Text categorization with Support Vector Machines: Learning with many relevant features', in *Machine Learning: ECML-98*, *Chemnitz, Germany*, Springer Berlin / Heidelberg.

Jones, K. S., 1999, 'What is the role of NLP in text retrieval?', in *Natural Language Information Retrieval*, T. Strzalkowski, ed., Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 1-24.

Kifer, M., Lausen, G., & Wu, J., 1995, 'Logical foundations of object-oriented and frame-based languages', *Journal of the ACM*, 42, (4) pp. 741-843.

Kim, J. & Seoul, K., 2006, 'What is a recommender system', in *Proceedings of Recommenders06.com*, pp. 1-21.

Kleinberg, J., 1999, 'Authoritative Sources in a Hyperlinked Environment', *Journal of the ACM*, 46, (5) pp. 604-632.

Klyne, G. & Carroll, J. J., *Resource Description Framework (RDF): Concepts and Abstract Syntax,* W3C. Retrieved 10 March, 2009, from http://www.w3.org/TR/rdf-concepts/.

Kobsa, A., 2007, 'Privacy-Enhanced Web Personalization', in *The Adaptive Web: Methods and Strategies for Web Personalization*, vol. 4321 P. Brusilovsky, A. Kobsa, & W. Nejdl, eds., Springer Berlin / Heidelbeg, pp. 628-670.

Koivunen, M. R. & Miller, E., *W3C Semantic Web Activity,* W3C. Retrieved 17 September, 2009, from http://www.w3.org/2001/12/semweb-fin/w3csw.

Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., & Riedl, J., 1997, 'GroupLens: applying collaborative filtering to Usenet news', *Communications of the ACM*, 40, (3) pp. 77-87.

Langville, A.N. & Meyer, C.D., 2006, *Google's Pagerank and Beyond: The Science of Search Engine Rankings,* Princeton University Press, Princeton, New Jersey, USA.

Lee, J. H., 1995, 'Combining multiple evidence from different properties of weighting schemes', in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, *Seattle, Washington, USA*, ACM, New York, NY, USA, pp. 180-188.

Li, X. & Croft, W. B., 2003, 'Time-based language models', in *Proceedings of the twelfth international conference on Information and knowledge management*, *New Orleans, LA, USA*, ACM, New York, NY, USA, pp. 469-475.

Liu, F., Yu, C., & Meng, W., 2004, 'Personalized web search for improving retrieval effectiveness', *IEEE Transactions on Knowledge and Data Engineering*, 16, (1) pp. 28-40.

Manola, F. & Miller, F., *RDF Primer,* W3C. Retrieved 10 March, 2009, from *http://www.w3.org/TR/rdf-primer/*.

Maßun, M. 2008. *Collaborative Information Management in Enterprises.* University of Regensburg.

McGuinnes, D. L. & Van Harmelen, F., *OWL Web Ontology Language Overview,* W3C. Retrieved 10 March, 2009, from *http://www.w3.org/TR/owl-features*.

McNemar, Q., 1947, 'Note on the sampling error of the difference between correlated proportions or percentages', *Psychometrika*, 12, (2) pp. 153-157.

Micarelli, A., Gasparetti, F., Sciarrone, F., & Gauch, S., 2007, 'Personalized search on the World Wide Web', in *The Adaptive Web: Methods and Strategies for Web Personalization*, vol. 4321 P. Brusilovsky, A. Kobsa, & W. Nejdl, eds., Springer Berlin / Heidelbeg, pp. 195-230.

Micarelli, A. & Sciarrone, F., 2004, 'Anatomy and empirical evaluation of an adaptive web-based information filtering system', *User Modeling and User-Adapted Interaction*, 14, (2-3) pp. 159-200.

Middleton, C. & Baeza-Yates, R., *A comparison of open source search engines,* Universitat Pompeu Fabra Department of Technologies. Retrieved 1 January, 2009, from *http://wrg.upf.edu/WRG/dctos/Middleton-Baeza.pdf*.

Middleton, S.E., Shadbolt, N.R., & De Roure, D.C., 2004, 'Ontological user profiling in recommender systems', *ACM Transactions on Information Systems*, 22, (1) pp. 54-88.

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K.J., 1990, 'WordNet: An on-line lexical database', *International journal of lexicography*, 3, (4) pp. 235-244.

Miller, S., *Information-seeking Behaviour of Academic Scientists in the Electronic Age, a Literature Review,* Canadian Research Knowledge Network. Retrieved 30 March, 2009, from *http://www.cnslp.ca/initiatives/evaluation/LitReview-SusanMiller.pdf*.

Mitchell, T., 1997, *Machine Learning,* McGraw-Hill.

Mühlbacher, S. 2008. *Scientific Information Literacy in Enterprises.* University of Regensburg.

Nadeau, D. & Sekine, S., 2007, 'A survey of named entity recognition and classification', *Linguisticae Investigationes*, 30, (1) pp. 3-26.

Najork, M. A., Zaragoza, H., & Taylor, M. J., 2007, 'Hits on the web: how does it compare?', in *Proceedings of the 30th annual international ACM SIGIR conference on*

Research and development in information retrieval, *Amsterdam, Netherlands*, ACM, New York, NY, USA, pp. 471-478.

Noll, M. G. & Meinel, C., 2007, 'Web Search Personalization Via Social Bookmarking and Tagging', in *The Semantic Web*, *Busan, Korea*, Springer Berlin / Heidelberg, pp. 367-380.

Page, L., Brin, S., Motwani, R., & Winograd, T., *The pagerank citation ranking: Bringing order to the web,* Stanford InfoLab. Retrieved 17 September, 2009, from *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1768*.

Pitkow, J., Schntze, H., Cass, T., Cooley, R., Turnbull, D., Edmonds, A., Adar, E., Breuel, T., Methodology, T., & Directions, R.F., 2002, 'Personalized search', *Communications of the ACM*, 45, (9) pp. 50-55.

Porter, M., 1980, 'An Algorithm for Suffix Stripping Program', *Program*, 14, (3) pp. 130-137.

Pretschner, A. & Gauch, S., 1999, 'Ontology Based Personalized Search', in *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, *Chicago, IL, USA*, IEEE Computer Society, Washington, DC, USA, pp. 391-398.

Prud'hommeaux, E. & Seaborne, A., *SPARQL Query Language for RDF,* The World Wide Web Consortium. Retrieved 23 March, 2009, from *http://www.w3.org/TR/rdf-sparql-query/*.

Qiu, F. & Cho, J., 2006, 'Automatic identification of user interest for personalized search', in *Proceedings of the 15th international conference on World Wide Web*, *Edinburgh, Scotland*, ACM, New York, NY, USA, pp. 727-736.

Radlinski, F. & Joachims, T., 2005, 'Query chains: learning to rank from implicit feedback', in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, *Chicago, Illinois, USA*, ACM, New York, NY, USA, pp. 239-248.

Raghavan, V. V. & Sever, H., 1995, 'On the reuse of past optimal queries', in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, *Seattle, WA, USA*, ACM, New York, NY, USA, pp. 344-350.

Reeve, L. & Han, H., 2005, 'Survey of semantic annotation platforms', in *Proceedings of the 2005 ACM symposium on Applied computing*, *Santa Fe, New Mexico*, ACM, New York, NY, USA, pp. 1634-1638.

Rich, E., 1979, 'User modeling via stereotypes', *Cognitive Science*, 3, (4) pp. 329-354.

Robertson, S., 2004, 'Understanding inverse document frequency: on theoretical arguments for IDF', *Journal of Documentation*, 60, pp. 503-520.

Robertson, S.E. & Jones, S., 1976, 'Relevance Weighting of Search Terms', *Journal of the American Society for Information Science*, 27, (3) pp. 129-146.

Robertson, S. E. & Walker, S., 1994, 'Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval', in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, *Dublin, Ireland*, Springer-Verlag New York, Inc., New York, NY, USA, pp. 232-241.

Robertson, S.E., Walker, S., & Beaulieu, M., 2000, 'Experimentation as a way of life: Okapi at TREC', *Information Processing and Management*, 36, (1) pp. 95-108.

Rogati, M., 2002, 'High-performing feature selection for text classification', in *Proceedings of the eleventh international conference on Information and knowledge management*, *McLean, Virginia, USA*, ACM, New York, NY, USA, pp. 659-661.

Rose, D. E. & Levinson, D., 2004, 'Understanding user goals in web search', in *Proceedings of the 13th international conference on World Wide Web*, *New York, NY, USA*, ACM, New York, NY, USA, pp. 13-19.

Safavian, S.R. & Landgrebe, D., 1991, 'A survey of decision tree classifier methodology', *IEEE Transactions on Systems, Man, and Cybernetics*, 21, (3) pp. 660-673.

Salton, G., Fox, E.A., & Wu, H., 1983, 'Extended Boolean information retrieval', *Communications of the ACM*, 26, (11) pp. 1022-1036.

Salton, G., Wong, A., & Yang, C.S., 1975, 'A vector space model for automatic indexing', *Communications of the ACM*, 18, (11) pp. 613-620.

Schocken, S. & Finin, T., 1987, 'Prolog Meta-Interpreters for Rule-Based Inference Under Uncertainty', *Information Systems Working Papers Series*.

Sebastiani, F., 2002, 'Machine learning in automated text categorization', *ACM computing surveys*, 34, (1) pp. 1-47.

Selberg, E. & Etzioni, O., 1997, 'The MetaCrawler architecture for resource aggregation on the Web', *IEEE Expert*, 12, (1) pp. 11-14.

Sidney, S., 1957, 'Nonparametric statistics for the behavioral sciences', *The Journal of Nervous and Mental Disease*, 125, (3) p. 497.

Silverstein, C., Marais, H., Henzinger, M., & Moricz, M., 1999, 'Analysis of a very large web search engine query log', in *ACM SIGIR Forum*, ACM, New York, NY, USA, pp. 6-12.

Singhal, A., 2001, 'Modern information retrieval: A brief overview', *IEEE Data Engineering Bulletin*, 24, (4) pp. 35-43.

Singhal, A., Buckley, C., & Mitra, M., 1996, 'Mitra. M.(1996) Pivoted document length normalization', in *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, *Zurich, Switzerland*, ACM, New York, NY, USA, pp. 21-29.

Singhal, A., Choi, J., Hindle, D., Lewis, D., & Pereira, F., 1999, 'At&t at TREC-7', in *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, NIST Special Publication, pp. 239-252.

Smirnov, I., *Overview of Stemming Algorithms,* DePaul University. Retrieved 27 November, 2008, from *http://the-smirnovs.org/info/stemming.pdf*.

Sormunen, E. 2000. *A Method for Measuring Wide Range Performance of Boolean Queries in Full-text Databases.* University of Tampere.

Speretta, M. & Gauch, S., 2005, 'Personalized search based on user search histories', in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, *France*, IEEE Computer Society, Washington, DC, USA, pp. 622-628.

Spink, A., Jansen, B.J., Blakely, C., & Koshman, S., 2006, 'A study of results overlap and uniqueness among major web search engines', *Information Processing and Management*, 42, (5) pp. 1379-1391.

Steele, R., 2001, 'Techniques for specialized search engines', in *Proceedings of Internet Computing '01*, *Las Vegas, NV, USA*.

Stojanovic, N. 2005. *Ontology-based Information Retrieval: Methods and Tools for Cooperative Query Answering.*

Sugiyama, K., Hatano, K., & Yoshikawa, M., 2004, 'Adaptive web search based on user profile constructed without any effort from users', in *Proceedings of the 13th international conference on World Wide Web*, *New York, NY, USA*, ACM, New York, NY, USA, pp. 675-684.

Sun, J. T., Zeng, H. J., Liu, H., Lu, Y., & Chen, Z., 2005, 'CubeSVD: a novel approach to personalized Web search', in *Proceedings of the 14th international conference on World Wide Web*, *Chiba, Japan*, ACM, New York, NY, USA, pp. 382-390.

Surowiecki, J., 2004, *The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations,* Doubleday Books.

Tachau, J., *Analysis of Three Personalized Search Tools in Relation to Information Search: iGoogle, LeapTag, and Yahoo! MyWeb,* University of Oregon, Portland, Oregon, USA. Retrieved 25 June, 2009, from *https://scholarsbank.uoregon.edu/xmlui/handle/1794/7661?show=full*.

Thompson, C.A., Goker, M.H., & Langley, P., 2004, 'A personalized system for conversational recommendations', *Journal of Artificial Intelligence Research*, 21, pp. 393-428.

Turtle, H., 1994, 'Natural language vs. Boolean query evaluation: a comparison of retrieval performance', in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, *Dublin, Ireland*, Springer-Verlag New York, Inc., New York, NY, USA, pp. 212-220.

Turtle, H. & Croft, W. B., 1989, 'Inference networks for document retrieval', in *Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval*, *Brussels, Belgium*, ACM, New York, NY, USA, pp. 1-24.

Van Gelder, A., Ross, K.A., & Schlipf, J.S., 1991, 'The well-founded semantics for general logic programs', *Journal of the ACM*, 38, (3) pp. 619-649.

Van Rijsbergen, C.J., 1979, *Information retrieval,* Butterworth-Heinemann, Newton, MA, USA.

Vandervalk, B.P., McCarthy, E.L., & Wilkinson, M.D., 2009, 'Moby and Moby 2: Creatures of the Deep (Web)', *Briefings in Bioinformatics*, 10, (2) pp. 114-128http://bib.oxfordjournals.org/cgi/content/abstract/10/2/114.

Von Ahn, L. & Dabbish, L., 2004, 'Labeling images with a computer game', in *Proceedings of the SIGCHI conference on Human factors in computing systems*, *Vienna, Austria*, ACM, New York, NY, USA, pp. 319-326.

Weinberger, D., 2007, *Everything is miscellaneous: The power of the new digital disorder,* Henry Holt, New York, NY, USA.

Westerveld, T., Hiemstra, D., & Kraaij, W., 2002, 'Retrieving Web Pages Using Content, Links, URLs and Anchors', in *TREC 2001: tenth text retrieval conference*, *Gaithersburg, MD*, National Institute of Standards and Technology, Gaithersburg, MD, USA, pp. 663-672.

Wikipedia, *Amazon.com,* Retrieved 17 April, 2009, from *http://en.wikipedia.org/wiki/Amazon.com*.

Wilcoxon, F., 1945, 'Individual comparisons by ranking methods', *Biometrics Bulletin* pp. 80-83.

Wilkinson, R. & Hingston, P., 1991, 'Using the cosine measure in a neural network for document retrieval', in *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, *Chicago, IL, USA*, ACM, New York, NY, USA, pp. 202-210.

Wong, S. K. M., Ziarko, W., & Wong, P. C. N., 1985, 'Generalized vector spaces model in information retrieval', in *Proceedings of the 8th annual international ACM SIGIR*

conference on Research and development in information retrieval, *Montreal, Quebec, Canada*, ACM, New York, NY, USA, pp. 18-25.

Xue, G. R., Zeng, H. J., Chen, Z., Ma, W. Y., Zhang, H. J., & Lu, C. J., 2003, 'Implicit link analysis for small web search', in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, *Toronto, Canada*, ACM, New York, NY, USA, pp. 56-63.

Xue, G. R., Zeng, H. J., Chen, Z., Yu, Y., Ma, W. Y., Xi, W. S., & Fan, W. G., 2004, 'Optimizing web search using web click-through data', in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, *Washington, DC, USA*, ACM, New York, NY, USA, pp. 118-126.

Yee, K. P., Swearingen, K., Li, K., & Hearst, M., 2003, 'Faceted metadata for image search and browsing', in *Proceedings of the SIGCHI conference on Human factors in computing systems*, *Ft. Lauderdale, FL, USA*, ACM, New York, NY, USA, pp. 401-408.

Zamir, O., Korn, J.L., Fikes, A.B., & Lawrence, S.R., 2005, 'Personalization of played content ordering in search results', Patent US 2005/0240580 A1.

Zobel, J. & Moffat, A., 1998, 'Exploring the similarity space', *ACM SIGIR Forum*, 32, (1) pp. 18-34.

List of Figures

Fig. 2-1: Information retrieval process. 15
Fig. 2-2: Query processing 19
Fig. 2-3: A vector space matrix consisting of document vectors (vertical) and term vectors (horizontal). 20
Fig. 2-4: Illustration of two 2-dimensional vector spaces. 20
Fig. 2-5: Gartner's magic quadrant of information access 40
Fig. 2-6: Precision – Recall trade off graph 49
Fig. 3-1: Relationships between a user model and user profiles 52
Fig. 3-2: Types of personalization processes. The user profile can occur during the retrieval process (a), a-posteriori in a re-ranking step (b), or a priori in a query modification step (c). 55
Fig. 3-3: Principles of recommender systems. A black arrow represents the interest (e.g. rating or purchase) of a user for an item (A, B or C). Features of users and items are denoted by "<…>". A red arrow marks correlated users or items. The green arrow represents the predicted recommendation for the active user (bold). After [Kim & Seoul 2006]. 65
Fig. 4-1: Semantic Web layers 72
Fig. 4-2: Alternative Semantic Web layers 73
Fig. 4-3: An RDF graph describing Eric Miller [Manola & Miller 2004]. 74
Fig. 4-4: Sample RDF(S) graph. 76
Fig. 4-5: An exemplary class hierarchy with instances 79
Fig. 5-1: Typically, an employee having a specific information need is confronted with five main sources of data, each having countless data records. 90
Fig. 5-2: Distribution of file types on the PRPZ-WebSite. 91
Fig. 5-3: File type distribution on the PRPZ-Share. 92
Fig. 5-4: Usage of search engines linked from the PRPZ-WebSite. Data logged over a period of one month in 2007. 94
Fig. 6-1: Assigning metadata to "unstructured" text. 102
Fig. 6-2: Converting "unstructured" text into structured metadata. 103
Fig. 6-3: Role-based adaptation of the ranking of results. 103
Fig. 6-4: Excerpt of the classification ontology. Classes in gray are filled when reading the document from the source. The blue relationships are determined by means of named entity recognition. The green relationships are determined by reasoning or by machine learning 106
Fig. 6-5: Excerpt of the annotation ontology, displaying classes and their relationships. The given entity types are used as features to annotate whole documents. 107
Fig. 6-6: Excerpt of the adaptation ontology. 108

Fig. 6-7: Excerpt of the classification ontology. The grayed classes are detected by the classifier, while white classes are not considered in context of this thesis. 109
Fig. 6-8: Text processing pipeline..... 110
Fig. 6-9: KB classification pipeline. 113
Fig. 6-10: ML pipeline for learning a model. 117
Fig. 6-11: Classification pipeline..... 118
Fig. 6-12: Inferring a user's assumed interests. 120
Fig. 6-13: Rule "interestInDepartment" 121
Fig. 6-14: Rule "interestInArea" 121
Fig. 6-15: Rule "interestInProject" 122
Fig. 6-16: Incorporating click-through feedback into the document index. 123
Fig. 6-17: User profile example of a person. 125
Fig. 6-18: Each column illustrates the distribution of the similarity score of a different query. The blue graphs denote the baseline ranking, i.e. the default VSM and the red graphs denote the context-boosted ranking. The gray line marks the median position / score. 127
Fig. 7-1: Multi-tier architecture of YASA's search & retrieval part. 134
Fig. 7-2: The components of YASA. 135
Fig. 7-3: Screenshot of YASA's search interface. 138
Fig. 8-1: Usage of search engines linked from the PRPZ-WebSite. Data logged over a period of 6 month (Jan '09 – June '09). 143
Fig. 8-2: Number of queries transmitted to YASA per month. Data logged over a period of 18 month (Jan '08 – June '09). 144
Fig. 8-3: Average number of queries per hour. Data logged over a period of 18 month (Jan '09 – June '09). 144
Fig. 8-4: Query distribution on sources within YASA. Data logged over a period of 3 month (Apr '09 – June '09) 145
Fig. 8-5: Click distribution of sources within YASA. Data logged over a period of 6 month (Jan '09 – June '09). 146
Fig. 8-6: Usage of facets. Data logged over a period of 6 month (Jan 09 – June 09). 147
Fig. 8-7: Merging the results of two different retrieval functions. Clicked items are marked with a star. 155
Fig. 8-8: Click distribution of (B) and (BF) 158
Fig. 8-9: Frequency of clicks per query of (B) and (BF) 158
Fig. 8-10: Click distribution of (B) and (BC) after removal of duplicates 159
Fig. 8-11: Histogram of clicks per query of (B) and (BC) after removal of duplicates 159
Fig. 8-12: Hierarchical structure of the GQM model 162
Fig. 8-13: Evaluation process overview..... 163
Fig. 8-14: Density plot of the execution time. The red curve represents conventional tools and the blue curve represents YASA. 169
Fig. 8-15: Density plot of the mouse movement. The red curve represents conventional tools and the blue curve represents YASA. 169
Fig. 8-16: Prediction profile. The left picture shows the estimated average execution time of a task in case conventional tools are used. The right picture shows the average execution time for task number 5 and its dependence on the used tool. .. 171

Fig. 8-17: Prediction profile. The left picture shows the estimated average execution time of a task in case YASA is used. The right picture shows the average execution time for task number 5 and its dependence on the used tool. 172
Fig. 8-18: Questionnaire results for questions using IsoMetrics (scale from 1 to 5). The blue points represent the average value and the black error lines represent one standard deviation. 173
Fig. 8-19: Results for questions 3, 4, 5, 7, 38, and 39. 174

List of Tables

Table 2-1: Sample corpus consisting of three different documents. The document index terms are the result of stop-word removal and Porter's stemming algorithm. 18
Table 2-2: Overview of term weighting components. Let N be the number of documents in the collection, and let T be the number of terms in the collection. Further, let t be a term, let d be a document, let w_i be the weight of term i, and let $occ_{t,d}$ refer to the number of times term t occurs in document d. Further, let df_t be the number of documents in which term t occurs. Then, the term weighting components can be defined as follows. 23
Table 2-3: Data Retrieval vs. Information Retrieval after [Van Rijsbergen 1979] 39
Table 2-4: Query (in)dependent evidence factors. +: applicable, -: not applicable, o: applicable if the institution's privacy policy allows the usage of personal data 45
Table 3-1: Overview of hybrid-based filtering approaches 64
Table 4-1: RDF Schema primitives 76
Table 4-2: OWL language features 78
Table 6-1: Overview of the examined features, the corresponding sources on which the features are evaluated, and the methods applied. 105
Table 6-2: Statistics of result distribution for each sample query. 128
Table 8-1: YASA index statistics in July '09. 142
Table 8-2: Number of terms per query. 146
Table 8-3: Number of Boolean operators per query. 147
Table 8-4: Statistics of query duplicates. 147
Table 8-5: Queries per session. 148
Table 8-6: Result pages of queries viewed per session. 148
Table 8-7: Term modifications per query session. 149
Table 8-8: Clicks per query. 149
Table 8-9: Training- and test-set. 151
Table 8-10: Text categorization performance for different text processing pipelines (whitespace tokenization and lower case filtering was always applied). Optimization was conducted on the training set using a stratified 10-fold cross validation so that no bias on the test set evaluation is introduced. 152
Table 8-11: ML Classification performance of step 1 and step 2. 153
Table 8-12: Confusion matrix of ML step 1 153
Table 8-13: Confusion matrix of ML step 2 153
Table 8-14: Performance comparison between KB and ML. 154
Table 8-15: Size of the PRPZ-Share logs at the time the analysis was conducted. 157
Table 8-16: Goal, questions, methods and metrics 162
Table 8-17: Educational background of participants. 164
Table 8-18: Academic degree 164

Table 8-19: List of tasks. *: Baseline task. #: A gene name not to be disclosed. 164
Table 8-20: List of questions used in the questionnaire.....165
Table 8-21: Fraction of wrong answers per task. 168
Table 8-22: Influence of the order (Group 1 vs. Group 2) on the performance.....169
Table 8-23: Fixed Effect Tests using the F-test. Statistical significant values are marked by *. 170
Table 8-24: Parameter estimates. The estimates tell how much slower (positive values) or faster (negative values) the performance is on average depending on the task, tool, or task*tool combination. Significance is determined using the t-test. Statistical significant effects are marked by *. 170

Abbreviations

ACL	Access Control List
DB	Database
HCLS	Health Care & Life Sciences
ID	Identifier
idf	inverse document frequency
IR	Information Retrieval
KB	Knowledge Base
KE	Knowledge Engineering
LSI	Latent Semantic Indexing
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
OBIR	Ontology-based Information Retrieval
OWL	Web Ontology Language
PRPZ	Pharmaceutical Research Penzberg
QID	Query Identifier
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
R&D	Research & Development
SCC	Strongly Connected Component
SOP	Standard Operating Procedure
SVM	Support Vector Machine
tf	term frequency
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
VSM	Vector Space Model
WWW	World Wide Web
YASA	Your Adaptive Search Agent

Publications of the author related to this thesis

A. Kohn, F. Bry, S. Klostermann, and A. Manta, 2007, 'Concepts for an Intelligent Information Portal in Pharmaceutical Research', in *Proceedings of the I-Semantics '07 conference*, Graz, Austria.

A. Kohn, F. Bry, and A. Manta, 2008, 'Exploiting a Company's Knowledge: The Adaptive Search Agent YASE', in *Proceedings of the I-Semantics '08 conference*, Graz, Austria.

A. Kohn, F. Bry, and A. Manta, 2008, 'Professional Search: Requirements, Prototype and Preliminary Experience Report', in *Proceedings of the IADIS International Conference WWW/Internet 2008*, Freiburg, Germany.

Curriculum Vitae

Name	Alex Kohn
Date of birth	21.01.1981 in Timişoara, Romania
Nationality	German
E-Mail	alex.kohn@pms.ifi.lmu.de

1987 – 1992	Elementary school
1992 – 2000	Secondary school at the Lion-Feuchtwanger-Gymnasium in Munich
2001 – 2006	Study of Bioinformatics at the Ludwig-Maximilians-University / Technical University of Munich.
Since 2006	Doctoral student at the Ludwig-Maximilians-University, department of computer science, in collaboration with Roche, department of Pharma Research Scientific Informatics, Penzberg, Germany