Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation

Richard Berendsen

Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
Prof. dr. D.C. van den Boom
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op donderdag 12 november 2015, te 10:00 uur

door

Richard Willem Berendsen

geboren te Meppel

Promotiecommissie

Promotor:	Prof. dr. M. de Rijke	Universiteit van Amsterdam
Copromotor:	Dr. K. Balog	Universitetet i Stavanger
Overige leden:	Dr. E. Kanoulas	Universiteit van Amsterdam
	Dr. C. Monz	Universiteit van Amsterdam
	Prof. dr. V. Petras	Humboldt-Universität zu Berlin
	Prof. dr. A. de Vries	Technische Universiteit Delft
	Prof. dr. M. Worring	Universiteit van Amsterdam
Faculteit:	Faculteit der Natuurwetenschappen, Wiskunde en Informatica	

SIKS Dissertation Series No. 2015-24
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

The research was partially supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 258191 (PROMISE Network of Excellence).

Printed by Off Page, Amsterdam

ISBN: 978-94-6182-611-4

Acknowledgements

Maarten de Rijke leads the Information and Language Proccesing Systems group (ILPS). Over the years, ILPS has been growing into a large information retrieval and natural language processing research group that operates at the crossroads between academia and industry. The atmosphere of collaboration in this group, its active interaction with industry, and the many ties with other researchers in Europe and beyond make it a very dynamic environment, with many opportunities for interesting research. The research chapters in this dissertation are five such opportunities, seized.

I joined ILPS in September 2010, with Maarten as my promotor. I would like to thank Maarten for his respectful and professional weekly supervision, his creativity, his overview of the field and its current fronts, and his ability to tell a structured story from any angle, at any level of abstraction.

I am also much indebted to all of my co-authors, among whom my co-promotor Krisztian Balog, on whose work in expertise retrieval I could build, and whose comments on a draft of this dissertation helped me to make many improvements. Manos and Wouter, my paranymphs, had tested the waters with pseudo test collection generation, and I could count on their help and contributions all along the way to several conference deadlines. The same is true for Bogomil, with whom I enjoyed many programming sessions in the early years of my PhD. Edgar was also involved in several of my research chapters, and his insightful comments and always instantaneous help have been valuable.

The enthousiasm and generosity of all ILPSers that have come and gone in exchanging ideas and feedback over coffee or drinks has been tremendously encouraging, as was the contact with researchers of neighbouring labs. Beyond work, we shared good times on the football and volleybal courts, in the city, abroad at conferences and a winter school, and over the chess board. I would also like to thank the non-scientific staff for their reliable and enthousiastic support.

The academic community thrives on interactions between researchers of different universities and institutions. The EU-funded PROMISE Network of Excellence that funded a large part of my PhD has been a very friendly and inspiring environment for me. I have learned a lot in our project meetings and research exchanges. The project goals and ambitions allowed us to recognize and shape the opportunities for the research that went into this dissertation. I thank the reviewers of my publications for their helpful comments and suggestions. And I thank the members of my committee for their commitment, corrections, and time.

Bart Nijstad, thank you for your beautiful art.

To my family, my friends, and to Marloes, your love and support mean the world to me.

Richard Berendsen
September 2015

Contents

1 Introduction

There is a growing diversity of information access applications. While general web search has been dominant in the past few decades, a wide variety of so-called *vertical search* tasks and applications have come to the fore. Vertical search is an often used term for search that targets specific content. Examples are Youtube video search, Facebook graph search, Spotify music recommendation, product search [190], expertise retrieval [24], and scientific literature search [234]. The heavy usage of such applications is testimony to the importance of vertical search. Verticals are gaining ground also in general web search. This is connected with estimating what kind of content a user is looking for, i.e., the *information need* of the user. If a query is entity-oriented, a modern web search engine will display a box with information from a knowledge base about relevant entities. Web search engines typically also include videos, images, or scientific articles to their search result page if deemed appopriate.

In a vertical search application, we typically have some background knowledge about the context in which search is taking place. We may know something about the user population, about the tasks they wish to perform, about their information needs, and about the information objects in the collection we make available to them. In this dissertation we focus mainly on background knowledge related to information needs and information objects.

In web search, Broder [38] observed that there are three broad classes of information needs underlying web search queries: informational, navigational and transactional information needs. People who enter an informational query want to learn more about a subject, e.g., "gardening." A navigational query means that a user just wants to reach a particular webpage, e.g., "facebook." Transactional queries indicate a desire to do something, e.g., download software or buy a piece of clothing. In a vertical search application, there may be classes of queries different from those in web search [153]. For example, in blog search so-called context and concept queries were observed. Both are informational queries. Context queries aim to find contexts in which an entitity such as a politician or a product is mentioned. Concept queries are aimed at finding blogs on an interest area of the user, such as "stock trading" [153].

Characteristics of information objects have also been studied in web search. Modern web search engines combine web documents, images, and videos on their search result page; each type of information object requires a different retrieval approach. Zooming in on ranking web documents, the similarity of the documents to the query is important. But

query-independent document features are also used. The PageRank score of web pages is a well-known example [167]. Learning to rank algorithms [138], which represent the current state of the art in web search, are able to combine many of such features into a single ranking. In a vertical search application, we may know more about the information objects in our collection. For example, in product search we may have a complete catalog of products. This catalog may hold information on the product type, manufacturer, price, availability, and so on. Therefore, the search engine result page may offer filters to narrow down the search results. Users can restrict search results to a certain price range, or to a certain brand, and so on. This kind of search is called faceted search [222].

A particular source of background knowledge are annotations. An example of these are keywords (e.g., "information retrieval") added to scientific articles. Sometimes, authors of papers are asked to select these keywords from a thesaurus of research areas. Sometimes, curators of a collection of articles label articles. Annotations added by people are valuable. They enrich items in the collection, making it easier to group, browse and retrieve these.

Annotations may be of very high quality, created by experts in the domain, as in the above scientific literature example. In other domains we may have more noisy annotations. Tweets, for example, contain hashtags, which may be used to indicate that a tweet is part of a larger discussion. Even noisier are clicks on search results, which can be viewed as user annotations. They often indicate some level of interest in the result being clicked, but they may also have occured by chance.

In this dissertation, we will study several vertical search applications. For each of them, we focus on using characteristics of information needs or information objects that are unique to the domain of the search application. We use such characteristics for query understanding, for optimizing and evaluating search algorithms, and for error analysis.

1.1 Research outline and questions

The shared theme of the research in this thesis is to explore how domain knowledge can play a role in understanding, design and evaluation of vertical search applications. We develop this theme in a number of case studies. We start out with people search. People search is about searching for people. This happens in web search increasingly [144, 217], on Facebook, and many dedicated people search engines exist, such as pipl.com, 123people.com, and, of course, a range of dating sites.

Motivated by the wish to gain a better understanding of the usage of a dedicated people search engine, in [231], we perform a query log analysis and distinguish two types of person name queries: ones that are only occasionally entered and also do not refer to a famous person (low-profile), and ones that are often entered and/or refer to a famous person (high-profile). High-profile queries were subdivided further into queries that were most likely entered in connection with a recent event or a recent hype (event-based queries), and queries that most likely refer to well-known persons who are known for more than any single event (regular high-profile queries).

In Chapter 3 we follow up on this observation, with the aim of investigating if this classification can be automated. If this can be done with reasonable accuracy, it becomes an option to take the predicted query class into accont to improve search algorithms or

result presentation. We also want to understand which features (e.g., news mentions, Wikipedia mentions, search volume) most strongly influence the decisions of the query classifier. This gives insight in the strengths and weaknesses of the classifier, and it also sheds light on the relationship between the context in which person names are mentioned and the class of the query. Our research question in this first case study is:

RQ1 Is it possible to classify person name queries submitted to a people search engine as either low-profile, event-based and regular high-profile classes such that reasonable agreement with human classifying decisions is achieved? Among a set of features capturing online context of person names and interactions of people search engine users, which features have the largest influence on classification decisions?

A major issue in people search is the high degree of ambiguity of person names [10]. To address this ambiguity, several evaluation campaigns were organized around the task of grouping or clustering search results around persons being referred to [11–13]. For that task, person name queries were submitted to a general web search engine. For each query, a clustering for the top-100 results was required. In Chapter 4 we are interested in the differences between this setting and the setting of a vertical people search engine. The latter has a different presentation of search results, and includes much more social media profiles. In particular, we are interested in how well the lightweight and state-of-the-art hierarchical agglommerative clustering (HAC) methods do in this setting.

We find that the main problem for the abovementioned HAC approaches is that social media profiles are textually sparse, and they do contain "boilerplate" elements that makes profiles from the same platform (e.g., Facebook) look very similar to each other. These profiles present so difficult a problem that we propose to treat them altogether differently. First, we set out to cluster social media profiles only. We explore the use of non-textual features, such as cross-links between profiles, evidence from query log files, and a visual clue: the user profile picture.

Meanwhile, we cluster the rest of the documents as before. Finally, we merge the social media clusters with the "regular" clusters. The intuition is that these regular clusters now contain more evidence, and there is a higher chance that elements in it will match with elements in social media clusters. Our main research question in this chapter is:

RQ2 State-of-the-art hierarchical aglommerative clustering (HAC) methods for clustering web search results for person name queries break on search results of a people search engine, which contain many social media profiles. Can we remedy this problem by treating social media profiles differently from regular web documents, clustering the two types of documents separately and then merging the clusterings back together?

While in the research outlined above we proposed algorithms, for our next research question we shift our attention towards evaluation. And from general people search we move to *expert profiling*. Balog et al. [20] define the task of expert profiling as producing a ranking of a set of known knowledge areas for a given expert. Expert profiling is related to the task of *expert finding*, which Balog et al. [20] operationalize as ranking experts for a knowledge area. To evaluate expert profiling algorithms, they use the UvT expert

collection as a test bed. This test collection contains expert profiles with knowledge areas that each expert has voluntarily selected for him- or herself from an alphabetical list (*self-selected profiles*). The evaluation has been done as in a document retrieval task. To see how this works, think of the expert as a query, and think of their selected knowledge areas as relevant items. Thus, standard evaluation metric scores have been reported. A major concern is that the self-selected profiles may be sparse, resulting in unrealistically low scores. Therefore, in Chapter 5, we conduct a self-assessment experiment where we ask experts to judge a profile we generated for them. These *judged system-generated* profiles can also be used as a set of ground truth to evaluate expert profiling algorithms.

First, we analyze the quality of the generated profiles. We look into the characteriscs of difficult experts. We consider the number of knowledge areas an expert has, the number and *types* of documents authored by an expert, the job he or she fullfills, and so on. We also perform a content analysis of free-text feedback supplied by experts during the self-assessment experiment, to identify issues with the generated profiles that experts agree on.

Next, we take a step back. Researchers developing expert profiling algorithms benefit from quick and automatic evaluation using a set of relevance assessments. Recall that self-selected profiles have so far been used for this. Now, we release the judged system-generated profiles. If researchers use these for evaluation, what will be the differences in evaluation outcomes? The judged system-generated profiles are different in a number of ways, e.g., there are fewer of them (not all experts participated in the assessment experiment), they contain additional knowledge areas, and for these knowledge areas we now have a level of expertise on a scale of one to five. We look at how each of these differences affects evaluation outcomes. Our research question in this expert profiling study is:

RQ3 We ask experts to judge profiles we generate for them. What is the quality of our generated profiles? Which experts are hard to generate a profile for and why? Previously, evaluation of expert profiling algorithms has been done by using profiles of knowledge areas that experts selected from an alphabetical list. If we use judged system-generated profiles for evaluation, what are differences in evaluation outcomes?

The above research question is mainly about evaluation. In our next case study, we explore an idea that can be used for both evaluation and optimization. We stay in an academic context, but look at a different task: scientific literature search. Given an information need, a user wants to find all relevant literature. Since the literature is vast and rapidly growing, this is a hard goal. To support search, there exist fully indexed scientific literature collections. People well familiar with indexing terms can help others formulate their information need in the vocabulary of the index, and locate additional relevant documents. In Chapter 6, we explore methods to re-use the huge annotation effort gone into such collections for the purposes of evaluation and optimization of search algorithms. Typically, so-called IR test collections [193] are used for both. The main ingredient for such test collections is a set of descriptions of information needs. For each information need, usually a single query is formulated, and a set of relevant documents is obtained. Especially that last step is costly: it takes a lot of human effort.

We turn the process of creating a test collection on its head. We start by locating relevant documents. We do this through the annotations: the indexing terms. We propose a number of strategies. In the simplest, we start from a group of documents that is indexed with a particular topical indexing term, e.g., "information retrieval." We think of all these documents as relevant to an information need. In our next strategy, we recognize that perhaps not all indexing terms can represent realistic information needs. Often, they are overly general: for example, no PhD student ever searches for all papers on "information retrieval." Therefore, we experiment with sampling combinations of indexing terms. Having obtained a set of relevant documents, we consider two approaches to generate a query. In the first, we simply concatenate all words that constitute the indexing terms. In the second approach, we sample discriminative words from the indexed group of documents. The Cartesian product of our strategies for sampling relevant documents and query terms yields six recipes for generating a test collection. Because they are generated automatically, we refer to them as pseudo test collections (PTCs).

How do we evaluate the usefulness of a PTC for evaluation and for optimization? For evaluation, we look at how a set of ten retrieval algorithms is ranked by a traditional hand-crafted test collection. We compare this to how the same algorithms are ranked by our PTCs. For optimization, we consider the optimization of a particular algorithm, as a proof-of-concept. We choose a learning to rank (L2R) algorithm. L2R approaches are increasingly common in information retrieval, and represent the current state-of-the-art [138]. We compare two scenarios. In both, we test on a hand-crafted test collection. In the first scenario, we train on a different hand-crafted test collection. This is the usual way in empirical machine learning research: we have some human-annotated ground truth, and we optimize on it. The hope is that the optimized algorithm will perform well in the real world, on different queries. To gain an estimate of this, we evaluate the optimized algorithm on an unseen set of test queries. In the second scenario, we test on the same unseen test queries. But we optimize on a PTC. That way, we gain an estimate of how well our search algorithm would do in the real world if we do not have any human ground truth to optimize on. Our research question in this study is:

RQ4 Collections of scientific literature are sometimes indexed with different kinds of annotations. Can the result of these annotation efforts be used to generate a pseudo test collection (PTC)? When retrieval algorithms are evaluated using a PTC, are they ranked as they would be by a hand-crafted test collection? And when we test a L2R algorithm on a hand-crafted test collection, what is the best way to train it: on a different hand-crafted test collection, or on a PTC?

After exploring the idea of creating a PTC for scientific literature search—where content and annotations are of a high quality—we perform a more extensive study of the same general idea in the much more noisy domain of microblog search [166]. In particular, we study Twitter, where posts are also called "tweets." Microblogging platforms are real-time, in the sense that many posts are about current events. And typically authors of posts have followers. Followers receive all post of their "followees" in real-time, if they are logged-in. Some posts are personal in nature, other posts are of interest to a broader public. If a follower wants to pass on a tweet to his or her own followers, he or she can "re-tweet" it. It is reasonable to expect that some users of a microblog search engine

are looking for recent updates on a topic of their interest. In an initial benchmarking task around microblog search, systems had to retrieve tweets in chronological order, omitting non-relevant tweets. Later editions of the same benchmark required systems to order tweets by relevance, as in most retrieval settings. This is the task we study. In our fifth case study (Chapter 7), we focus on the optimization of microblog search algorithms. As before, we compare this with optimization on hand-crafted microblog test collections [166, 212]. Testing is always done on hand-crafted collections.

The main idea to the creation of PTCs here is to use hashtags. Hashtags are annotations by the authors of a microblog post. Hashtags may refer to events, recent hypes, causes or campaigns started anywhere in the world, TV-shows, particular genres of jokes (#blackparentquotes), emotions shared by many people (#thankgoditsfriday), and so on. We assume that tweets with the same hashtag share a topic, whatever that topic is. Unlike in scientific literature collections, there may be many tweets on the same topic, but without the hashtag. Also, there may be tweets on a different topic, but with the same hashtag (e.g., spam). In short, hashtags are noisy. Still, we construct PTCs here, taking the set of tweets that share a hashtag as a starting point.

The next step in the PTC creation process is to generate a query for each hashtag. We do this by sampling discriminative terms from posts with the hashtag. This is our first PTC-generation recipe. To give algorithms optimized on a PTC a chance to capture recency, we propose a simple method to derive a query-timestamp for each hashtag-derived query, from the publication dates of the tweets with that hashtag. This is our second PTC-generation recipe. In microblog search, other factors besides recency play a role. Indeed, quality may be captured in relevance assessments of IR test collections. Among documents that are on-topic, higher quality documents may receive a higher relevance grade, or may have more chance to be judged relevant at all. In the creation of the microblog search test collections we use, assessors took 'interestingness" of tweets into account. They did not only consider if tweets are on-topic, but also if they contain additional information [166]. Our third PTC-generation recipe aims to capture interestingness of tweets as well. It takes into account only features that are query-independent, e.g., does the Tweet contain a link, does it contain a username, etc.

The three generated PTCs are compared in terms of how useful they are as an optimization test-bed. We optimize three commonly used L2R methods. All three are optimized in two steps. First, the query-dependent features are optimized. Most query-dependent features are retrieval scores of full-blown retrieval algorithms. Consider for example language modeling with Dirichlet smoothing. It has one free parameter, which we tune on a PTC. Second, a function is learned that combines all L2R features, both query-dependent and query-independent. Our research question in this study is:

RQ5 Hashtags are used in microblog search to indicate that a tweet is part of a larger discussion. We assume that tweets sharing the same hashtag share the same topic, by and large. Can we build on this assumption to generate a PTC? And when we test an L2R algorithm on a hand-crafted microblog search collection, what yields better performance: training on a different hand-crafted collection, or training on a PTC? We consider three recipes for generating a PTC. What is their relative merit as training material? We consider three strong L2R algorithms. How will our findings vary with a different choice of L2R algorithms? And, how succesful can free parameters of individual retrieval algo-

rithms such as language modeling be tuned on our PTCs?

We answer the above five questions in our five research chapters (Chapters 3–7), in the discussion and conclusion sections of each chapter. In Chapter 8, we summarize our findings.

1.2 Main contributions

The contributions in this thesis are observational and algorithmic in nature. We also contribute resources, namely test collections. We list our contributions below.

Analyses and algorithms

A feature analysis for person name query classification Weerkamp et al. [231] propose a query classification scheme for person name queries, and perform a classification experiment. We repeat this experiment and provide insights in the importance of features that capture online context of person names and interaction of users with a people search engine.

A two-stage clustering model for people search results We observe that state-of-the-art algorithms for clustering general web search results for person name queries break down when applied to results of a people search engine. We recognize a main problem: the larger fraction of social media profiles in people search engine results. We obtain a dramatic performance increase with a two-stage clustering model that clusters social media profiles and other web-documents separately and then merges the two clusterings.

An analysis of the quality of generated expert profiles We collect and analyze free-text feedback from employees of a Dutch university on profiles containing knowledge areas we ranked for them using a combination of strong expert profiling algorithms. We also perform a quantitative error analysis using explicit judgments they provided for knowledge areas.

An analysis of expert profiling evaluation outcomes The judged system-generated profiles obtained during the self-assessment experiment described above can be used for the evaluation of expert finding and expert profiling tasks. This holds true also for the self-selected profiles, which contain knowledge areas employees had selected voluntarily from an alphabetical list. We prove an extensive analysis of the differences in evaluation outcomes for the expert profiling task. We also offer considerations about the use of both sets of ground truth for the expert finding task.

Methods for generating test collections for scientifc literature search Inspired by the automatically generated test collections for known-item search in [15], we build pseudo test collections for the evaluation and optimization of algorithms performing a more general document retrieval task: scientific literature search.

Methods for generating test collections for microblog search We apply the idea of using annotations for building pseudo test collections for use in the more noisy domain of microblog search. We extend our methods, taking into account recency of documents and quality indicators in the generation process. We report on extensive experiments in which pseudo test collections are used for optimization.

Resources

The Dutch People Search Results Dataset Search results of a people search engine for 33 popular queries submitted to a Dutch people search engine were crawled. The crawl is made available on `http://ilps.science.uva.nl/resources/ecir2012rdwps`. Manually created ground truth is included for the task of clustering search results around the individuals they refer to.

The TU Expert Collection We have crawled the TU Webwijs website, providing an update of the UvT Expert Collection [20]. We performed a self-assessment experiment and make available the relevance judgments we obtained in it. These judged system-generated profiles can be used for the evaluation of expert profiling and expert finding tasks. We also release the self-selected profiles that contain knowledge areas that employees had originally selected from an alphabetical list: these profiles are an update of the self-selected profiles that have previously been used for the evaluation of both tasks [20]. The TU expert collection can be found online at `http://ilps.science.uva.nl/tu-expert-collection`.

1.3 Thesis overview

This thesis is organized in eight chapters. After a background chapter, we present five research chapters, each containing a case study of a particular vertical search application.

Chapter 2—Background We place our research in the broader context of information retrieval. After a brief outline of that field, and of vertical search applications in particular, we review people search, expertise retrieval, scientific literature search and microblog search literature.

Chapter 3—Query Classification in People Search We classify person name queries submitted to a people search engine into low-profile, event-based and regular high-profile queries. We investigate the importance of, amongst others, news, Wikipedia and search log features for this task.

Chapter 4—Result Disambiguation in People Search We study the problem of grouping people search engine results around the individuals they refer to. We find that social media profiles and other web documents should be treated altogether differently and propose a two-stage clustering algorithm that does this.

Chapter 5—On the Assessment of Expertise Profiles We perform an evaluation study of expert profiling algorithms [20]. We ask experts to judge profiles of knowledge areas that we generated for them in a self-assessment experiment. We perform an

error analysis of the generated profiles and we analyze evaluation outcomes when the judgments obtained in the self-assesment experiment are used as ground truth.

Chapter 6—Pseudo Test Collections for Scientific Literature Search We study scientific literature search. In particular, we re-use the annotation effort of professionals who indexed a collection of scientific articles to generate a pseudo test collection. This collection can be used for evaluation and optimization and we report on both.

Chapter 7—Pseudo Test Collections for Microblog Search We apply the same general idea of generating pseudo test collection in the microblog search domain, which is more challenging due to the noisy nature of tweets. We extend our generation methods. We report on extensive optimization experiments in which our pseudo test collections are used.

Chapter 8—Conclusions We summarize our main findings and point out directions for future research.

1.4 Origins

For each research chapter we list on which publication it was based, and we discuss briefly the role of co-authors.

Chapter 3 This chapter is based on Berendsen, Kovachev, Meij, De Rijke, and Weerkamp [30] "Classifying Queries Submitted to a Vertical Search Engine." *Proceedings of the 3rd International Web Science Conference.* ACM, 2011. We perform additional experiments based on the same classification scheme, ground truth annotation interface, and experimental setup used in [231], to the design of which all authors contributed. Berendsen and Kovachev performed the classification experiments and analysis, and all authors contributed to the text.

Chapter 4 This chapter is based on Berendsen, Kovachev, Nastou, De Rijke, and Weerkamp [31] "Result disambiguation in web people search." *Advances in Information Retrieval. Proceedings of the 34th European Conference on IR Research.* Springer Berlin Heidelberg, 2012. We mine popular queries from the query logs of a Dutch people search engine, re-issue these queries, crawl all search results, and perform clustering experiments. All authors contributed to the design of the algorithms and to the text. The experiments and analysis were carried out mostly by Berendsen and Kovachev.

Chapter 5 This chapter is based on Berendsen, De Rijke, Balog, Bogers, and Van den Bosch [29] "On the assessment of expertise profiles." *Journal of the American Society for Information Science and Technology.* 2013. We perform a self-assessment experiment, which was designed and carried out by all authors save Berendsen. The free text feedback of experts was coded by Bogers and Van den Bosch. The scope and the design of the analyses—the error analysis, the analysis of evaluation outmoces—was mostly developed by Berendsen, De Rijke and Balog, and carried out and reported on mostly by Berendsen.

Chapter 6 This chapter is based on Berendsen, Tsagkias, De Rijke, and Meij [32] "Generating Pseudo Test Collections for Learning to Rank Scientific Articles." *Information Access Evaluation. Multilinguality, Multimodality, and Visual Analytics. Proceedings of the Third International Conference of the CLEF Initiative*. Springer Berlin Heidelberg, 2012. All authors contributed to the design of the algorithms and experiments, and to the text. The experiments were carried out by Berendsen and Tsagkias.

Chapter 7 This chapter is based on Berendsen, Tsagkias, Weerkamp, and De Rijke [33] "Pseudo test collections for training and tuning microblog rankers." *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval.* ACM, 2013. All authors contributed to the design of the algorithms and experiments. Experiments were carried out mostly by Berendsen, with contributions from Tsagkias. All authors had substantial contributions to the text.

Work on other publications also contributed to the thesis, albeit indirectly. We mention five papers: (1) a query log analysis of a Dutch people search engine [231], with contributions in analysis, a classification experiment and writing; (2) an analysis of the logs of the search engine for The European Library [86], with contributions in analysis and writing; (3) the editing of chapter four in [1], on challenges in test collection design and exploitation; (4) work in which an approach to microblog search was proposed that uses syntactic features [201], with contributions in reproducing results from our L2R experiments [33] and in writing; (5) an analysis of how experts annotate online content with regards to its impact on the reputation of particular brands [170], with contributions in writing.

2 Background

We first give a brief overview of the field of information retrieval (IR) in Section 2.1, highlighting algorithms that we will use in our research chapters. Then, in Section 2.2, we discuss evaluation methodology in IR, with a focus on methods we use or build upon in this dissertation, i.e., Cranfield style test collection based evaluation and the automatic generation of test collections. Finally, in Section 2.3, we give background on the domain-specific search applications that are the subject of Chapters 3–7.

2.1 Information retrieval

Usually, in information retrieval research, there are at least the following key ingredients: (1) a user with an information need; (2) a collection of information objects; (3) a notion of relevance, i.e., the information need can be satisfied by certain *relevant* information objects in the collection; and (4) a system, which generates a response to to the information need, and which defines the set of possible interactions for the user, e.g., entering a query in a text-box, and interacting with a search engine result page (SERP).

In this chapter and in this dissertation, we will see many instantiations of each of these ingredients. In this section, our main focus is a quite general and much studied instantiation, "ad hoc" document search. There are different kinds of ad hoc search, e.g., searching with a general purpose web search engine, searching for scientific articles, or searching for news. Ad hoc search scenarios have in common that the user has an information need which is expressed in a query of, typically, a few words. The query is fired against an index of text documents, and the results that best match the query are retrieved. We briefly describe the main underlying algorithms in use today, giving only as much detail as is useful for understanding the experiments we perform in later chapters.

Besides retrieval of documents, text categorization and document clustering are recurring tasks in information retrieval. In Chapter 3 we study query classification, and in Chapter 4 we perform search result clustering. We give background on these algorithms in Section 2.3, where we discuss the domain-specific search applications from the above-mentioned chapters. Before we give an overview of retrieval algorithms for the vanilla ad hoc retrieval scenario, we give a brief history of active research areas in IR.

2.1.1 A very brief history of IR

In the early decades of information retrieval, from the 1950s until the early 1990s, much research was done on retrieval of scientific literature. In the article collections, often only metadata of each article was available, such as its title, authors, keywords, and abstract [193]. Interestingly, even as recently as 2007 and 2008, this very setting was studied in the "domain-specific" track of a yearly European evaluation activity, CLEF (Cross Lingual Evaluation Forum) [173, 174]. This shows that retrieval, even for a narrow domain like this, is highly challenging. In Chapter 6, we perform experiments on data collections from this track. In the 1990s, much attention was devoted to the ad-hoc search scenario, often with collections of news articles. Document collections were increasing in size now [193]. From the 2000s onwards, a growing diversity of search settings was studied, and collections grew even bigger [193]. The dominant application now is general purpose web search, which has large document collections. The diversity of tasks studied increased as well, increasing the need for new algorithms and evaluation methodology. It is easy to expand on the list of domain-specific search settings we listed in Chapter 1 with more examples that received much attention, e.g., patent retrieval [140], image retrieval [206], book search [124], and so on. The scenarios from Chapters 3–7 fit in this general picture.

2.1.2 Algorithms for document retrieval

In document retrieval, documents are compared to queries. To do this, all algorithms discussed in this section segment text into words first, through *tokenization*. Then, optionally, *stopwords* may be ignored for the purposes of retrieval. Typically, very high frequency words are treated as stopwords. Intuitively, this will prevent a query from matching too many documents. Next, again optionally, terms may be normalized. This is typically done by *stemming*, where inflections of the same stem are treated as the same term; in this way query terms can potentially match more document terms, reducing the chance of missing some important information. Once queries and documents are both segmented into words, or *terms*, we can think about how to treat matches between query and document terms. We discuss a selection of influential algorithms proposed in the literature, guiding the selection by which algorithms we use in Chapter 6 and 7.

Boolean retrieval The first approaches to information retrieval, prominent in the 1950s and 1960s [145], are based on whether or not exact matches occur; they are *boolean* retrieval models. Operators to combine query terms are: `AND`, `OR` and `NOT`. In Chapters 6 and 7 we use two flavours of boolean retrieval. Both use only the `AND` operator. In the first approach, all query terms must occur in the document in same order as in the query. In the second approach, all terms must match as well, but their order does not matter. Both models allow setting a window size of a particular number of terms. If we slide a window of that size over the document terms, all terms must occur in that window at least once for the query to match. We refer to the first model as "BOW" (boolean ordered window), and to the second as "BUW" (boolean unordered window). Boolean queries define a set of documents to be retrieved, but no ranking of these results. Adding terms with the `AND` operator makes queries restrictive, and yields narrow result sets. `OR`

queries can match very large result sets. In larger collections, both soon lead to unsatisfactory results: we either miss many important documents or match many uninformative documents, as a result of which a user has to examine many search results.

Tf-idf and the vector space model To make ranking possible, the concept of *term weighting* was established. Two basic observations lie at the basis of so called *tf-idf* weighting scheme [215]; we can weight term importance for terms occuring in documents, giving more importance to terms which: (1) occur more frequently in a document; and (2) occur in less documents. The first goal is achieved by taking into account the (relative) *term frequency* $tf(t, d)$ of a term t in a document d; the second by defining some variant of *inverse document frequency* (*idf*). In all variants of idf, the fraction $\frac{1}{df(t)}$ plays a role, where $df(t)$ is the number of documents a term t occurs in.

One way to use term weighting to compute a score between a query q and a document d is to represent both query and document as a vector, with one element for each word in a vocabulary, e.g., all words that occur in the document collection. Each element contains a weight, e.g., a tf-idf weight. Then, the cosine similarity between query and document vectors can be used as a score to rank documents.

The probability ranking principle and BM25 It is also possible to approach and formulate the ranking problem probabilistically. The most direct formulation here strives to estimate the probability $P(R = 1|d, q)$ that a document d is relevant for query q. The famous *probability ranking principle* states that ranking documents by this probability should be the most useful ranking to the user, assuming that the relevance of a document to a query does not depend on the relevance of results that the user may have already seen [187].

The estimation of $P(R = 1|d, q)$, however, is not straightforward if we have seen no examples of relevant documents for query q. One influential and very successful retrieval algorithm, BM25 [186], can be interpreted as a probabilistic model, where several heuristic choices have been made to estimate unknown probabilities; see [185] for a nice overview of relations between BM25 and other, probabilistic retrieval methods.

Tuning the parameters of BM25 We treat BM25 in a bit more detail, because of its wide use even today, and because in Chapter 7, one of the retrieval models we show detailed parameter tuning results of, is based on BM25 (See Figure 7.2a). To understand the effect of parameter tuning on retrieval scores, we reproduce the formula that computes the BM25 retrieval score of a document d for a query q; we base it on Indri[1] source code. In this formulation, d and q are modeled as *multisets* (bags) of terms t:

$$BM25(d, q) = \sum_{t \in q} w(t, q) \times tfn(t, d) \times idf(t), \tag{2.1}$$

with

$$w(t, q) = \frac{tf(t, q)(k_3 + 1)}{tf(t, q) + k_3}, \tag{2.2}$$

[1] Indri 5.1, available at `http://www.lemurproject.org/indri.php`

$$tfn(t,d) = \frac{tf(t,d)(k_1+1)}{tf(t,d) + k_1(1-b+b(\frac{\sum_{t\in d} tf(t,d)}{\Delta}))}, \tag{2.3}$$

and

$$idf(t) = \log(\frac{|D|+1}{df(t)+0.5}).$$

Here, $w(t,q)$ is the term weight of term t in query q, $tf(t,q)$ the raw frequency of term t in the query, $tfn(t,d)$ is the normalized term frequency of term t in document d, $tf(t,d)$ is the raw frequency count, $idf(t)$ is the inverse document frequency of t in the collection, D is the document collection, and Δ is the average document length in the collection.

The free parameters k_1, b, and k_3 are constants. We give an interpretation of these parameters based on [183]. Parameter k_1 can be used to manipulate the effect that $tf(t,d)$ has on document score for each term $t \in q$. A k_1 of zero has two effects: (i) the influence of parameter b disappears, and (ii) only term occurrence matters, not the frequency. A higher value for k_1 will increase the importance of term frequency, and also the influence of parameter b. Parameter b is used to normalize for document length. A b of zero indicates that the ratio of document length and average document length is not deemed important, term frequency in longer documents is not discounted, the assumption being that longer documents are long because they treat multiple topics. Setting b to one discounts term frequency in longer documents on the assumption that longer documents are simply more verbose [183]. Parameter k_3 is mainly useful for longer queries where words can appear multiple times. Setting k_3 to zero has the effect of not taking into account the frequency of a term in the query. Setting k_3 to a higher value favours terms that occur more often in the query. In our experiments in Chapters 6 and 7, we set k_3 to zero, because in our setting, we expect little or no repetitions of query terms.

Note that the BM25 formula uses term weighting, both taking the term frequency and the inverse document frequency into account. Indeed, both the Indri and the Terrier[2] implementations of tf-idf weighting that we use in Chapters 6 and 7 are in fact variations of the above BM25 formula. They have the same parameters k_1 and b, and these parameters play similar roles. The Terrier tf-idf implementation only allows manipulation of b, the *document length normalization* parameter: this is the retrieval model of which performance scores are shown in the abovementioned Figure 7.2a.

Language modeling In the late 1990s, *language modeling*, a technique originating from the field of speech recognition, was introduced in IR practice [176]. Instead of estimating $P(R = 1|d,q)$ directly, for which it would be convenient to have examples of relevant documents for q, it ranks documents by the *query likelihood* $P(q|d)$: the probability of observing the query q given d. A document d is then modeled as an observation from an underlying distribution; the multinomial distribution is a common choice [145]. By assuming conditional independence of query terms given the document, we can rewrite $P(q|d)$ as $\prod_{t\in q} P(t|d)$. There are many ways to estimate the latter. A successful approach is to use *Dirichlet smoothing*:

$$P(t|d) = \frac{tf(t,d) + \mu P(t|D)}{\sum_{t'\in d} tf(t',d) + \mu}, \tag{2.4}$$

[2]Terrier 3.5, available at `http://terrier.org`

where D is the document collection, as before in this chapter, and μ is a free parameter. The idea here is that a document is generated by a multinomial distribution, the parameters of which are influenced by a *prior* distribution: the Dirichlet distribution. The free parameter μ governs how much smoothing occurs: higher values give more weight to $P(t|D)$, the probability that a term occurs in the document collection. For a thorough study of this and several other smoothing methods, see [242]. The language modeling (LM) variant we use in Chapters 6 and 7 is the Indri implementation of it, which implements almost exactly Equation 2.4 above. We treated language modeling in some detail, because in Chapter 7, Figure 7.2b, we show the impact of varying μ on performance.

Divergence from randomness Another branch of probalistic retrieval models that we use in our experiments are *divergence from randomness* (DFR) models, developed in the early 2000's. These models take into account how term frequencies differ from what may be expected from a random process [5]. PL2 is one instantiation of the DFR framework. The Terrier implementation of it allows setting a parameter c, which represents a multiplying factor for the average document length in the collection, and thus allows one to influence the amount of document length normalization that occurs [3, 4]. Later, parameter free hypergeometric DFR models were developed [3], the DFRee model we use in Chapters 6 and 7 is a variation on these models. Later, a model was proposed that takes occurence of pairs of query terms in documents into account [171]. This model allows setting a *window size* within which the terms must occur. We use it in our experiments, referring to it as DFR-FD, and we opt for the *full dependence* model, in which the order in which both terms occur does not matter.

Pseudo relevance feedback All models discussed so far can be used without information about the relevance of any documents to a query. If a user of a search engine would provide some feedback on the relevance or non-relevance of some documents, a better query may be constructed, weighting query terms using the provided feedback [145]. An early approach allowing this is the Rocchio algorithm [188]. In the abscence of explicit *relevance feedback*, a popular technique is to treat a fixed number of retrieved documents from the top of an initial retrieval ranking as pseudo relevant documents; this is called *pseudo relevance feedback* (PRF). Lavrenko and Croft [130] propose a way to estimate $P(t|R)$, the probability of observing a term given observing a (pseudo) relevant document, which has become popular in PRF approaches. The Indri PRF implementation used in Chapter 6, for example, uses these term probabilities for *query expansion*. It ranks candidate terms by $P(t|R)$, selects a fixed number of terms from the top of this ranking, and weights the expanded query terms by $P(t|R)$ as well. The PRF implementation of Terrier uses a DFR model to rank candidate expansion terms occuring in the top retrieved documents [165].

Fielded retrieval models One step further than treating each document as a bag of words is to consider structure in documents. If term matches occur in the title of a document, it may be beneficial to give more importance to it than a match in the body of a document. The inference network retrieval model proposed in Metzler and Croft [152] allows matching terms in specific fields, as do the fielded extension to BM25:

BM25F [241], and the mixture of language models described in [163].

Other query and document features There are many other factors besides content based ranking that affect the usefulness of a document to a user at a given time in a certain context. A famous example is the "PageRank" of a web page [167], which is motivated by the interpretation of a link to web page as a recommendation of that page. The publication date of a document can also be an important factor, e.g., in microblog search, the topic of Chapter 7, where users are often looking for recent results [221]. The experience and status of the author of a scientific paper may affect the potential relevance of that paper, cf. Chapter 6. Document representations and query representations can be enriched by detecting and linking entities in them to a knowledge base [150]. If interaction data with search results is available, from user clicks useful signals can be obtained [53, 112].

Learning to rank To combine all these different aspects into a single ranking of documents for a query is a difficult challenge. *Learning to rank* (L2R) [138] algorithms have become the weapons of choice for this. They are *supervised machine learning* algorithms that can learn from labeled data about the relevance of documents. They fit the parameters of a function of a set of *features* in such a way that a *performance score* is optimized on a *training set* of labeled data. Then, the fit function can be applied to new, unseen queries (a *test set*). We will say a bit more about the features, the kinds of functions L2R models learn, and the performance scores they optimize now. Features are usually numerical representations of the aspects we discussed above, e.g., PageRank, recency, number of publications of the author of a paper, and so on. In modern web search systems, typically hundreds of features are used. Retrieval algorithms such as BM25 and language modeling are almost always amongst the features used. Because domain-specific features can easily be incorporated, L2R algorithms are ideally suited for domain-specific applications also. L2R algorithms differ in the kinds of functions they can learn. *Linear* algorithms learn *linear combinations* of the input features. Other models learn non-linear functions, e.g., gradient boosted decision trees [85] learn a sequence of *decision trees*, the predictions of which are combined in a final ranking score. The kind of labeled data and the performance score that is optimized also varies. *Pointwise* L2R algorithmns learn to predict as for each query-document pair the degree to which the document is relevant to the query. *Pairwise* algorithms learn to predict the correct ordering for pairs of documents given a query. *Listwise* algorithms learn to produce for each query a ranking such that a *retrieval evaluation metric* for the whole list is optimized; we will discuss evaluation metrics in the next section. The L2R models we apply in Chapters 6 and 7 are linear. Pegasos SVM [197, 202] and RankSVM [111] are pairwise models, while Coordinate ascent [151] is a listwise method. L2R technology is actively developed into many directions, e.g., online L2R [99], diversifying search results [196], multi-task learning [49], semi-supervised L2R [219], but at this point we have covered enough ground to facilitate understanding of forthcoming chapters.

In the next section, we discuss evaluation of IR algorithms, devoting quite some attention to automatic evaluation. One theme in Chapters 6 and 7 is that evaluation and optimization are related. In those chapters, we generate ground truth that can be used

for both optimization and evaluation. When evaluation is performed automatically, the distinction between the two becomes smaller, and when we discuss automatic evaluation, we will see again a L2R approach [14].

2.2 IR evaluation

In this section on IR evaluation methodology, we focus on Cranfield style, test collection based evaluation, which has become dominant over the decades; see [193] for a comprehensive and thorough overview of the history of this tradition, ranging from the earliest Cranfield experiments to today's TREC campaigns. This methodology is characterized by static sets of test queries, documents and relevance assessments, with which a collection of systems can be benchmarked in an offline fashion. Of course, there are other important and popular methods, such as user studies [235], A/B testing [123], or interleaving [180], but the work in this dissertation is primarily centered around the evaluation using IR test collections (although in Chapter 5, we perform a content analysis [131] of assessor comments on the quality of the ranking of their expertise areas). We start with a brief discussion on benchmarking and on how it has been analyzed in the literature. This will help us understand how we assess the quality of our test test collection for *expert profiling* in Chapter 5 (we will point forward to Chapter 5 repeatedly). It will also give some background to understand the automatically generated test collections in Chapters 6 and 7. After discussing benchmarking, we zoom in on the ingredients that constitute a test collection. Next, we consider some related work on error analysis: error analysis is key to gaining insight into why systems perform as they do. Finally, we devote a section to automatic evaluation, with the focus on approaches that generate "pseudo test collections" (PTCs) for subsequent benchmarking or optimization.

2.2.1 On benchmarking

To be able to create a yardstick for benchmarking, simplifying assumptions have to be made about users, their tasks, and their notion of relevance. For example, in the TREC ad hoc collections, a user is assumed to have an information need that is often assumed to be informational [38], and a document is relevant if it contains a relevant piece of information, even if it is duplicate information. By framing the expert profiling task from Chapter 5 as a ranking task, we also make some simplifying assumptions. For example, users are satisfied with a ranking of expertise areas for an expert, and an area is relevant if the expert has judged it so him- or herself. Typically, evaluation methodologies are assessed by comparing them to each other, performing detailed analyses in terms of sensitivity, stability, and robustness. Sensitivity of an evaluation methodology has been tested by comparing it to a hypothesized correct ranking of systems [101, 179]. Stability and robustness are closely related concepts. An evaluation methodology can be said to be *stable* with respect to some changing variable, or robust to changes in that variable. For example, Radlinski and Craswell [179] examine how evaluation changes when queries are subsampled. Buckley and Voorhees [41] examine changes when relevance assessments are subsampled. In the study in Chapter 5, we are interested in comparing and analyzing the outcomes of evaluating with two sets of relevance assessments—self-

selected vs. judged system-generated knowledge areas—and we consider two criteria: stability and sensitivity. To analyze stability, we identify four differences between our two sets of ground truth and ask how evaluation outcomes vary with respect to these differences. For analyzing sensitivity, we do not have a hypothesized correct or preferred ranking. Instead, we investigate how many significant differences can be detected with each set of ground truth.

When two evaluation approaches generate quantitative output for multiple systems, they can be correlated to each other. Often this is done by comparing the ordering of all pairs of systems in one ranking to the ordering of the corresponding pair in the other ranking. One often used measure is *accuracy*, the ratio of pairs for which both rankings agree, see e.g., [101, 179, 195, 225]. Another commonly used measure [41, 224] that we use in Chapters 5 and 6 is Kendall's tau [119], a rank correlation coefficient that can be used to establish if there is a monotonic relationship between two variables [204].

2.2.2 Ingredients of a test collection

A test collection based evaluation methodology consists of a document collection, a set of information needs (or test queries), a set of relevance assessments, an evaluation metric, and possibly a significance test to be able to claim that the differences found would generalize to a larger population of test queries.

Information needs Information need creation in the TREC ad hoc tracks is typically done by assessors, who aim to construct information needs that users of the search application at hand may have. Information needs for document retrieval often consist of a natural language *description* of the information need, a short formulation of that information need, sometimes called *title* (the actual query a user supposedly submits to the search engine), and, sometimes, a *narrative*, which may give further details on relevance criteria for documents. To help make sure information needs reflect user interests, query logs are a helpful source of information, e.g., when information needs were created for the TREC 2000 Web Track, they were retro-fitted around queries sampled from web query logs [227]. In vertical search applications, information needs can take on very different forms. For example, in our study on expert profiling in Chapter 5, information needs are about the knowledge a given expert has. Experts for these information needs are readily available: they are potentially all experts from the knowledge intensive organization being considered. For convenience, we also refer to information needs simply as queries, or test queries.

At the TREC ad hoc tracks, test queries with too few or too many relevant documents are sometimes rejected [94, 227]. Harman [94] reports that for all created queries a trial run on a sample of documents from the complete collection yielded between 25 (narrow query) and 100 (broad query) relevant documents. Zobel [247] notes that selecting queries based on the number of relevant documents may introduce a bias. In our experiments in Chapter 5, we retain all test queries (i.e., all experts) that have at least one relevant knowledge area.

Relevance assessments Relevance assessments are typically created by assessors. Assessors try to estimate which documents an actual user of a search engine might find

relevant or useful given an information need. Assessor agreement is one way to estimate the difficulty of this task. Voorhees [224] studies the impact of obtaining relevance assessments from different assessors on system ranking using Kendall's tau. Although different assessors may judge different documents relevant, system ranking is highly stable no matter from which assessor the assessments are taken. In Chapter 5, the different sets of relevance assessments for each expert are created by the same expert but through different means: in one case the assessor had to manually go through a list, in the other the assessor was offered suggestions. We find that system ranking may be affected by these differences.

An important aspect is the *completeness* of relevance assessments. When test collections were still small, all items in them were judged for every test query [193]. The experts who participated in our experiments have little time, however, and were simply not available to do this. A well-known method to evaluate systems without complete assessments is pooling. It was proposed in [114] as a method for building larger test collections. The idea is to pool *independent* searches using any available information and device [193]. In our study in Chapter 5, we also perform a specific kind of pooling. We use eight systems to generate expertise profiles, i.e., lists of knowledge areas characterizing the expertise of a person. The eight systems are not independent, but are all possible combinations of one of two retrieval models, one of two languages, and one of two strategies concerning the utilization of a thesaurus of knowledge areas. Unlike the methodology used at TREC [227], we do not take a fixed pooling depth for each run, perform a merge, and order results randomly for the assessor. Instead, to minimize the time required for experts to find relevant knowledge areas, we aim to produce the best possible ranking by using a combination algorithm. We also have something akin to a manual run: it consists of the knowledge areas selected by experts in the TU-*Webwijs* system before we presented them with a generated profile. We confirm in this study that this manual run contributes many unique relevant results, that is, the automatic systems fail to find a significant amount of these knowledge areas.

Zobel [247] performs a study on *pooling bias*. The concern here is that assessments are biased towards contributing runs and very different systems would receive a score that is too low. In particular, systems that are good at finding difficult documents would be penalized. For several TREC collections, he found that when relevant documents contributed by any particular run were taken out, performance of that run would only slightly decrease. In our study in Chapter 5, experts only judged those knowledge areas that the automatic systems found. We study the effect of regarding unpooled areas as non-relevant on system ranking, and find that it has hardly any impact. For this, like Buckley and Voorhees [41], we use Kendall's tau.

Evaluation metrics Given a set of relevance assessments for a query, evaluation metrics can be computed for a system ranking. Early metrics include *precision* and *recall*, and so-called precision-recall curves. Later, *mean average precision* (MAP) was proposed, a metric that computes at each rank where a relevant document occurs the precision, sums these precision values, and then divides by the total number of known relevant documents for this query. Hence, the metric takes both precision and recall into account. It is a well-understood and popular metric [193], and we use it in Chapters 5–7. One drawback of MAP is that it treats documents as either relevant or non-relevant.

Another popular metric, *normalized discounted cumulative gain* (nDCG) [109], can differentiate between different levels of relevance. We report on this metric, too, in Chapter 5. The development of evaluation metrics for information retrieval is a very active research area, and a great many evaluation metrics have been developed over the years, see e.g., [48, 156]

Significance tests In IR, significance tests are mostly used to estimate which findings about average system performance on a set of test queries will generalize to other queries from the same assumed underlying population of queries. A simple rule of thumb is that an absolute performance difference of less than five percent is not notable [216]. Pairwise significance tests are common in cases where different systems can be evaluated on the same set of test queries. Voorhees and Buckley [225] test the five percent rule of thumb with a fixed set of systems and a fixed document collection. Sanderson and Zobel [195] extend this research and consider relative rather than absolute performance differences; they prefer a pairwise t-test over the sign test and the Wilcoxon test. Smucker et al. [209] compared p-values (for the null hypothesis that pairs of TREC runs do not differ) computed with five significance tests. They find that the Fisher's pairwise randomization test, matched pairs Student's t-test, and bootstrap test all agree with each other, while the Wilcoxon and sign tests disagree with these three and with each other. They prefer Fisher's pairwise randomization test, which is what we use in Chapters 3, 4, 5, and 6. In Chapter 7, we use the ubiquitous matched pairs Student's t-test.

In our work in Chapter 5, we vary query sets and sets of relevance assessments. Then, keeping the significance test used fixed, we measure the average number of systems each system differs significantly from. We view this as a rough indication of the ability of each set of assessments to distinguish between systems. The difference in this number between sets of relevance assessments is a rough heuristic for the difference in sensitivity of the sets. Cohen [56], who is interested in repeating a benchmarking experiment using a more stringent alpha value in the significance test, computed the average number of systems each system differs from for both values of alpha. He calls the difference between these numbers the *criterion differential*, saying it is a rough heuristic for the difference in sensitivity of both alpha values.

2.2.3 Error analysis

While test collections enable us to discriminate between systems based on their average performance over a set of queries with a certain reliability and sensitivity, Harman and Buckley [95] stress that it is important to understand variance in performance over queries. Often, performance of single systems varies more over queries then performance on one query varies over systems. Variation in performance over queries does not simply correlate with the number of relevant documents; there is an interaction between query, system, and document collection [226]. Amongst other chapters, in Chapter 5, in Section 5.4.2, we perform an error analysis of expert profiling algorithms. We also find no immediate correlations at first (e.g., between query performance and number of relevant items). But when we dig a bit deeper and look at query (expert) characteristics, such as the number of documents associated with an expert, and, in particular, the *kinds* of documents, we can explain some of the variance of performance over queries (experts).

Summarizing, we have elaborated on the Cranfield evaluation methodology for IR, upon which much of our work builds. We now have the tools in hand to analyze and reason about the test collections in this dissertation, be they hand-crafted or generated automatically. The latter kind are the subject of the next section.

2.2.4 Automatic evaluation

There is a large body of literature regarding the use of interaction data, for online IR evaluation, as we mentioned before. We focus on complementary approaches, which aim to construct Cranfield style IR test collections with queries and relevance assessments, for offline evaluation and optimization. Some approaches automate only some aspect of the normal test collection creation; we will start with these. Other approaches are fully automatic. We refer to the resulting test collections as *pseudo test collections* (PTCs). Often in this line of work, system ranking produced by PTCs are compared to system rankings produced by hand-crafted test collections, like we do in Chapter 6.

As we have discussed before this section, the usual test collection creation process already involves automatic components, e.g., probing a document collection with a search engine to see if relevant documents for a candidate query exist, and *pooling* results of several algorithms before judging these results. Some research aims to enhance test collections with automated methods, while still requiring the same annotator effort. Büttcher et al. [42] counter *pooling bias*—i.e., the problem that new, previously unpooled search results from a new system will never be rewarded—by training a classifier on known relevant documents, and adding documents which are predicted to be relevant to the pool.

Other research focuses on *reducing* annotator effort, for example by sampling documents from the pool for which it would be valuable to obtain judgments: an *active learning* setup. For example, Carterette et al. [45] sample a document from the pool that would make the difference between MAP performance of systems large, if its relevance label would be known. The document is presented to an assesor, and the process is repeated until confidence in the estimated MAP value is high enough. This reduces the number of assessments needed for a topic. Carterette et al. [46] find that using the method a much larger set of topics can be created than in typical IR test collections, with less annotator effort, and with similar reliability of evaluation results. Quite differently, Rajput et al. [181] extract so called "information nuggets" from currently considered relevant documents. Then, these nuggets are used to locate and score additional potentially relevant documents, which are shown to the assessor. They show that many relevant documents are located with this method, compared to typical pooling based test collections. Instead of requiring human assessments of document relevance, Amitay et al. [8] propose to let assesors create *term relevance sets* for each query to benchmark systems with.

Work has been done also on eliminating the need for human assessments altogether. Soboroff et al. [210] select *random* documents from the pool of documents for a query and treats these documents as relevant. They rank systems with these pseudo judgments and compare system ranking with a ranking based on editorial judgments. Correlations are positive and significant, but much lower than correlations observed between different human assessors. Leaving duplicates in the pool, giving popular documents a higher chance of being sampled, improves the correlations a bit. This finding has prompted research that tried to select *good* documents from the pool and treat these as relevant. Wu

and Crestani [236] rank systems according to how well they agree with other rankers. Nuray and Can [162] first select systems with different strategies, then perform pooling, and finally compute a ranking of documents using several voting strategies. A top percentage of this document ranking is then used as pseudo-judgments. Spoerri [218] assume that different systems retrieve different sets of non-relevant documents. By randomly grouping systems they estimate for each system how many documents it found that no other systems found. If this number is high, the intuition is that these documents are likely non-relevant. Indeed, they find a negative correlation with retrieval performance. Diaz [71] predict performance of a run without using pooled results from other runs. He observes that if similar documents do not have similar retrieval scores, this correlates with poor retrieval performance. This can be interpreted as an application of the *clustering hypothesis* [108], which states that "associations between documents convey information about the relevance of documents to requests" [145]. Hauff et al. [97] apply the methods in [71, 162, 210, 218, 236] to correlate system rankings with those observed on variety of TREC test collections, and find that, surprisingly, random sampling of documents from the pool [210], leaving duplicates in the pool, performs best overall.

Most of the above approaches assume a set of information needs is available, produced by human assessors. There is also work that aims to generate information needs automatically. This reduces annotator effort still more, and much more queries may be generated in this way. That may be beneficial for estimating the generalization of system performance to new, unseen queries, if the queries are representative of the population of real queries. It may similarly improve generalization performance of L2R methods. To generate information needs, researchers often turn to the document collections over which search is being performed. The test collection creating process is turned on its head: groups of documents are the starting point to represent an information need, and these documents are treated as relevant. Query terms are obtained from text associated with these documents.

An example that illustrates nicely that domain-specific collections may have interpretable structure which can be used to create PTCs comes from patent search [140]. Here, the information need is to find prior art related to a patent application. One can now turn to granted, older patent applications, and consider the group of documents it cites. These documents would have been relevant to the patent application when it was submitted. The associated query is simply the patent application itself, with the documents it cites removed from it [88, 164]. Related work in this area is [238]. Another domain where the approach outlined above could be applied is scientific literature search, the subject of Chapter 6.

In the realm of web search, Beitzel et al. [27, 28] use the Open Directory Project (ODP), which categorized web pages, to create a PTC. They propose two kinds. The first kind constructs a PTC for known-item search. It samples a document from the ODP and assumes it is relevant to a query (obtained from the logs of a web search engine) if that query matches the title of the document exactly. The second kind targets more informational queries. It samples documents listed under a category, and assumes these documents are relevant to a query that exactly matches the title of the category. Azzopardi et al. [15] build a PTC for known-item search by first sampling a document, and then sampling query terms from that document using several term-weighting strategies: sampling terms uniformly, sampling popular terms, or sampling discriminative terms. For doc-

ument sampling their results show that sampling proportional to the number of inlinks of documents yields PTCs that better mirror evaluation outcomes obtained using hand-crafted test collections. Huurnink et al. [102] develop similar document and term sampling strategies in a cultural heritage setting, where documents consist of metadata organized in different fields. Different strategies for building PTCs are developed based on insights from how media professionals search in relation to these fields. Similar document and query sampling techniques have also been adapted for desktop search [120, 121], cultural heritage search [102]. The document and query term sampling techniques we propose in Chapters 6 and 7 are closely related to the above techniques; in particular to the idea of sampling discriminative, over-represented terms from documents.

Asadi et al. [14] create pseudo test collections for web search. They use anchor texts as candidate pseudo-queries based on previous research in which it was shown that anchor texts can be used for query reformulation [66]. The group of documents that have inlinks with a particular anchor text are treated as relevant documents for the anchor text query. Several techniques to decide which anchor texts to sample as pseudo-queries are propesed, based on the number of documents they link to and the quality of documents they link to. The quality of documents is estimated using features such as spamminess and Pagerank. The main goal of their work is to use the generated PTCs as training data for a simple learning to rank algorithm, just like we do in Chapters 6 and 7, where we create PTCs for scientific literature search and microblog search, respectively. The idea of using anchor texts for PTC generation may be applicable in scientific literature search, using texts associated with references to cited papers. However, in Chapter 6, we propose another idea that makes use of the rich metadata, e.g., keywords, present in scientific articles. In microblog search, posts are hardly linked to. To create PTCs for microblog search, we make use of hashtags, which label tweets with a topic, and we rely on a query term sampling technique to generate queries.

So far, the approaches we have always used a particular document collection as a starting point for PTC creation, although queries and relevance judgments were generated automatically. Interestingly, it is one of the earliest PTC creation efforts where a more radical approach was take. Tague [220] propose a probabilistic framework to model document collections as well as queries, retrieval algorithms, and evaluation metrics, with the purpose of better understanding the behaviour of IR algorithms.

In the previous section and this section, we have discussed general IR algorithms and evaluation, and we have already seen some examples of how such methods were adapted for domain-specific settings. In the next section we provide some background information about the domain-specific search applications we study in this dissertation.

2.3 Vertical search applications

In web search, results from different *verticals* are typically combined nowadays, e.g., text, images, video, and news [125]. In that setting, research into *aggregated search* deals with which verticals to show results from in response to a query, and where to present these results [9, 125, 159]. In this dissertation, instead, we study several vertical search applications in isolation. In the design of search technology for domain-specific search applications, it is useful to start from the ingredients shared by most IR settings

share, and see how each is instantiated in a particular domain. How are information needs represented? What are information objects? How do we think about the notion of relevance? What are the affordances of the system we are considering, and how do systems approach this task? In the following subsections we will provide background on people search (including expert search and expert profiling), scientific literature search, and microblog search.

2.3.1 Finding people

Searching for people has received a lot of attention in the 2000s. Spink et al, 2004 observe that about 4% of web search queries contains person names. About one in four of those queries contained the name of a celebrity. A 2010 report by Pew Research Center on reputation management in social media questionnaired adult American internet users and found that people search for themselves, people from their past, friends, colleagues, business associates, and so on.[3] In recent years, various services offering dedicated people search facilities have emerged, e.g., `intelius.com`, `123people.com`, `spokeo.com`, `peoplesmart.com`. In [231], we study the logs of a Dutch dedicated people search engine, and observe that about seven percent of queries contains the name of a high-profile person (e.g., a celebrity, politician, or somebody who was recently in the news). In Chapter 3, we automatically classify queries into high- and low-profile queries. Whether or not the query targets a celebrity, the query intent is a particular individual, and returning relevant results is complicated by the high ambiguity of person names. One approach to tackle this problem is to group search results around the individuals mentioned in these results. Three benchmarking activities have been organized around this task [11–13, 22], and Chapter 4 revisits it in the context of a dedicated people search engine. People search is also happening in knowledge intensive organizations, where the search is mainly driven by a need for expertise [18]. Expertise retrieval is the umbrella term for this research area. Two main tasks in expertise retrieval are expert finding and expert profiling. In expert finding, the information need is usually an area of expertise, which is to be satisfied by the names of experts in that area. In expert profiling, the aim is to capture the areas of expertise of a given expert. The latter task is the subject of Chapter 5. In the remainder of this section, we first take a look at query classification, then at search result clustering, and finally at expertise retrieval.

Query classification Query classification aims to correctly identify the underlying information need of an incoming query, with respect to a fixed classification scheme. In Chapter 1 we discussed Broder's taxonomy for web search, which distinguishes informational, navigational, and transactional information needs [38]. This typology has served as the basis for a number of query classification schemes, including those by [107, 117, 189]. For an automatic web search query classification challenge, Li et al. [134] propose 67 classes, organized by topic. In vertical search engines, similar classification schemes have been proposed. For search in an audiovisual archive, Huurnink et al. [103] match query terms with thesaurus terms. Mishne and de Rijke [153] observes that in blog search, all information needs are informational, and propose a further classification

[3] `http://pewinternet.org/Reports/2010/Reputation-Management.aspx`

scheme. As we will discuss in Chapter 3, our people search query classification can be seen as a further specialization of [153]. Jones and Diaz [113] focus on collections where documents are timestamped, e.g., in news search, and blog search. They observe three classes of temporal profiles for queries, based on the time series of timestamps of result sets for these queries: (1) *atemporal queries*, for which retrieved documents are published uniformly over time; (2) *temporally unambiguous queries*, for which a single peak can be observed in publication dates; and (3) temporally ambiguous queries, for which there are multiple peaks. Kulkarni et al. [127], studying web search queries, distinguish between several types of peaks in the number of searches issued over time, depending on how quickly user interest surges or declines before and after a peak.

When the population of queries to be classified is large, automatic classification is necessary. For this, features have to be formalized. Broadly, features can be categorized as (1) capturing user behavior as observed in transaction logs, (2) capturing coverage of the state of affairs in the real world related to the query as found in some document collection, e.g., the internet. Both can be studied as a time series, of searches, clicks, or timestamped documents associated with a query string. In news search and blog search, Jones and Diaz [113], interested in classifying queries with regard to the temporal profile of result set publication dates, compute temporal features from this time series, and a result set *content clarity* feature [63]. [107] define features that detect whether a query contains e.g., business names (indicative of navigational queries), song titles (indicative of transactional queries), and question words (indicative of informational queries); they also use some session characteristics, e.g., how many result pages a user inspected. As an aside, defining what a session is, is not trivial, and it can have an impact on the outcome of a transaction log analysis. Session boundary detection has been the main subject of several studies, e.g., [90, 91]. An often used way of gaining more infomation about a query is issuing it to large knowledge bases, e.g., Wikipedia, The Open Directory, Yahoo! categories [153], or a general purpose web search engine [134].

Given the features, algorithms that have been used for query classification include rule-based approaches [107], decision trees [113], Naive Bayes classifiers, support vector machines, and other supervised machine learning techniques [134].

Evaluation of query classification algorithms makes use of general classification performance metrics, but it is interesting to take a look at which considerations play a role in choosing one or more metrics. Query classes are typically imbalanced [107, 113], and evaluation under such circumstances has been studied in the literature, e.g., in Daskalaki et al. [67]. Jansen et al. [107] report on the number of misclassifications of their method. Care must be taken with this metric, because a classifier that would always predict the majority class would achieve a high score for this metric if classes are highly imbalanced. A common approach is to report precision and recall for each class. If a single metric to estimate the quality of a classifier is desired, usually precision and recall are combined in an F-score. Li et al. [134] organize a query classification challenge, and use the micro-averaged F-score [214]. As with accuracy, though, a method that performs well on the majority class will score high on this metric. Most query classification experiments follow this basic scheme: researchers propose a taxonomy, annotate queries with class labels, define features, perform a supervised pattern recognition experiment, and compute multiclass classification metrics. To conclude our discussion of related work on query classification, we give an example of a completely different experimental setup, for the

classification of Youtube videos by [59]. They focus on videos for which the number of views over time displays a single peak, and propose a very simple classification scheme. If a factor outside the network of Youtube users caused the peak (*exogenous*), the onset of the peak will be sudden. Then, if the video is not spread through the network with high enough probability, most of the mass ($> 80\%$) of the views will be concentrated in the peak itself (the exogenous *subcritical* class). If the video is shared enough, only between 20 and 80% of the mass will be concentrated in the peak (the exogenous critical class). If the peak was caused at least in part by users in the network sharing the video (endogenous), mass in user views will be spread before, during, and after the peak, leaving only $< 20\%$ percent of mass in the peak (the endogenous critical class). Subsequently, they propose mathematical models for each of the three classes, which predict the rate at which the number of views will decay after the peak, assuming an exponential decay. These predictions are evaluated by fitting an exponential decay to videos belonging to the three classes, and observing that, indeed, the most common best fits correspond to the predictions of the models.

Search result clustering An important characteristic of people search queries is their very high ambiguity, which is caused by the very high ambiguity of person names. One way of dealing with this problem is to disambiguate search results. Disambiguating search results can be approached by clustering the result documents. This clustering is usually aimed at organizing and categorizing results to facilitate, for example, search refinement or exploratory search. Various approaches to (web) search result clustering have been proposed. Examples include key phrase extraction [243], latent semantic indexing on result documents [148], and clustering based on document snippets [84, 87, 245]. A lot of work in the area focuses on selecting clusters to present to users (e.g., [58]) or assigning labels to the clusters (e.g., [106]); see [43] for an overview on web search results clustering.

As a special case of search result clustering, result disambiguation for person name queries deals with clustering search results per person, such that each cluster contain documents referring to a particular individual. Because a document can refer to several people with the same name, documents are allowed to appear in several clusters. The WePS campaigns [11–13] provide an evaluation framework for this particular task. Most of the best performing approaches use hierarchical agglomerative clustering (HAC). HAC is a greedy iterative clustering procedure that starts out with a clustering in which each document is assigned to a singleton cluster. Then, the algorithm chooses the pair of clusters that has the highest cluster similarity. If their similarity exceeds a fixed minimal similarity threshold, the clusters are merged and the process continues, or, alternatively, the process continues a fixed number of iterations. Differences between WePS participants mainly exist in the features that are being used, and in how the similarity between clusters is computed. Chen et al. [51] use tokens from sentences in which the query name occurs, tokens from the entire web result page, and bigrams as features in their clustering approach; each feature is weighed using tf.idf. Monz and Weerkamp [158] compare a large range of features for HAC, including windows around the query name, different tf.idf weighting schemes, and various schemes for computing cluster similarity. Ikeda et al. [104] use a two-stage clustering algorithm, in which HAC (based on named entities, compound keywords, and links) is followed by keyword extraction

from the resulting clusters to add documents to these clusters. Yoshida et al. [240] build on the system created by [104], but replace the second stage with a bootstrapping algorithm: Espresso [168]; this bootstrapping algorithm uses single words as patterns and documents as instances. When the number of clusters is known beforehand, the person name disambiguation task can be formulated as a classification problem. The work in Chapter 4 differs from previous work in that it focuses on a people search engine setting, in which social media profiles play an important role. We show that the task of disambiguation is harder when these profiles are common. We propose to treat social media profiles separately and are able to overcome this problem. In a similar approach to ours, Delgado et al. [69] refuse to compute similarity between social media profiles from the same platform (e.g., two Facebook profiles) while clustering, although in a later stage of the clustering algorithm, the two profiles could still end up in the same cluster. In their experiments, this strategy also leads to an improvement.

In the WePS campaigns, evaluation is performed by creating a gold standard clustering, and computing metrics that compare participant clusterings to the gold standard. Amigó et al. [6] propose four constraints that such metrics should satisfy, and show that B-cubed precision and recall are the only of a wide range of cluster evaluation metrics that do so. They also extend B-cubed precision and recall to handle overlapping clusters. In the second edition of the WePS benchmarking activity [12], extended B-cubed precision and recall were reported on, as well as on a weighted F-score of the two. By weighting precision and recall differently, F-scores for systems varied, and sometimes large changes in the rank of a system relative to other systems were observed. Amigó et al. [7] propose a Unanimous Improvement Ratio (UIR) which can be used to calculate the degree to which observed differences in F-score are robust with regard to how precision and recall are weighted. In Chapter 4, we report on extended B-cubed precision and recall, an F-score in which they are equally weighted (i.e., their harmonic mean), and on UIR.

Result disambiguation in people search is related to other tasks, such as named entity disambiguation [65, 93, 175] and entity linking [203]. In these two tasks, most commonly, mentions of entities are linked to known entities in a knowledge base. Gruetze et al. [89] propose an interesting connection between people search result clustering and entity linking. If we use a social media platform such as Facebook as a knowledge base of individuals, people search result clustering can be approached as a classification task, where each document must be assigned to the right social media profile.

Expertise retrieval A special case of people search is expert retrieval [18]. An excellent overview of this area can be found in [24], who broadly group information needs in expertise retrieval in two categories: (1) Who is an expert on a given knowledge area, and (2) What are knowledge areas of a given expert?. In the first scenario, an example query could be "information retrieval", and the information may be satisfied by a list of people ranked by their expertise in information retrieval. This is the *expert finding* task. In the second scenario, an example query could be "Bruce Croft", and the information need may be satisfied by a list of knowledge areas ranked by the degree to which Bruce Croft is an expert in them. This is the *expert profiling* task. In Chapter 5 we study the second scenario.

One important challenge in both scenarios is that in most document collections, ex-

perts and knowledge areas are not directly represented [24]. Rather, expert names may appear as authors of documents, or teachers in a course, for example. Expert names are ambiguous, and knowledge areas may be represented by different words in document collections. The evaluation test-bed we study in Chapter 5 mitigates these problems to some extent, because the document collection was obtained from a knowledge intensive organization, experts are known by their employee number, and many unambiguous associations between experts and documents in the collection are known. In addition, the task is defined as ranking knowledge areas from a specific taxonomy of knowledge areas. Still, there are no associations between documents and knowledge areas, so matches between a document and a knowledge area have to be estimated by the profiling algorithms.

Though documents, e.g., publications, e-mails, technical reports, course descrptions, etc., are not what an expert finding or expert profiling system has to retrieve, it is clear that such documents provide essential evidence for both tasks. Estimating expertise from such documents is a difficult problem, complicated by the heterogeneity of documents, e.g., their multilinguality, their document type (course description vs scientific article). These challenges are also presented by the document collection used for expert profiling in Chapter 5.

To support innovation in expertise retrieval, from 2005 until 2008, benchmarking activities for expertise retrieval were organized in the TREC Enterprise Track [17, 23, 60, 211]. At the time, the state of the art was represented by variants of two *generative* models [24]. In our evaluation study in Chapter 5, which was initiated in the late 2000s, we use eight variants of these generative expertise retrieval models. The first of the two generative models is expert-centric. From all documents associated with an expert, a *language model* for that expert is estimated. Using that language model, the expertise of the expert can be estimated in turn. This model became known as simply Model 1. The second generative model is document-centric. It builds a language model for each document, which can be used to estimate the association between documents and knowledge areas. By accumulating evidence over documents that a candidate expert is associated with, his or her expertise on a given knowledge area can be estimated. This document-centric model is often referred to as Model 2. Both Model 1 and Model 2 can be applied to expert finding as well as expert profiling. In Chapter 5, Section 5.2 we introduce these models formally. Other effective models models for expertise retrieval include discriminative models [82], voting models [142], and graph based models [200].

To evaluation expertise retrieval algorithms, several test collections are available. The first TREC Enterprise Track edition [60] used a collection crawled from W3C for their expert finding task. As expertise areas, topics of W3C working groups were used, and members of these groups were considered experts. This test collection has also been used in the paper that introduced the expert profiling task [19]. In the next edition of the Enterprise Track [211], participants contributed information needs and relevance assessments. Because it proved hard for outsiders to create realistic information needs for expertise in a knowledge intensive organization, in the next edition of the track [17] information needs and relevance judgments were contributed by employees of CSIRO, the Australian Commonwealth Scientific and Industrial Research Oganization [16]. In the final year of the track, information needs were mined and from an e-mail log, selecting email requests that were answered with a link to at least one CSIRO website, and making sure the set of requests have a wide coverage. The TU expert collection that we release in Chapter 5

is an update and extension of the "UvT collection" released in [20] which has previously been used for expert profiling and expert finding. In the UvT collection, as we noted in Chapter 1, relevance assessments are produced by employees from Tilburg University, who select knowledge areas from a taxonomy of expertise areas to describe their own expertise. The sparseness of these relevance judgments motivates our work in Chapter 5. For an overview including these and a number of other test collections, we refer to [24].

Automatic generation of test collections has also been done. We give a couple of examples. Seo and Croft [199] use Apple Discussions[4] forums as expertise areas and use the top ten rated answerers for each forum as experts in the ground truth. Jurczyk and Agichtein [115] consider the author ranking task, where authors have to be ranked according to the quality of their contributions. This task is related to expert finding except that authors are not ranked by quality of contributions on a specific query. They use Yahoo! Answers'[5] thumbs-up/thumbs-down votes and average number of stars received for best answers as ground truth for the author ranking task.

2.3.2 Finding papers

In Chapter 6, we propose methods for the automatic generation of ground truth in the domain of scientific literature search and digital libraries. This ground truth can then be used for optimizing and evaluating retrieval algorithms. In this chapter, we have already discussed related work for automatic generation of ground truth. Here, we will present background on scientific literature search itself, and on retrieval algorithms proposed for this task.

Scientific literature retrieval is an evolving research area with a long history. White [234] offers a rich review, focusing on the difficult task of writing a literature survey, and the exhaustive literature search that is a part of it. To support such exhaustive search, it is natural that scientific literature retrieval requires high recall in the first place. Similar demands on search engines are made in the context of patent retrieval [140], and legal research [105]. Literature surveys and meta analyses are more commnonly produced in a rigorous fashion, e.g., with criteria for inclusion and exclusion of studies, explicit mentioning of search strategies, and so on [234]. Besides writing literature surveys, researchers have other information needs, such as known-item finding [133], or exploratory search to formulate a research question [40]. Search strategies in literature search can be categorized as locating prior research by starting from references of articles you know (footnote chasing), the consultation of your colleagues or other experts, search using a controlled vocabulary, keyword search (ad-hoc search), browsing, and looking for articles that cite a work you know (searching citations) [234].

To support such strategies, scientific literature search applications should offer a rich set of features, such as navigating to prior work and later work through references and citations; highlighting key authors and navigating to other work by these authors; advanced search functionality to search controlled vocabulary terms; strong, preferably full-text ad-hoc search; the ability to save documents; recommendation of related literature; and so on. We zoom in on search using controlled vocabulary terms, and ad-hoc search.

[4] http://discussions.apple.com
[5] http://answers.yahoo.com

Controlled vocabularies are used to index document collections of scientific articles exhaustively. To give an example, the GIRT collection [122], which we use in Chapter 6, contains annotations that categorize articles by research area, topic, and the methodology used. The idea is that if an information need can be formulated in terms of the controlled vocabularies, then all relevant documents can be retrieved. The annotation effort to index entire document collection is enormous, and often performed by professional indexers [234]. Controlled vocabularies can also be used to improve retrieval performance for ad-hoc keyword queries, e.g., by associating natural language terms from titles of articles with the controlled vocabulary terms the articles are labeled with. Then, for incoming queries, strongly assocatiated controlled vocabulary terms can be used for query expansion [172]. Meij and de Rijke [149] propose a similar idea, first translating a query to annotations, and then sampling distinguishing terms from these annotations to expand the query.

One benchmarking activity for ad-hoc, domain-specific search in scientific articles is the CLEF Domain Specific Track, the last two editions of which were organized in 2007 [174] and 2008 [173]. We perform experiments on the two IR test-collections used in these two tracks in Chapter 6, focusing on the monolingual (English) retrieval task. In the 2007 edition, participants experimented with different retrieval models, query expansion, and data fusion, combining scores of several retrieval runs in a single score. The best performing group fused language modeling and divergence from randomness (DFR) retrieval runs. Their query expansion using controlled vocabulary terms did not lead to improvements [83]. Their approach is very similar to our approach in Chapter 6, where we combine scores for similar retrieval models using learning to rank (L2R). Performance scores between our runs and theirs cannot be compared directly, however, because Fautsch et al. [83] use both the title and the description fields of the information needs, whereas we opted to use only the title field, which represents the query that a user submits to a search engine. Participants in the 2008 edition of the CLEF Domain Specific track experimented with different stemming methods, retrieval models, query expansion, and data fusion. The best performing system fused runs for which different stemming algorithms had been used, used standard pseudo relevance feedback (PRF), and experimented with query expansion using controlled vocabulary terms, although the latter did not help performance [110]. Meij and de Rijke [149] do observe improvements when expanding queries using controlled vocabulary annotations (see above). In Chapter 6, we confirm that standard PRF is very strong, with our Terrier divergence from randomness-based PRF run.

In Chapter 6, we propose document features, to be used in a L2R setup. Largely, these features concern the authors of papers. This highlights an interesting connection between scientific literature search and expertise retrieval. In expertise retrieval, it is recognized that locating good documents is essential to aggregate evidence of expertise. For scientific document retrieval, author-based features reflect the belief that expertise retrieval can help document retrieval. While we did not convincingly demonstrate a positive effect of the document features in our experiments, we still believe it is a promising direction. One way to improve our author-based features would be to base them on properly disambiguated surface forms of author names, using techniques from research on author name disambiguation [205], and people search result disambiguation (Chapter 4).

2.3.3 Finding posts

In Chapter 7 we extend our pseudo test collection (PTC) generation ideas and apply them in the domain of microblog search. This section discuss work related to finding posts; in particular, *microblog* posts.

Microblog search is a relatively young research area, naturally, since microblogging platforms have only recently emerged, the largest exponent being Twitter. Hundreds of millions of people use microblogging platforms to post so-called *status updates*, short messages consisting of a limited number of characters (140, in Twitter), on topics ranging from personal (e.g., "pointless babble" [118], cited from [76], to political, to instantaneaus reports of current events, e.g., earthquakes [192]. In addition to offering people the opportunity to blog, microblogging platforms are social networks. Authors have followers, and "followees": other authors they follow in turn. On Twitter, tweets can be *re-tweeted* with little effort, which relays messages to followers of the retweeter. It is also possible to address other authors explicitly by their username in a tweet, and to reply to a post. To indicate that microblog posts are part of a larger discussion, *hashtags* can be inserted in a post. Another important aspect of microblog posts is that they contain links (usually in a shortened form) to web pages and content, which can provide rich context regarding the topic of the tweet. The spread of information in microblogging networks and the influence that authors exert as a consequence of this has attracted much attention [47, 129, 233].

Microblog *search* is essential, because of the sheer volume of information on microblogging platforms. Twitter search, for example, handled over 1.6B queries a day by 2011.[6] The real-time and personal nature of microblog posts is reflected in the information needs of searchers, who look for recent content pertaining to events, breaking news, and trends, and for information about people [221]. Also, repeatet queries are observed, where people appear to monitor search results [221].

Determining the relevance or usefulness of microblog posts with regard to a query poses new problems and challenges [76, 221]. Massoudi et al. [146] report on an early study of retrieval in microblogs, and introduce a retrieval and query expansion method to account for the fact that microblog posts are short, which increases the risk that there is a *vocabulary mismatch* between a query and a post. Naveed et al. [161] show that because posts tend to have the same length, *document length normalization* is not helpful for retrieval. In addition, they recognize that estimating the probability that a tweet is retweeted can boost retrieval performance, in particular for underspecified queries. Efron and Golovchinsky [77] investigate the temporal aspects of documents on query expansion using pseudo relevance feedback. Efron [75] recognizes the significance of hashtags and proposes to retrieve hashtags which are often used to label tweets that are relevant to a query.

In 2011, TREC launched the microblog search track, as a part of which an ad-hoc search task was organized. For this task, systems are asked to return relevant and interesting tweets given a query issued at a particular time [166]. The organizers required systems to retrieve a set of relevant results and then rank them in reverse chronological order. The main evaluation metric used was P30, and many succesful systems ranked by relevance first, cut off the ranking after the 30st result, and only then ranked tweets

[6] `https://en.wikipedia.org/wiki/Twitter`

by their publication time. This constitutes evidence that ranking by relevance is more useful for microblog search. In the 2012 edition of the TREC task, organizers required an ordering by relevance, and this is the evaluation setup we use in Chapter 7.

In the 2011 edition of the TREC microblog search track, the temporal aspect of Twitter and its characteristics, e.g., hashtags and existence of hyperlinks, were exploited by many participants, for query expansion, filtering, or learning to rank [166]. For learning to rank, in TREC 2011, teams depended on self-constructed, manually labeled training data to train their model on. The lack of training data that necessitated these manual annotation efforts provides a strong motivation for our work in Chapter 7, where we generate training data (pseudo test collections) automatically and are able to achieve a very competitive performance (top two, for P30) by training our L2R methods on these PTCs. Particpant systems in TREC microblog 2011 were subdivided according to whether or not they used external resources (e.g., by following outlinks and obtaining terms from linked-to web pages), and according to whether or not they used future evidence (e.g., by computing IDF scores for tweets published later than a query timestamp). At least one run without using external sources and future data was required. Our methods in Chapter 7 use no external sources, and we build separate indexes for every query to avoid using future data.[7] In the 2012 [212], 2013 [137] and 2014 editions of the TREC microblog search track, participants started to use external sources heavily, and this led to large improvements in retrieval performance.

As effective approaches for microblog retrieval, query expansion, temporal information retrieval, document expansion, and learning to rank stand out. Even without using external resources, large improvements have been observed for query expansion [52, 80, 244, 246]. Further improvements have been obtained by using *context* outside of the status updates themselves. For example, by doing query expansion against other collections [25, 72, 78, 79, 154, 230, 237, 244]. Temporal retrieval information is developing into a field on its own [160], and a score of work has investigated temporal aspects of microblog retrieval, particularly for query expansion [50, 52, 77, 116, 128, 146, 155, 169]. Document expansion approaches using external sources, in particular by following outlinks from tweets and using web content terms to expand tweet representations, yield improvements as well [135, 141, 147, 237, 246]. Many of the abovementioned studies use simple linear L2R approaches such as the ones we use in Chapter 7 to combine their features and obtain improvements. Some more complex models have been shown to perform well, e.g., data fusion methods [136], ranking factorization machines [177], ensembles of L2R systems [79], and support vector machines with kernels that take syntactic similarities between queries and tweets into account [201].

[7] The use of future data was later found not to have a noticeable effect on performance scores [228].

3
Query Classification in People Search

In this chapter we address RQ1, which is concerned with automatic classification of information needs in the people search domain. People search is an important aspect of human search behavior. E.g., in web search an estimated 3.6–4% of the queries contain person names [217]. People search for themselves, people from their past, friends, colleagues, business associates, etc.[1] We examine queries submitted to a people search engine and are particularly interested in what types of queries are submitted. That is, what types of people are searchers looking for? This increases our understanding of user behaviour. We then describe features for automatically classifying people into these types. This is useful for a number of reasons. A search engine can return different kinds of results or apply a different ranking algorithm depending on the predicted category of an incoming query. Different types of query may also give rise to different ways of presenting results.

In [231], we analyze the logs of a Dutch people search engine and do indeed observe that most queries are entered only infrequently and target unknown people: possibly friends, colleagues, and so on. We refer to such queries as *low-profile* queries. We also identify two types of *high-profile* people queries. Here, we rephrase the definitions of the two classes, not intending to change their meaning, but merely to complement them.

1. *event-based*: such a person is well-known and is likely being searched for because she was recently in the news or involved in a recent event or hype, and

2. *regular*: such a person is well-known because she is a celebrity, politician, etc., and most likely not queried because of any particular event, but rather because of the accumulation of events that made her well-known.

We emphasize the uncertainty that is always associated with interpreting query logs: we do not know the reasons people had when entering this query or even if an entered person name corresponds to the name of, e.g., a celebrity or politican only by chance. Nevertheless, the two categories have a clear presence in the abovementioned logs. Many examples of event-based query targets can be observed, including murder victims, suspects, and so on. After their names have been published there is a sudden and huge peak in the frequency of searches for them. There are also people who continuously attract significant attention from searchers; they are clear cases of regular high-profile targets.

[1] `http://pewinternet.org/Reports/2010/Reputation-Management.aspx`

We do not assume that the type of a query is fixed over time, e.g., a soldier who died in Afghanistan may have been low-profile before he died, but may become event-based high profile afterwards. The three class definitions pertain to *query instances*: queries entered by a particular user at a particular time.

With regards to Broder's influential taxonomy of web search queries [38], it is reasonable to assume that most queries in our logs are informational, as in blog search [153]; we confirm this in [231]. Mishne and de Rijke [153] propose context queries ("locate contexts in which a name appears") and content queries ("locate blogs or blog posts that deal with the searcher's interest areas"). With respect to this taxonomy, our queries are all context queries: tracking references to the target person; thus, the above taxonomy refines the one in [153]. From the definition of low- and high-profile queries, it is clear that the user interest in the person name over time should play a role in determining the class of a person name query. As discussed in Chapter 2, Section 2.3.1, Jones and Diaz [113] classify queries into three temporal classes according to the distribution of publication dates of result sets. There is a resemblance between our taxonomy and theirs. In particular, the temporally unambiguous queries observed by [113], where a single peak can be observed in a time series of publication dates, intuitively correspond to our event-based high-profile queries.

A general motivation for query classification was provided at the start of this section. Concerning person name queries, if a people search engine can establish with reasonable accuracy the most likely class of an incoming query instance, it may use this in various ways. E.g., for low-profile queries, the result page could include results from social media, contact information and images. For event-based high-profile queries, it could include information about relevant news stories, that may be presented on a time line. For regular high-profile queries, there may be many news stories to be found, as well as many images, video clips, social media pages, etc. Here, a sensible strategy is for the result list to include a diverse set of material about the target so as to facilitate exploratory search. Because person names are highly ambiguous (see Chapter 4), it may be that a user is looking for a non-famous person sharing the name of a celebrity; in such cases, the search engine may want to adapt its strategy so as to avoid result pages from being dominated by hits relating to the famous person.

In [231], we propose a limited set of features for people search query classification, and perform two classification experiments. The results suggest that classification into low- and high-profile queries is feasible, with high precision and recall scores for both classes. A three-way classification that also requires the distinction between event-based and regular high-profile queries is harder, with lower precision and recall scores for both high-profile classes. We redo these experiments. The focus is not on improving upon our previous performance scores, but rather on increasing our understanding of the challenges in this relatively new classification task. We perform annotation again, in an annotation interface that gives the annotator more information, and report annotator agreement. We propose several new features. We slightly modify the experimental setup. In [231], we reported test performance scores on a dataset in which the more frequent classes were downsampled. Here, we report test performance on a test set in which classes are as imbalanced as in the entire dataset, using a stratified cross validation setup. In a realistic setting, we would also expect class proportions in unseen data points to mirror class proportions in a training set. Finally, and most importantly, we perform a

feature analysis to shed light on which factors play a role in classification decisions.

We address the following research question from the introduction:

RQ1 Is it possible to classify person name queries submitted to a people search engine as either low-profile, event-based and regular high-profile classes such that reasonable agreement with human classifying decisions is achieved? Among a set of features capturing online context of person names and interactions of people search engine users, which features have the largest influence on classification decisions?

In Section 3.1 we present our data and methods, in Section 3.2 we present our results and discuss them, and in Section 3.3 we provide conclusions.

3.1 Data and methods

To inform our study, we use four months of query logs of a Dutch people search engine, collected during September–December, 2010. In [231], we describe the search engine and the logs, but we repeat the essentials here.

A people search engine The default search box provided by the engine is a single free text box in which the user is suggested to enter a first and a last name, see Figure 3.1. After entering a query and performing a search, the user is presented with search results

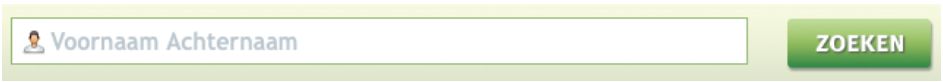


Figure 3.1: Simple search interface: a single search box with a search button.

from social media platforms, e.g., Hyves[2], Facebook, LinkedIn, and so on. Results are grouped per platform. Users can expand results for a platform to get a listing of profiles. Expanding again shows more information from a single profile, and an outlink to the profile itself. If this outlink is followed, a clickout is registered in the query logs. Via tabs at the top of the page, users can navigate to other kinds of search results: (1) documents obtained from general purpose web search engines, (2) multimedia content, and (3) miscellaneous content [231].

Query logs The query logs contain queries and clickouts. Query entries consist of a first name, last name, an optional keyword, a timestamp and an associated unique ID of a persistent cookie. Even though a cookie need not correspond one to one with a person, we interpret it as a unique visitor, i.e., a user. Clickouts consist of a URL, a top level domain (TLD), a timestamp and a persistent cookie ID. We define sessions as consecutive query entries with the same cookie, and a maximum time interval of forty minutes between each consecutive query pair, following [231]. In the three month data collection period, we

[2] A Dutch social networking site, which no longer exists. At the time of this study, its results were clicked most of all social media platforms.

observed about 13 million queries. These queries: (1) amount to 4 million unique query strings; (2) are issued by almost 7 million users, of which 1.5 million users issued more than one query; and (3) result in 4 million clickouts, on a subset of 2.4 million queries (17.6% of all queries). Of all the clickouts, the majority landed on social media profiles (66%), followed by web search engine results (17%), miscellaneous content (8.5%), and multimedia content (3.1%). 4.7% of clickouts landed on advertisements. For further details, we refer the reader to [231].

Query selection For this study, we are interested in queries that are issued by at least twenty different users over time from September 1st until December the 31st. The main reason for this is discarding a large pool of queries that are likely to be low-profile anyway. If we are interested in classifying these queries, an approach without interaction data features may be indicated. The queries we keep with this filtering step are queries for which the user population has shown some interest, and the challenge becomes to detect what kind of interest that is. Note that the twenty-user requirement does not imply that there are no low profile query instances. First, if the query instance is one of the first searches for this query, it may well be annotated as low profile: we noted already that the class of a query may change over time. Also, names are ambiguous, and searches for various low profile persons with the same name may add up to above the threshold. From these queries, then, we want to classify instances submitted from October 1st onwards, to give us, for each query, at least a month of history to extract features from. From this population of query instances, we sampled randomly for annotation.

Annotation For a query selected for annotation, in an annotation interface, annotators could see the query itself, search and news volume up to the date of the query instance, previous queries from the same session, and the values of the classification features we describe below. Finally, they were allowed to use a web search engine. Annotators were asked to determine if a query likely targeted a high- or a low-profile person at the time of the query instance. In case an annotator labeled a target as high-profile, we also required a decision whether the query instance was event-based high-profile or regular high-profile. This decision can be subtle; we illustrate this with an example.

In Figure 3.2 we plot the search volume and the number of mentions in RSS feeds of national newspapers of two high-profile targets, highlighting the difference between a 'regular' and an 'event-based' instance. On the left-hand side, we show the graph of a controversial politician frequently mentioned in the news (Geert Wilders). The query instance is most likely not related to a particular event, but rather to the sum of the events that made him well-known, it is a regular high profile instance. On the right-hand side we display the search volume and number of mentions in RSS feeds of an actress who is much less mentioned in the news; however, when she is mentioned in the news (because of the tragic passing of her husband), this is followed by a very clear peak in search volume. This instance is a quite clear example of an event-based high-profile query. The subtlety of the decision lies in the fact that she was already a well-known person before the news event that led to a boost in attention occurred. A query instance with a timestamp before this event would most likely have been classified as a regular high-profile query instance, despite the low search volume.

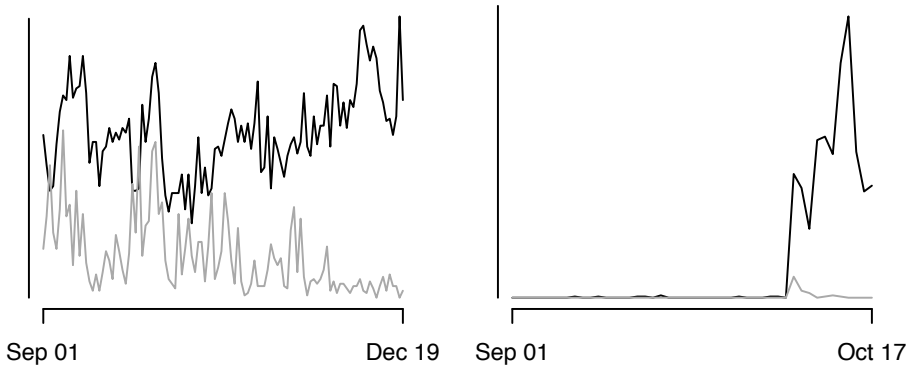


Figure 3.2: Search volume (black lines) and mentions in RSS feeds of national newspapers (grey lines) of a regular high profile (left) and an event based high-profile (right) query instance.

In total, 216 people query instances were manually labeled, 200 of which were doubly annotated. Inter-annotator agreement on the 200 doubly annotated queries was 0.70 (Cohen's kappa). Conflicting annotations were resolved through discussion. Of the 216 instances annotated, 132 were found to be low profile, 60 event-based high profile, 24 regular high-profile. The relatively low number of regular high profile queries may be due to people preferring to search for celebrities directly in general purpose web engines. It seems likely that people will resort to a dedicated vertical, i.e., a specialized people search engine, predominantly when they are unsatisfied with the results of a general purpose web search engine.

Features For automatic query classification, we use the features listed in Table 3.1. There are 6 types of features, with 15 features in total: (1) search volume, (2) clickout volume, (3) burstiness, (4) clickout entropy, (5) Wikipedia presence, and (6) news volume. The first four of these types reflect user behaviour associated with a person name, as observed in the query logs. The last two types capture coverage of real-world events and state-of-affairs associated with person names. The search volume, clickout volume, burstiness, and clickout entropy features are computed from the shape of search volume and clickout time series, similar to the features used in [113]. The Wikipedia presence features are static over time. The clickout volume, burstiness, and clickout entropy feature types are additional features with regard to the classification experiment in [231]. Below, we describe all features, explaining the intuitions behind them.

Search volume for a query is measured in unique *daily* visitors issuing the same query string.[3] Since query instances can have a timestamp ranging from October, 1st until December, 31st, the number of days during which search volume amasses ranges from 30 to 90 days. Therefore, we divide search volume by the number of days between

[3]Thus, if the same user issues a query string on multiple days, each day is counted once.

Table 3.1: Features grouped by type. There are 6 feature types, and 15 features in total.

Name (abbr)	**Description**
1. *Search volume*	*Unique daily visitors*
– From Sep, 1st (**SVFS**)	From Sep 1st until query instance, on average
– Last week (**SV7D**)	Week before query instance, on average
– Trend (**SVT**)	Difference between the previous two: **SV7D** - **SVFS**
2. *Clickout volume*	*Clickouts per day*
– From Sep, 1st (**CVFS**)	From Sep 1st until query instance, on average
– Last week (**CV7D**)	Week before query instance, on average
– Trend (**CVT**)	Difference between the previous two: **CV7D** - **CVFS**
3. *Burstiness*	*A burst is defined here as a period of one or more consecutive days on which the unique daily visitors per day issuing the query is more than the mean plus two times the standard deviation.*
– Number of bursts (**NB**)	
– Ratio search volume in bursts and total search volume (**BV/SV**)	
– One divided by the number of days since last burst (**1/DsLB**)	
4. *Clickout entropy*	*The amount of uncertainty associated with clicks on TLDs of URLs in the result set, see Equation 3.1.*
5. *Wikipedia presence*	*Based on Dutch Wikipedia dump dated August 26, 2010*
– Title match (**WPTM**)	Query person name matches title Wikipedia page (yes or no)
– Frequency (**WPF**)	Frequency of occurrence of person name in Wikipedia
6. *News volume*	*Number of mentions in RSS feeds of national news papers, per day*
– From Sep, 1st (**NVFS**)	From Sep 1st until this query instance, on average
– Last week (**NV7D**)	Week before query instance, on average
– Trend (**NVT**)	Difference between the previous two: **NV7D** - **NVFS**

September, 1st and the date of the query instance. This yields the first feature: **SVFS** (an acronym for "search volume from september"). We define two more search volume features, to capture the trend in search volume. The intuition behind this is that for event-based queries, most searches should occur frequently after the event of interest. If that is true, an event-based query is likely to follow many other, recent queries for the same person name. The second search volume feature therefore computes the average unique daily vistors entering the person name *over the last seven days* (**SV7D**). The third search volume feature captures the *trend* in volume (**SVT**); it is computed as **SV7D** - **SVFS**. This trend feature is new with regards to the experiments we performed in [231]. A positive value for **SVT** indicates an increased interest in the person name in the last week, compared to the whole period from September, 1st.

Clickout features are interesting because they reflect strong interest in a search result. More so in this people search engine than in general purpose web search engines, because in the former, users spend more effort in producing a clickout. Clickout volume is computed in a similar fashion as search volume. It is defined as the total number of clickouts following previous queries for the same person name as the current query instance. It is also normalized by the number of days in the period considered (either from September, 1st (**CVFS**), or the last week (**CV7D**). The trend (**CVT**) is computed in the same way (**CV7D** - **CVFS**).

Burstiness captures the degree to which interest in a person name peaks in the query logs. A burst is defined here as a period of one or more consecutive days on which the number of unique daily visitors exceeds the mean by two times the standard deviation. Intuitively, peaks in user interest may correspond to events in the real world. The number of bursts (**NB**) is computed. In addition, the fraction of search volume that occurs in bursts (**BV/SV**) should be high if user interest peaks strongly. Finally, for event-based queries, we expect the last burst to be recent, and we define a feature as: 1 divided by the number of days since the last burst (**1/DsLB**).

Clickout entropy measures the uncertainty associated with clicks on the result set; [127] use it in their descriptive analysis of web search queries, instantiating the well known information theoretic entropy formula as follows:

$$-\sum_{d \in D} P(c_d) * \log_2 P(c_d), \tag{3.1}$$

where $P(c_d)$ is the probability of a click landing on $d \in D$, and D is the set of URLs. Because in our data set, clickouts are sparse, we aggregate clickouts on the level of top-level domains (TLDs), setting D to be set of TLDs of the URLs in the result set. If users click on many different TLDs, and each of these events are rare, the uncertainty (or entropy) is high. If users click on a small number of TLDs only, and these events are less rare, uncertainty is smaller.

Wikipedia presence of a person name is an indication that this person name is of interest to the general public, not necessarily because of any single particular event, and should intuitively help us to detect regular high-profile queries. A binary feature, Wikipedia title match (**WPTM**), and the frequency of person name occurence in Wikipedia pages (**WPF**) are computed. These two features replace the Wikipedia result set size count from [231], which was obtained by issuing the person name query to Yahoo! while

restricting Yahoo! to Wikipedia results. News volume, finally, is interesting to capture, because news covers events happening in the real world. We expect both regular high-profile and event-based person name queries to appear in news. News volume is computed based on a time series: the number of mentions of a person name in a collection of RSS news feeds of national news papers, per day. Three features are computed, in the same way as the corresponding search and clickout volume features: news volume since September, 1st (**NVFS**), in the last seven days (**NV7D**), and the difference between the latter and the former (**NVT**).

Classification We use three different classifiers. First, a non-linear decision tree classifier, specifically, the C4.5 decision tree classifier [178]. It was used also by [113], who classify news and blog queries. Second, a Naive Bayes (NB) classifier. This classifier estimates class conditional distributions for the training points, and checks which class most likely generated unseen test points. Third, we use a support vector machine (SVM) [57] to classify the instances. This is a *discriminative* approach. SVMs do not model the data, but only separate training points from different classes by a margin as wide as possible. Using three different classifiers on this relatively unstudied people search query classification task is useful to get an initial idea for which approach works best out-of-the-box. We use the implementations available in the Weka toolkit [92], with default hyperparameters. For the C4.5 decision tree, this means we set the confidence threshold for pruning to 0.25, and the minimum instances per leaf to 2. For the SVM classifier, we set the complexity constant to 1, normalized features, and used a linear kernel. For the three-way classification experiment, the SVM classifier uses a one vs. one scheme, where a binary classifier is learned for all pairs of classes, and the majority vote decides. Default hyperparameter settings often reflect experience on a wide range of datasets, and are perhaps the best uninformed bet when approaching a new task. It is probably possible to improve performance for the classifiers we use by optimizing their hyperparameters, e.g., choosing a different kernel for SVM, or choosing a different complexity constant. We leave this to future work.

Evaluation For each class, we report on precision (P), recall (R), and their harmonic mean (F1-score). We also report the *macro-averaged* precison (P_M) and recall (R_M) over classes. These are defined as the average of the precision (or recall) scores for the individual classes [214]. We also report the *macro-averaged* F1-score (F_M), which we compute as the harmonic mean of P_M and R_M:

$$F_M = 2\frac{P_M R_M}{P_M + R_M}, \tag{3.2}$$

following [214].[4] At the KDD 2005 cup on query classification [134], the F1-score of *micro-averaged* precision (P_μ) and recall (R_μ) was used instead to rank participant systems. We opted against using micro-averaging because we wanted to weight precision and recall for each class equally, regardless of how many instances a class has. In our dataset, the low-profile query class is most common. A method that would find many true positives for the low-profile class, while performing bad on other classes, could still

[4] An alternative macro-averaged F1-score is the average of the per-class F1-scores; both variants are used.

achieve high micro-averaged precision and recall scores. This is not true for macro-averaged scores.

We use ten-fold cross-validation to test the predictive power of our classifiers. Because some of our classes have few instances, to make sure each class is represented in each test fold, we apply *stratified* cross validation. This entails that for each class, we aim to have about the same proportion of instances in all folds. We divide the data in ten stratified folds randomly, once. Then, to test a classifier, on each test fold, predictions are generated by a model which is trained on the remaining folds. The test predictions for all folds are then concatenated, and precision and recall scores are computed on the entire set.

To test if an observed performance difference for a metric between two systems are significant, we use a pairwise randomization test. A nice exposition of the this test in the context of a classification task can be found in [239]; we use the two-tailed variant. We report significance for two α levels, a strict level ($\alpha = 0.001$), denoted by $^{\blacktriangledown}$, and a lenient level ($\alpha = 0.05$), denoted by $^{\triangledown}$.

3.2 Results and discussion

We report on two experiments: (1) a two-way experiment in which we aim to automatically distinguish between high-profile and low-profile people queries; and (2) a three-way experiment in which we aim to distinguish between event-based high-profile, regular high-profile and low-profile queries. The results of both classification experiments are given in Table 3.2.

Table 3.2: Results of two stratified ten fold cross validation experiments: (1) low-profile vs. high-profile classification, and (2) classification of queries as low-profile, regular high-profile or event-based high profile. The data is divided into ten stratified folds randomly. For each algorithm (C4.5, NB, and SVM), on each test fold, predictions are generated by models trained on the remaining folds. The test predictions are then concatenated, and precision and recall scores are computed on the entire set. For both classification tasks, for each metric, highest scores are listed in boldface. The classifier with the best F_M score on both tasks is C4.5. For the NB and SVM classifiers, for each metric, significant performance differences with regard to C4.5 are marked with $^{\blacktriangledown}$or $^{\blacktriangle}$($\alpha = 0.001$); or with $^{\triangledown}$or $^{\triangle}$($\alpha = 0.05$). A two-tailed pairwise randomization test was used.

	C4.5			NB			SVM		
Query type	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
High-profile (84)	0.85	**0.82**	**0.84**	**0.89**	0.64$^{\blacktriangledown}$	0.74$^{\triangledown}$	0.88	0.60$^{\blacktriangledown}$	0.71$^{\blacktriangledown}$
Low-profile (132)	**0.89**	0.91	**0.90**	0.81$^{\blacktriangledown}$	**0.95**	0.87	0.79$^{\blacktriangledown}$	**0.95**	0.86$^{\triangledown}$
P_M, R_M, and F_M	**0.87**	**0.87**	**0.87**	0.85	0.79$^{\triangledown}$	0.82	0.83	0.77$^{\blacktriangledown}$	0.80$^{\triangledown}$
Event-based (60)	0.83	**0.87**	**0.85**	0.74	0.62$^{\triangledown}$	0.67$^{\triangledown}$	**0.85**	0.55$^{\blacktriangledown}$	0.67$^{\triangledown}$
Regular (24)	**0.57**	**0.54**	**0.55**	0.53	0.33	0.41	0.45	0.38	0.41
Low-profile (132)	**0.92**	0.90	**0.91**	0.81$^{\blacktriangledown}$	0.92	0.86$^{\triangledown}$	0.80$^{\blacktriangledown}$	**0.95**$^{\triangle}$	0.87
P_M, R_M, and F_M	**0.77**	**0.77**	**0.77**	0.69	0.62$^{\triangledown}$	0.66$^{\triangledown}$	0.70	0.63$^{\triangledown}$	0.66$^{\triangledown}$

After discussing the outcomes of the two experiments we will analyze the results of the C4.5 decision tree in more detail because (1) it is the best overall performing classifier in our experiments, and (2) because it produces models that are easily interpretable.

In our setting, decision tree classifiers like C4.5 perform well because they can combine nominal and ratio features and they handle dependencies in features well. Our features are somewhat redundant and depend on each other, e.g., if the average number of unique visitors per day that entered a given query since September the 1st is high, the average over the week before the query is more likely to be high. Since Naive Bayes assumes class conditional independence of features, this may explain why it performs a bit worse.

3.2.1 High profile versus low profile classification

We first examine the outcomes of the two-way classification experiment; see the top half of Table 3.2. Clearly, it is feasible to classify query instances into the high- and low-profile classes with a C4.5 decision tree classifier. Recall of the *high-profile* instances is significantly worse with Naive Bayes and with an SVM ($^{\blacktriangledown}$). At the cost of this drop in recall, both Naive Bayes and SVM do achieve higher precision for the high-profile instances, but these differences are not significant. Precision of the *low-profile* predictions is significantly worse for Naive Bayes and SVM ($^{\blacktriangledown}$). Both Naive Bayes and SVM improve recall of the low-profile class with regard to the decision tree classifier, but these differences are not significant. Looking at our summary metric F_M (macro-averaged F-score), only the difference between the decision tree (0.87) and the SVM classifier (0.80) is significant ($^{\triangledown}$).

In Figure 3.3 we show a partial decision tree. This tree is learned on the entire dataset. On each node the training samples are split on the indicated feature, see Table 3.1. Each edge shows the threshold value on which it is split. The leaf nodes indicate the class that the tree will predict for new examples that satisfy the requirements to reach the node. "H" and "L" represent the high- and low-profile classes. In brackets the number of queries within that class is listed. If training examples are misclassified their number is reported after a slash. Some leaf nodes contain a feature and a number of classes. Here, the decision tree visualization was truncated to save space; the feature listed will yield the next splitting criterion; in brackets it is shown how many data points were classified as high-profile ("H") and low-profile ("L"), respectively, in the remainder of the tree. As before, the number of miclassifications for each class is reported after a slash.

The most important feature is the average number of clicks per day over the last week. A surprisingly low number of clicks is sufficient to classify as many as 32 queries as high-profile queries. One explanation for this is that clicks in the people search engine we study require substantial effort on the part of the user. Recall that search results are displayed grouped social media platform or search engine. If a user wants to find, e.g., a social media profile, she has to expand the results for the social media platform of choice, then expand a text snippet (a short description of a particular search result), before finally an outlink may be followed. This explains why there are not many clickouts in the query log files. A few clicks may well have resulted from many searches.

The second feature used is the number of bursts. This feature uses the search volume history. If there are one or more bursts, then the query is high profile. In the absence of

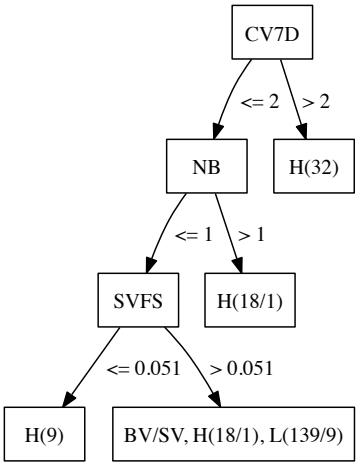


Figure 3.3: Partial decision tree for the two-way classification experiment. Nodes are labeled with the feature based on which the data is split, according to the conditions listed next to the outgoing edges. In leaf nodes, the predicted class of data points in that node are indicated: H for high-profile, L for low-profile. The numbers of correct predictions are given between brackets, followed by a forward slash and the number of incorrect predictions, if any. We truncated the tree to save space: the node at the right bottom will split on the feature **BV/SV** next, and yield predictions for both classes in later leaf nodes.

bursts, the third split is counterintuitive. The average unique number of unique visitors on which the remaining set is split seems very low. Even so, queries that were issued by even fewer people are all high profile queries in this dataset. And the bulk of the instances with a higher search volume is low-profile. This is surprising because we defined high profile persons as well-known people, either because of some recent event (event-based) or because they are a public figure, celebrity, or generally much sought after.

The news volume features do not appear at all in the decision tree for the two-way experiment. We will see that they do play a role in the three way experiment, however.

3.2.2 Low profile, event-based, and regular high-profile classification

We now turn to the three-way classification experiment; see the bottom half of Table 3.2. Three-way classification into event-based high-profile ("H"), regular high-profile ("R"), and low-profile ("L") is harder than two-way classification. For the C4.5 decision tree, performance on the low-profile and event-based high-profile is strong, but precision and recall for regular high-profile needs improvement. Results for this category suffer from the fact that there are only 24 regular high-profile instances in the data set. Looking at the Naive Bayes and SVM results, we see similar patterns as in the two-way classification experiment. Recall for the *event-based high-profile* class is significantly lower for Naive Bayes ($^{\triangledown}$) and SVM ($^{\blacktriangledown}$). For the *regular high-profile* class, it is mainly recall where performance scores of Naive Bayes and SVM drop substantially compared to the decision tree, but here we observe no significant differences. A possible explanation for the lack of significance here is that there are only 24 regular high-profile instances. Precision of the *low-profile* instances is significantly lower for both Naive Bayes and SVM ($^{\blacktriangledown}$). SVM does achieve a significant improvement of recall of low-profile instances over the decision tree classifier, however ($^{\triangle}$). Looking at the summary metric F_M, we see that the difference between the decision tree and both the Naive Bayes and SVM classifiers is significant ($^{\triangledown}$).

We can learn the contribution of individual features from the learned decision tree on the entire dataset in Figure 3.4. The first feature is again the average number of clicks per day over the last week before the query. But this time if it is higher than 2.0 the news volume comes into play. It seems counterintuitive that a low average number of mentions in the news per day over the last week leads the classifier to the conclusion that the query instance is event based. However: a few mentions in the news are often enough to cause a large interest in the person. If somebody passes away, this may be followed by a peak in search volume in the people search log even if it is hardly mentioned in the news. Many mentions in the news can be a sign that a person is famous but not well-known because of a particular event.

Again, the number of bursts is an important feature. In the absence of bursts, we find many low profile queries. Again, there is the curious exception of searches that also have a low average search volume until the date of the current instance: these are all event based queries. If there are bursts we see again that regular high profile queries have a higher search volume.

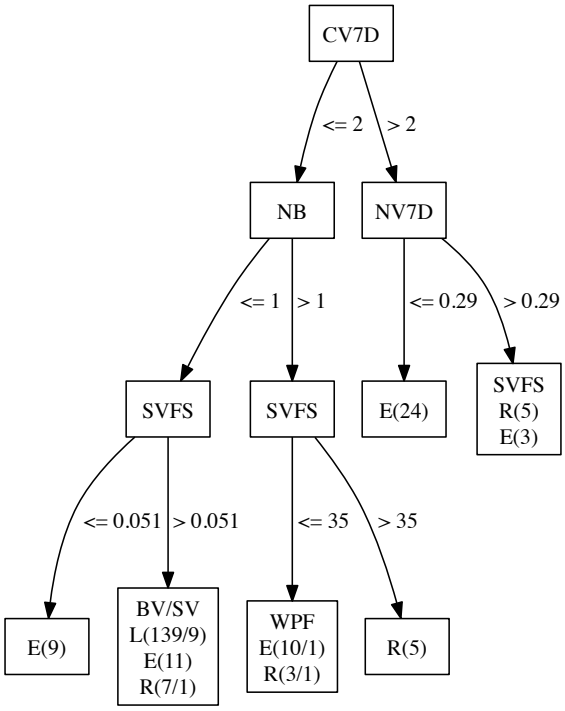


Figure 3.4: Partial decision tree for the three-way classification experiment. Nodes are labeled with the feature based on which the data is split, according to the conditions listed next to the outgoing edges. In leaf nodes, the predicted class of data points in that node are indicated: L for low-profile, E for event-based high-profile, and R for regular high-profile. The numbers of correct predictions are given between brackets, followed by a forward slash and the number of incorrect predictions, if any. We truncated the tree to save space: nodes that contain both a feature name and class predictions represent trees that will split on the indicated feature first, and, in later leaf nodes, yield the given class predictions.

3.2.3 Lessons learned in the two experiments

The similarities between the decision trees for both experiments are clear: the click and search volume features appear with the same threshold values. This is not very surprising as the high-profile class is nothing more than the union of the event-based and the regular high-profile class. There are also differences. When high-profile searches have to be split into event-based and regular query instances, the news volume feature group is one of the top features. Moreover, a Wikipedia feature appears. From each group in Tabel 3.1 a feature is now being used, except for the clickout entropy features: evidence from clicks, searches, news sources and Wikipedia all contribute.

Another finding is that different features from the same groups are quite redundant. From each group typically only one feature plays a prominent role in the decision trees.

We can now answer our research question posed in the introduction. Two-way classification into low- and high-profile queries seems feasible. High recall and precision is obtained for both classes, reflecting reasonable agreement with human classification decisions. Three-way classification is harder, precision and recall for regular high-profile queries are low. There may be several causes for this. First, there were only 24 regular high-profile queries in the dataset. Second, more features may need to be added to capture the meaning of being a well-known person. Particularly promising may be features obtained from the document collections being searched. Note that we have seen that human annotators do not always agree with each other: even for people the task is hard. Features that use clickouts, search volume and news volume are all important, especially for the three way task. It is not very useful to add much redundancy. For example, none of the "trend" features from Tabel 3.1 appeared high in the two decision trees. The clickout entropy feature also does not appear high in the decision trees. A possible explanation for this is that it does not contain very useful information once it has been aggregated to top-level domains only (which was done to ameliorate data sparsity). Since most clicks land on social media platforms, the clickout entropy feature tells us mostly if results from different social media platforms are clicked on. But it would be more informative to know if many different *profiles* are clicked on.

3.3 Conclusions

We discussed a query classification scheme for a specific vertical search engine, viz. a people search engine [231]. The scheme consists of low-profile people queries, event-based high-profile queries and regular high-profile queries. We have shown that people query instances can be automatically classified into high-profile queries and low-profile queries with high precision and recall scores, confirming our findings in [231]. Features that appeared to be particularly informative are click volume and the number of bursts. A further three-way classification into event-based and regular high-profile queries is harder. Here, the most informative features use clickouts, search volume and number of mentions in the news.

Our findings suggest that a people search engine could take into account the predicted class of an incoming query in its retrieval algorithms and in making decisions about result presentation. Even in the three-way classification experiment, event-based queries were

recognized with reasonably high precision and recall. Presenting search results on a timeline is an example of an interesting adaptation to the event-based query type.

Future work may develop more features to increase the effectiveness of classification according to the three-way classification scheme. It is also interesting to study the effect of using the predicted class of to inform the retrieval process. As an example, it may be used as a feature in a learning to rank system.

4

Result Disambiguation in People Search

In the previous chapter we studied what different types of person name queries are submitted to a people search engine. In this chapter we study the retrieval, and more in particular, the organization of search results for person name queries. Regardless of the type of person name query (low-profile, event-based high-profile, or regular high-profile), the high *ambiguity* of person names presents a difficult challenge for a search engine. For the purpose of studying how to deal with this name ambiguity, we treat all three types of person name queries in the same way in this chapter. For a person name query, ideally, a search engine would figure out in which individual a user is interested, and retrieve search results for that person only. But in most cases, search results will refer to many different people with the same name. A search application can meet this problem by performing *result disambiguation*. In the context of people search, this is the problem of finding correct referents for all occurrences of the query person name in the search results. This will allow for a result representation where documents are grouped by person, allowing the user to efficiently zoom in on the documents referring to the individual of interest. As we have discussed in Chapter 2, Section 2.3.1, Result disambiguation has been studied in the Web People Search (WePS) campaigns [11–13]. For each query in a set of person name queries, the organizers retrieved a a hundred search results using a major web search engine, and produced a gold standard clustering, to which clusterings generated by partipants in the WePS campaigns could be compared. One of the lessons learned in those campaigns is that standard hierarchical agglomerative clustering (HAC) approaches using textual features achieve high performance on the task.

We revisit the result disambiguation problem for people search and we do so in the setting of a people search engine, a vertical meta-search engine that aggregates people search results from a broad range of sources, both generic web search engines and social media platforms. As an example, consider the people search engine we studied in Chapter 2. Recall that it presents search results obtained from social media platforms on the SERP, while search results from web search engines are displayed on a separate page. This reflects recognition of the omnipresence of social media profiles on the web, and of their importance among search results. However, it is tedious for a user to find documents referring to the individual of interest among different SERPs, and for this chapter, we envisage a scenario where a single SERP would be generated, containing both social media profiles (from any platform) and web search engine results (from any web search engine), in a single result list. The inclusion of large numbers of social media proiles

in the aggregated result list poses new challenges for result disambiguation methods that have previously been shown to be very effective. Specifically, social media profiles are textually sparse and contain relatively large amounts of boilerplate material, making it non-trivial to extract good textual features from them.

We propose a *dual strategy*: different ways of treating social results and other, "non-social" results. We examine the effectiveness of various disambiguation techniques on social results and we contrast our findings with known results for non-social results. The two types of document require different strategies for effective result disambiguation. We then propose and examine techniques for combining the outcomes of result disambiguation on social results with those on non-social results. We address the second research question from the introduction and ask:

RQ2 State-of-the-art hierarchical aglommerative clustering (HAC) methods for clustering web search results for person name queries break down on search results of a people search engine, which contain many social media profiles. Can we remedy this problem by treating social media profiles differently from regular web documents, clustering the two types of documents separately and then merging the clusterings back together?

Our main contributions are: (1) signaling the problem that people search results contain increasing numbers of social media profiles plus its negative impact on existing disambiguation strategies for "traditional" web search results; (2) a new strategy of treating social and non-social results separately and combining the resulting clusterings; (3) a detailed error analysis. We also make available the data set (queries, ground truth, search results) used in this chapter.

In Section 4.1 we detail our methods, and in Section 4.2 our experimental setup. We show the results, answer our research question and perform an error analysis in Section 4.3. We discuss the parameter sensitivity of our methods in Section 4.4. Finally, we offer conclusions and point out directions for future work in Section 4.5.

4.1 Dual strategies for result disambiguation

In this section we describe our methods for disambiguating search engine results in the setting of web people search. We start with a high level overview and then zoom in on individual steps that make up our method.

As explained in the introduction, we work in the setting of a meta-search engine, one that aggregates search results from generic web search engines and from a range of social media platforms into a single result list D. The goal is to create a clustering of this aggregated result list such that documents are grouped around the individuals they refer to. We propose a dual strategy, with separate disambiguation steps for social and non-social documents, followed by a merge step in which we combine the results of the two clustering steps. Algorithm 4.1 outlines this strategy, introducing some notation. The following paragraphs discuss the individual steps of the dual strategy.

Algorithm 4.1: Dual strategy for result disambiguation

1: **Input**: query q, search engine results D
2: **Uses**: clustering methods M_w and M_s
3: Split document set D into two sets, social documents D_s and non-social documents D_w
4: **for** non-social documents D_w **do**
5: disambiguate D_w by creating clusters using method M_w
6: **return** clustering W
7: **end for**
8: **for** social documents D_s **do**
9: disambiguate D_s by creating clusters using method M_s
10: **return** clustering S
11: **end for**
12: Merge cluster results W and S into final clustering C
13: **return** merged clustering C

Splitting D into social and non-social documents Let D be the set of result documents retrieved for a given person name query. We use two methods to split the documents into social and non-social documents. The first method adds a search result d to the set of *social* results D_s if it was obtained through the API of a social media platform, i.e., in this study, Hyves, Facebook, LinkedIn, Twitter, and MySpace. Otherwise, it adds the document to the *non-social* results D_w. We call this splitting method "By source". The idea behind it is that D_w is qualitatively more similar to the WePS collections than D, because the WePS collections also contained search results obtained from a general purpose web search engine.[1] Note that when we split by source, D_w may still contain social media profiles, if they were returned by a web search engine, just like the WePS collections contain social media profiles. Thus, we split by source to enable a better comparison to results obtained in the WePS campaigns. Our second splitting method splits the social and non-social search results a bit more aggressively, to achieve a better performance. It adds d to the set of *social* results if the URL of a result document d contains the top level domain of one of the social media platforms we consider. Otherwise it adds d to the *non-social* results D_w. We dub this method "By URL."

Clustering methods considered for M_w and M_s We now discuss the clustering methods that we consider. Due to the fact that the algorithm has to run online, at query time, we have a strong preference for relatively light-weight methods.

- One-in-one: This simple baseline method creates a singleton cluster for each document.
- All-in-one: This simple baseline method creates one cluster that contains all documents.

[1] Yahoo!

- HAC: Hierarchical agglomerative clustering (HAC) approaches have been successful in WePS-2, as we have seen in Chapter 2. Recall that HAC iteratively merges cluster pairs with the highest similarity. It stops when this similarity does not exceed a fixed threshold, when a fixed number of iterations has been performed, or when all items have been merged in a single cluster. Here, we use a fixed similarity threshold. Different cluster similarity measures can be used. We report on experiments that use singe link clustering (the similarity between the two most similar points, one from each cluster), and centroid clustering (the similarity between the centroids of both clusters). To compute the similarity between two documents, we employ cosine similarity between document vectors with tf-idf term weights. Inverse document frequencies (idf) are calculated with respect to search results over all queries, and term frequencies are normalized as in [158]. We use Porter stemming. See Section 4.2 for details on the parameter settings.

Additional clustering methods considered for M_s We apply four methods that are specifically targeted at results obtained from social media platforms: (1) Cross links, (2) Co-clicks, (3) Clicked in the same burst, and (4) Face clustering. For the first three methods, we generate a binary similarity matrix, and subsequently perform HAC clustering. Our face clustering method generates a clustering of social media profile pictures, and we then generate the corresponding clustering of social media profiles. The four methods in a bit more detail:

- Cross links: If we find a hyperlink between two result documents, we set the similarity of these two documents to one, and otherwise the similarity is zero.

- Co-clicks: For each query and people search engine user, we check if this user clicked two results from different social media platforms for the same query. If this is the case for at least two users, we set the similarity between these results to one, otherwise to zero.

- Clicked in same burst: In the query logs of the people search engine, we count the unique daily visitors who issue a given query, resulting in a time series. We then define a burst in this time series to be a number of consecutive days in which the unique daily search count exceeds the mean daily search count plus two standard deviations. In addition, a burst-day needs to have at least ten unique daily searches. We record for each search the last clicked result document. If two search results are clicked on last during the same burst, we set their similarity to one, otherwise to zero.

- Face clustering: We extract all user profile pictures from the social media profiles and load them in Google Picasa, which has a built-in face clustering component. We let Picasa find groups of faces using its default parameters and without any user feedback. We then cluster the corresponding profiles.

Merging methods We consider two methods for merging clusterings W and S of the non-social and social result documents contained in D.

Algorithm 4.2: Merging social media cluster results S with non-social cluster results W

Input: (1) Social clustering $S = \{S_1, \ldots, S_I\}$, where S_i is a cluster of social search results from D_s; (2) non-social clustering $W = \{W_1, \ldots, W_J\}$, where W_j is a cluster of non-social search results from D_w; (3) a between-cluster similarity threshold $\tau \in [0, 1]$; (4) a social penalty $p \in \mathbb{R}_+$ that decreases similarity with clusters in W once they contain social results; and (5) a cluster similarity function $HACsim$.
Output: A final clustering C.
while $S \neq \emptyset$ **do**
  $A \longleftarrow \{A_1, \ldots, A_J\}$, where $A_j \longleftarrow \emptyset$; A_j will contain the set of social clusters that apply for a merge with the non-social cluster W_j.
  $T \longleftarrow \emptyset$; T will contain social clusters that cannot apply for a merge with any non-social cluster.
  for cluster S_i in S **do**
    find cluster $W_j \in W$ with highest $sim(S_i, W_j) = \frac{HACsim(S_i, W_j)}{1+np}$, where n is the number of social results already in W_j
    if $sim(S_i, W_j) < \tau$ **then**
      $T \longleftarrow T \cup \{S_i\}$
      $S \longleftarrow S \setminus \{S_i\}$
    else
      $A_j \longleftarrow A_j \cup \{S_i\}$
    end if
  end for
  for cluster W_j in W **do**
    find cluster $S_i \in A_j$ with highest $sim(W_j, S_i) = \frac{HACsim(W_j, S_i)}{1+np}$,
    $W_j \longleftarrow W_j \cup S_i$
    $S \longleftarrow S \setminus \{S_i\}$
  end for
  $W \longleftarrow W \cup T$
end while
return merged clustering W

Table 4.1: Result disambiguation methods.

Name	Splitting	M_w	M_s	Merge method
Dual baseline	By URL	HAC single link	One in one	$C \longleftarrow W \cup S$
Dual merge	By URL	HAC single link	One in one	Algorithm 4.2

- Baseline merge: This algorithm is precision oriented. It does not merge any social clusters with non-social clusters, but simply returns the union of the respective clusterings: $C \longleftarrow W \cup S$.
- Advanced merge: This algorithm does attempt merges between the two kinds of clusters. The main intuition behind it is that the clusters in W and, to a lesser extent, S, contain more evidence than individual search results do, making similarity estimates more reliable. Another goal of the algorithm is to discourage further merges between social clusters. Specifically, we apply Algorithm 4.2 to merge the social and non-social clusterings S and W. The behavior of the merging algorithm is controlled by three additional parameters. The minimal similarity threshold required to merge a non-social with a social cluster is given by τ. A penalty factor p regulates how much to decrease the similarity between a social cluster and a non-social cluster that already contains one or more social media results. The between-cluster similarity function that is used is given by $HACsim$, i.e., single-link or centroid similarity. The algorithm first enters a loop that will end when the set of social clusters is empty (line 3). In that loop, it organizes a round of merges. First, each social cluster applies for a merge with the non-social cluster that is most similar to it, unless the similarity is too low to warrant a merge, in which case the social cluster is set aside (lines 4–14). Next, each non-social cluster chooses amongst its applicants the social cluster closest to it, and merges it into itself (lines 15–19). Then, the previously set aside social clusters are added to the non-social clustering, before the next round of possible merges begin (line 20). Eventually, all social clusters will either have been merged to a non-social cluster, or they will have been set aside and added to the non-social clusters.

Dual result disambiguation methods considered The options for the splitting method, the clustering method for non-social results (M_w), the clustering method for social results (M_s), and the merge method, give rise to a large number of combinations for dual strategy runs (Algorithm 4.1). We limit ourselves to the methods for result disambiguation specified in Table 4.1, picking for each step the option for which we expect the best performance. Our expectations are based on our first set of experiments, and we will explain our choices in Section 4.3.

4.2 Experimental setup

To test previously established methods and our own methods, we use the experimental setup detailed in this section. We introduce the query and document sets used, report

on our ground truth creation, explain our parameters, and finally, discuss the metrics on which we report.

Query and document set We select queries from query logs of a people search engine. The logs have been collected between September 2010 and February 2011 and contain queries, associated clicks, and browser cookies (for user identification); Weerkamp et al. [231] present a detailed study of the logs; see also Chapter 3. To select ambiguous queries, we required queries to have clicks to at least three profiles within one social media platform. In addition, we required that at least seven searches were performed with clicks to at least two search engines or social media platforms, to make sure we have some evidence in our click data for clustering. From this population of queries, we randomly selected queries. For each selected query, the document set is constructed by retrieving 20 documents (profiles) from each of five large social media platforms (e.g., Facebook, LinkedIn, Twitter) and 50 documents from three major web search engines (Google, Yahoo! and Bing). The resulting document set is de-duplicated based on URL. Documents that do not contain the person name are ignored. After constructing the document set, we created a ground truth clustering, as described in the next paragraph. In total, we created document sets and annotations for 33 queries. We do not reprint all the person name queries here, for reasons of privacy, but the dataset and annotations are available.[2]

Ground truth Annotations were created in the same way as for the WePS campaigns, allowing for a comparison between results obtained on our data collection and results obtained on the WePS-1 and WePS-2 datasets (see Chapter 2). A document can be assigned to multiple clusters, if the document contains references to two or more persons with the same name. If the annotator could not say for sure whether a document belonged to existing clusters, a new cluster was created. Annotations were done on the full document set for a query and not on separated datasets as a result of splitting. In case no evidence whatsoever could be found in a document (e.g., private profiles without pictures), the document was discarded. We also discarded documents in the web results that were returned by other people search engines. We distributed queries to multiple annotators, with insufficient means to study inter annotator agreement.

Figure 4.1 shows the number of annotated results per person name query in our dataset. We have split out the results in non-social (D_w) and social results (D_s) "by URL." The queries are ordered by the number of annotated documents.

Parameter settings When it comes to applying HAC, we follow [158]. They did not report on a minimal threshold stopping criterion value. We perform a partial parameter sweep on the WePS-2 dataset, resulting in the value 0.225 which we use for both single link and centroid HAC on all datasets. For our dual strategy with merge (viz. Algorithm 4.2) we use the following parameter values: single link clustering as $HACsim$, $\tau = 0.5$, and $p = 1$. We did not use a separate training set, but explored a few combinations on our test set. We report on parameter sensitivity in Section 4.4.

[2] See `http://ilps.science.uva.nl/resources/ecir2012rdwps`.

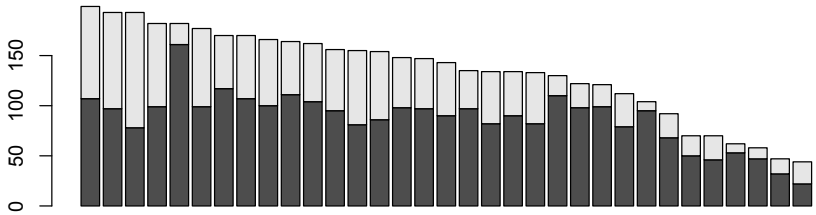


Figure 4.1: Number of documents (y-axis) per person name query (x-axis) in our dataset, split "by URL" in D_s and D_w. Dark boxes contain non-social results D_w; light boxes contain social results D_s.

Metrics We use the B-cubed metrics [6] for evaluation of cluster quality, as was done in WePS-2. An extended version of the metrics is used to accommodate for overlapping clusters. For each topic, B-cubed precision (B^3P) and B-cubed recall (B^3R) are computed, and a macro-averaged F-measure with $\beta = 0.5$ is computed. In the presentation and discussion of our results, when we discuss performance, we refer to performance in terms of $F_{\beta=0.5}$, unless we mention precision or recall explicitly, in which case we refer to the B-cubed variants of precision and recall.

Significance tests We use a paired randomization test, as in, e.g., [209]. We look for significant differences at an optimistic level $\alpha = 0.05$, denoted $^{\triangle}$(or $^{\triangledown}$) and a conservative level, $\alpha = 0.001$, denoted $^{\blacktriangle}$(or $^{\blacktriangledown}$).

Unanimous Improvement Ratio The F-measure weighs precision and recall with the β parameter. We set β to 0.5 as in the WePS campaigns, to favour neither precision nor recall. Choosing a different β may affect system ranking. To estimate which pairwise performance differences in $F_{\beta=0.5}$ are robust against different β values, we employ the Unanimous Improvement Ratio (UIR) [7]. For two systems A and B, let T_A be the set of queries for which system A achieves precision and recall scores that are greater than or equal to the scores of system B. For these queries, the F-score for system A will not be smaller for any value of β. Let T be the set of all queries. Then $\mathrm{UIR}(A, B) = (|T_A| - |T_B|)/|T| \in [-1, 1]$. For the people search clustering task, Amigó et al. [7] give a rule of thumb that we employ in our result section: if $|\mathrm{UIR}(A, B)| \geq 0.25$ then an observed performance difference in $F_{\beta=0.5}$ between system A and B is robust with regards to the particular value chosen for β. We refer to such performance differences simply as *robust* in the remainder of this chapter. We denote robust performance differences with $\uparrow$ or $\downarrow$.

Table 4.2: Performance in terms of B-cubed precision, B-cubed recall, and macro averaged F-measure of standard methods on search results from all sources, i.e., on D. For $F_{\beta=0.5}$, both significant and robust differences, if indicated, are with regard to HAC single link.

	B^3P	B^3R	$F_{\beta=0.5}$
All-in-one	0.17	1.00	0.25▼
One-in-one	1.00	0.48	0.62
HAC single link	0.56	0.87	0.67
HAC centroid	0.72	0.71	0.69

4.3 Results and analysis

We report on three sets of experiments. First, we examine the performance of methods from the literature on our people search engine data set, for which we consider the performance on the full data set, and its restrictions to search engine results and social media results. This is to confirm the premise of our research question, that state-of-the art methods for people search result clustering break on our data set, which contains more social media profiles. Second, we examine the performance of methods designed for social media profiles. Third, we present results of our dual strategies (Table 4.1) on the full data set. Based on the outcomes we answer RQ2. We finish this section with an error analysis of our methods.

4.3.1 Results

We present the results of applying default clustering methods for web people search results on our people search engine dataset in Table 4.2. The performance of single link HAC on our full dataset is substantially lower than scores reported for this algorithm on the WePS-2 data, e.g., Monz and Weerkamp [158] report an F-score of 0.81.[3] While absolute performance scores for the same algorithm on two different datasets may vary substantially, the large discrepancy provides a motivation to investigate what causes difficulties for the HAC algorithm. There is another indication that HAC is not very effective on our full dataset. In Table 4.2, the improvement of single link HAC over the one-in-one baseline is not significant or robust (UIR < 0.25).

In Table 4.3, we report performance of the same clustering methods, but this time on either just search results from web search engines (left side) or just search results from social media profiles (right side). When we restrict ourselves to documents returned by web search engines (left side), we find that the performance of HAC improves dramatically, approaching the levels reported for WePS-2. Here, single link HAC improves significantly over the one-in-one baseline. The improvement is not robust. On social media documents (right side), the performance of HAC is about as bad as the all-in-one baseline, which simply adds all documents to a single cluster; see the right side of

[3] Our HAC single link implementation achieved an $F_{\beta=0.5}$ score of 0.78 on the WePS-2 dataset. Differences like this may occur due to differences in preprocessing, e.g., tokenization.

Table 4.3: Performance in terms of B-cubed precision, B-cubed recall, and macro averaged F-measure of standard methods on either web search engine (D_w) or social media platform search results (D_w), where D was split "by source" into D_w and D_s. For $F_{\beta=0.5}$, both significant and robust differences, if indicated, are with regard to HAC single link. The best $F_{\beta=0.5}$ scores are in boldface.

	Search results from search engines			Search results from social media platforms		
	B^3P	B^3R	$F_{\beta=0.5}$	B^3P	B^3R	$F_{\beta=0.5}$
All-in-one	0.22	1.00	0.31 ▼	0.13	1.00	0.20
One-in-one	1.00	0.43	0.58 ▼	1.00	0.86	**0.92** ▲
HAC single link	0.76	0.86	**0.79**	0.14	1.00	0.21
HAC centroid	0.89	0.70	0.75	0.17	0.96	0.27 ▲

Table 4.4: Performance in terms of B-cubed precision and recall, and macro averaged F-measure of social clustering methods on search results from social media platforms. The best $F_{\beta=0.5}$ score, for the one-in-one method, is given in boldface. For $F_{\beta=0.5}$, significant and robust differences, if indicated, are with regards to the one-in-one method.

	B^3P	B^3R	$F_{\beta=0.5}$
One-in-one	1.00	0.86	**0.92**
Cross links	0.83	0.88	0.84 ▼↓
Co-clicks	0.99	0.87	0.91
Clicked in Same Burst	0.98	0.86	0.91
Picasa	1.00	0.86	0.92

Table 4.3.

These experiments also reveal that the difference between single link and centroid HAC is limited. On our full dataset and on the social media profiles, centroid HAC works best. If we limit ourselves to the web search results, we find that single link HAC performs better. The difference is only significant on the social media profiles, and it is nowhere robust. Looking at the social media results, we find that the one-in-one baseline is the best system. The observed difference in best performing approaches between the two document types clearly shows that the two behave very differently, which motivates our dual strategies. From the experimental results reported on social media documents, it is almost safe to assume that each document corresponds to a unique individual.

Next, we turn to result disambiguation methods that focus on social results. Table 4.4 lists the results of these experiments. We observe that none of the "social" methods is able to beat the one-in-one baseline. The only method that differs significantly from the one-in-one baseline is "Cross links," but it is worse (▼). It is also the only method over which the improvement of the one-in-one baseline is robust (↓).

Our final set of experimental results concerns the dual strategies (dual baseline and

Table 4.5: B-cubed precision and recall, macro averaged F-measure of the dual strategies on search results from all sources. The best $F_{\beta=0.5}$ score is given in boldface. In the two rightmost columns, it is indicated if observed performance differences in $F_{\beta=0.5}$ are significant and robust, with regard to the dual baseline and the dual merge method, respectively.

				Significance and robustness of performance difference with:	
	B^3P	B^3R	$F_{\beta=0.5}$	Dual baseline	Dual merge
All-in-one	0.17	1.00	0.25	▼	▼
One-in-one	1.00	0.48	0.62	▼	▼↓
HAC single link	0.56	0.87	0.67	▼	▼
HAC centroid	0.72	0.71	0.69	▼↓	▼↓
Dual baseline	0.90	0.78	0.82	-	▼↓
Dual merge	0.90	0.80	**0.83**	▲↑	-

dual merge). Recall that both the dual baseline method and the dual merge method split search results by URL, perform single-link HAC on the non-social search results D_w, and perform one-in-one clustering on the social results D_s, as listed in Table 4.1. We split by URL because we have observed in our experiments that the presence of social media profiles hurts clustering performance. Splitting by URL will remove even the social media profiles that were obtained from web search engines from D_w, and thus we expect it to do better than splitting by source. To cluster non-social results, we use single-link rather than centroid HAC, because single-link HAC performed best on non-social results. To cluster social results, finally, we chose the one-in-one clustering method, because it performed best on social results. The only difference between dual baseline and dual merge, then, is how they merge the social and non-social clusterings. The results for this final set of experiments are listed in Table 4.5. For convenience, besides reporting on the dual baseline and the dual merge method, we repeat the scores of the clustering methods from Table 4.2, which do not split search results in social and non-social results. In the rightmost two columns of Table 4.5, we indicate for each method if observed performance differences between it and the two dual strategy methods are significant and robust. We observe that even with a naive merging strategy (dual baseline), we manage to achieve scores comparable to those achieved with HAC on WePS-2 (e.g., in [158], $F_{\beta=0.5} = 0.81$). Clearly, we are able to suppress the negative impact resulting from social media results. The dual baseline has large and significant improvements (▲) over all other methods on our full dataset. It improves robustly only over centroid HAC, however. With regard to single link HAC, the dual baseline improves precision on all topics, but it also looses a bit of recall on all topics, indicating that there is room for improvement. The more sophisticated merge method (Algorithm 4.2) improves slightly but significantly (▲) and robustly (↑) on the dual baseline. It has higher recall on about six out of ten queries and never a lower precision: the intended effect of this method. Dual

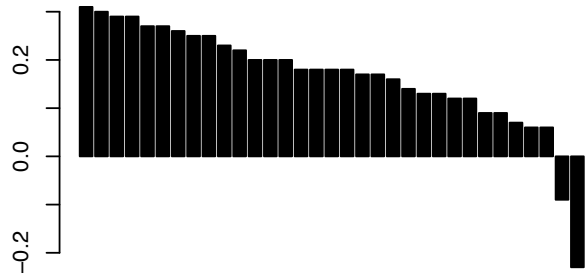


Figure 4.2: Difference in $F_{\beta=0.5}$ score per query between dual merge and HAC single link; a positive difference indicates a query where dual merge outperforms HAC single (and vice versa).

merge has robust improvements also over centroid HAC and the one-in-one baseline.

With these results, we can now answer RQ2. Indeed if we treat social media profiles differently from regular web documents, clustering the two types of documents separately and then merging the clusterings back together, we obtain performance on par with reported levels of performance in WePS campaigns [11–13]. For clustering social media profiles, the most succesful approach is singleton clustering, where each profile is used to form a cluster on its own. When merging the social and non-social clusterings, a robust method is to let the final clustering be the union of the clusters in the original clustering. It is possible to obtain additional improvements with our iterative method described in Algorithm 4.2. The improvements are small, but significant.

4.3.2 Analysis

In our analysis, we compare our dual strategy with the single link HAC baseline and we investigate why our "social" methods fail to improve over the one-in-one baseline.

Dual merge vs. single link HAC As shown in the previous subsection, single link HAC is the best performing method on the results from search engines, which is why we use it as M_w in our dual strategy approaches. Here, we compare this method to our dual merge strategy. Figure 4.2 compares the difference between the dual merge strategy and our baseline (single link HAC) on a per-query basis. For almost all queries, dual merge achieves a clear improvement over single link HAC.

Our strategy of treating social media documents in a separate manner leads to large improvements and we expect to see a stronger improvement in cases where more social media documents are present. Figure 4.3 shows, however, that there is no clear correlation between the ratio of social media results returned for a query and the improvement after distinguishing between social and non-social search results.

Returning to Figure 4.2, the query that shows the largest drop in performance, going from the HAC baseline to our dual strategy method, is the query with the highest ratio of social documents. For this query, all search engines return noise: after automatically filtering out results that did not contain the person name, we are left with only 27 results for this query. During annotation, another 17 of these results were discarded, leaving

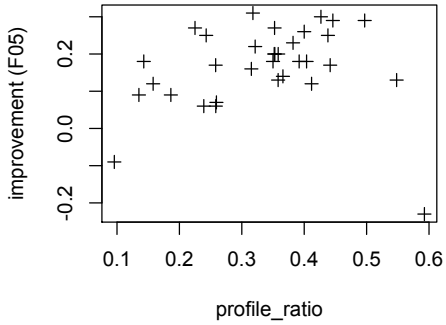


Figure 4.3: Improvement in $F_{\beta=0.5}$ (vertical axis) versus the ratio of social documents in the total result set for a given query (horizontal axis).

only ten documents for clustering. Some profiles among these documents do refer to the same person, leading to the degraded performance for this query.

The second query for which our dual merge method performs worse than the baseline concerns a not very common name, but it is the name of a celebrity (*Joey Spaan*). Consequently, this person dominates the search results completely. He has profiles on various social media platforms, and since our dual merge strategy is designed to cluster only few social media profiles, the small loss in recall for this query is unsurprising.

Analysis of "social" methods The performance of baseline clustering methods on social media results shows that such results should be treated differently from other web documents. The good performance of the one-in-one baseline is caused by people generally (1) having only one profile per platform, and (2) using different platforms for very different reasons. While web documents returned by general web search engines can be completely dominated by a single person, many people will typically be represented in the social media results. Besides, it is likely that people make an effort to keep their Facebook profile (for friends) separated from their LinkedIn profile (for work related contacts), leading to limited overlap in content.

The one-in-one baseline on social media profiles is not perfect. We investigate why our "social" methods fail to discover the few clusters that are there. First, in eight out of our 33 queries, the one-in-one baseline is actually perfect. We examined fifteen random queries of the remaining 25 and found that the main reasons for our annotators to cluster social documents together are: (1) a user profile picture (40 pairs of profiles), (2) a company name or affiliation (12 pairs) and (3) an occupation (11 pairs).

The method that leads to the highest recall is the cross links method, although it looses precision, too. A simple cause for this is that, for example, LinkedIn profiles contain links to other profiles with the same person name ("Find a different John Smith"). Adding a rule that ignores within platform cross links leads to too few links to make a

noticeable difference. The Co-clicks method and the Clicked in same burst method have almost no effect on performance. They share one problem: clicks are very sparse in the query logs of our people search engine, making them hard to use. Finally, our face recognition method fails to recognize and match enough faces in the user profile pictures to improve recall noticeably. It would be interesting to explore this direction further, perhaps taking into account other pictures beside profile pictures, and perhaps obtaining similarities between detected faces instead of the hard clustering we obtained from the Google Picasa software. All in all, we fail to improve over the one-in-one baseline with our specific "social" methods.

It proves challenging to identify sufficient textual evidence in the social documents. In follow-up experiments, we considered dedicated content extractors for each of the social media platforms, so that only relevant text is extracted and not the boilerplate material. Using these extractors results in an increase in precision for the single link HAC baseline, but it also leads to a drop in recall, resulting in little change in F-score.

4.4 Discussion

In this section we explore the impact of various parameters on our results. First, we look at the similarity threshold in HAC and second, we explore the parameters of our merging algorithm.

The minimal similarity threshold in HAC Artiles et al. [12] observe that performance of HAC is strongly dependent on the minimal similarity threshold used as a stopping criterion. Different topics have different optimal thresholds and the authors provide an upper and lower bound for HAC by doing a parameter sweep and taking for each query the optimal value. The variety in optimal threshold is such that learning an average optimal value on one dataset is no guarantee for success on another dataset.

We try a different, query-dependent approach to estimating the similarity threshold, based on the observation that if a name is very ambiguous, we would require more evidence to cluster two documents with this name together. For example, it would not be unlikely to have two different John Smiths's playing basketball in New York, but it would be unlikely to have two Jack Rumplestilskin's doing so. Thus we expect that if a name is more ambiguous, a higher similarity threshold would be required. One reasonable way of estimating name ambiguity is counting the number of unique user profiles on a social media platform that is widely used and where people tend to use their real name. We choose LinkedIn for this purpose. In Figure 4.4, after a parameter sweep on the WePS-1 data, we plot the best performing value minimal similarity threshold value of each person name query as a function of the number of profiles with that name. It turns out that the ambiguity of a name cannot explain the variance in the best performing threshold value very well. For names with a moderate number of profiles, the best threshold values cover the full range. Still, for a small number of person name queries with a large number of profiles, the best performing threshold values average to a value of approximately 0.36, which is higher than the average for all queries. Based on these observations, we perform a test run on the WePS-2 data with the following simple rule: if there are more than 500 LinkedIn profiles for a given query, use a similarity threshold of 0.360, otherwise

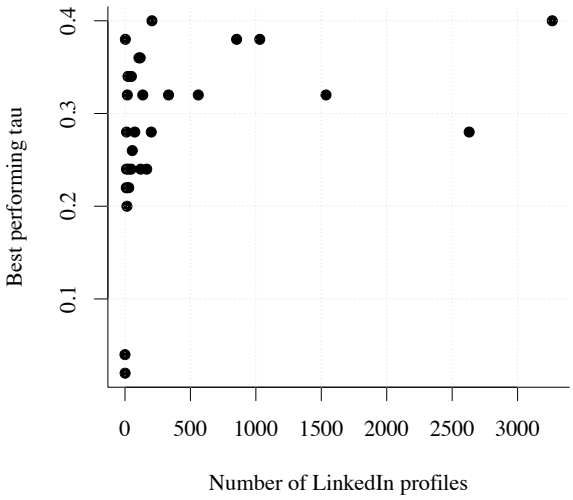


Figure 4.4: The best performing minimal similarity threshold τ (used as a stopping criterion in HAC single link) for each person name query in the WePS-1 dataset (vertical axis) plotted as a function of the number of LinkedIn profiles with that name (horizontal axis).

Table 4.6: Impact of parameters τ (Top) and p (Bottom) on the performance of the dual merge method.

τ	p	B^3P	B^3R	$F_{\beta=0.5}$
$\tau = 0.225$	$p = 1.0$	0.77	0.82	0.78
$\tau = 0.500$	$p = 1.0$	0.90	0.80	0.83
$\tau = 0.775$	$p = 1.0$	0.90	0.78	0.82

p	τ	B^3P	B^3R	$F_{\beta=0.5}$
$p = 0.0$	$\tau = 0.5$	0.74	0.82	0.76
$p = 0.5$	$\tau = 0.5$	0.86	0.81	0.82
$p = 1.0$	$\tau = 0.5$	0.90	0.80	0.83
$p = 1.5$	$\tau = 0.5$	0.90	0.78	0.82
$p = 2.0$	$\tau = 0.5$	0.90	0.78	0.82

use the default threshold of 0.225. Results of this experiment show a slight increase in precision, but an equal drop in recall, leading to the same F-score as a run without query-dependent thresholding. Finding a more sophisticated way to use query characteristics to predict thresholding parameters is an interesting direction for future research, e.g., by using multiple query features and try to fit a model on the optimal threshold for those queries using machine learning techniques.

Parameter sensitivity of our merging algorithm Algorithm 4.2 has a number of parameters. For $HACsim$ we use single link clustering, as it performs best overall. Here, we explore the impact of the minimal similarity threshold (τ) and the parameter p, which regulates how strong the similarity between a social and a non-social cluster decreases for each social cluster already present in the non-social cluster. Table 4.6 lists the performance for different values of τ (Top), while keeping p stable, and different values of p (Bottom), while keeping τ stable. We find that increasing τ leads to better precision, but decreasing recall. We find a similar pattern of improving precision with higher values for p, at the cost of recall. We recommend setting τ to 0.5 and p to 1, as it achieves highest precision, while sacrificing only little recall. Insisting on higher recall implies having to accept a sharper drop in precision.

4.5 Conclusion

In this chapter we studied the problem of disambiguating the search results of a people search engine. Our results show that the increasing availability of results retrieved from social media platforms causes state-of-the-art methods to break down.

We proposed a dual strategy where we treat social search results differently from non-social results. For non-social results, we used single link HAC as a strategy, the best performer when we evaluate on the subset of results obtained via generic web search

engines. For social results, we investigated several methods but were unable to beat the one-in-one baseline, which considers each document as a cluster on its own. Therefore, we selected the one-in-one baseline as a strategy here. We tested two methods for merging the two clusterings. The first is a baseline method that simply takes the union of both clusterings. This method already achieved a large boost in precision. With the second merging method we were able to gain recall without losing precision, obtaining results comparable to state of the art results obtained on WePS datasets.

Future work should explore possibilities to estimate query-dependent stopping thresholds for HAC. While face recognition failed to improve our results here, it seems a promising direction also, particularly if more images are used besides user profile images, and raw similarities between detected faces could be incorporated in the clustering algorithm, instead of only hard predictions.

5

On the Evaluation of Expertise Profiles

In the previous two chapters we have studied two problems in the people search setting. In this chapter, we study a specialization of people search, namely expertise retrieval [18]. We focus on evaluation. Given different ways of creating ground truth, what are differences in evaluation outcomes? Before repeating RQ3 from the introduction, we provide background material on expertise finding and topical profiling. An organization's intranet provides a means for exchanging information between employees and for facilitating collaborations between employees. To efficiently and effectively achieve this, it is necessary to provide search facilities that enable employees not only to access documents, but also to identify expert colleagues [98]. At the TREC Enterprise Track [17, 23, 60, 211] the need to study and understand *expertise retrieval* has been recognized through the introduction of the expert finding task. The goal of *expert finding* is to identify a list of people who are knowledgeable about a given topic: *Who are the experts on topic X?* This task is usually addressed by uncovering associations between people and topics [24]; commonly, a co-occurrence of the name of a person with topics in the same context is assumed to be evidence of expertise. An alternative task, building on the same underlying principle of computing people-topic associations, is *expert profiling*, where systems have to return a list of topics that a person is knowledgeable about [19, 20]. Essentially, (topical) expert profiling turns the expert finding task around, and asks: *What topics does a person know about?*

Expert profiling is useful in its own right for users who want to profile experts they already know. It is also a key task to address in any expert finding system; such systems rank experts, and users will want to navigate to profiles of these experts. Complete and accurate expert profiles enable people and search engines to effectively and efficiently locate the most appropriate experts for an information need. In addition to a topical profile, it is recognized that social factors play a large role in decisions about which experts to approach [19, 64, 100, 207].

We focus on the topical expert profiling task in a knowledge-intensive organization, i.e., a university, and release an updated version of the UvT expert collection [35], which was created with data from the *Universiteit van Tilburg* (UvT, *Tilburg University*). Because the university no longer uses the acronym UvT and has switched to TU instead, we call the updated collection the *TU expert collection*.[1] The TU expert collection is

[1]The TU expert collection is publicly available at `http://ilps.science.uva.nl/tu-expert-collection`. For a description of the items contained in the collection, please see the appendix to this

based on the *Webwijs* ("Webwise") system[2] developed at Tilburg University. *Webwijs* is a publicly accessible database of TU employees who are involved in research or teaching, where each expert can indicate his/her skills by selecting expertise areas from a list of knowledge areas. Prior work has used these self-selected areas as ground truth for both expert finding and expert profiling tasks [18, 20]. With the TU expert collection come updated profiles consisting of these self-selected knowledge areas; we refer to this set of areas as *self-selected knowledge areas*.

One problem with self-selected knowledge areas is that they may be sparse. There is a large number of possible expertise areas to choose from (more than 2000). When choosing their knowledge areas, experts may not necessarily browse the set of knowledge areas very thoroughly, especially since the interface in which they select the areas lists them in alphabetical order without providing links between related areas. This might result in sparse data with a limited number of knowledge areas assigned to each expert. Using these self-selected knowledge areas as ground truth for assessing automatic profiling systems may therefore not reflect the true predictive power of these systems. To find out more about how well these systems perform under real-world circumstances, we have asked TU employees to judge and comment on the profiles that have been automatically generated for them. Specifically, we have employed state-of-the-art expertise retrieval methods to construct topical expertise profiles. TU employees were then asked to re-assess their self-selected knowledge areas based on our recommendations; in addition, they were given the option to indicate the level of expertise for each selected area. Moreover, they could give free text comments on the quality of the expert profiles. We refer to this whole process as the *assessment experiment* in this chapter.

The research in this chapter can be grouped in two parts. In the first part, we perform a detailed analysis of the outcomes of the assessment experiment. One important outcome is a new set of graded relevance assessments, which we call the *judged system-generated knowledge areas*. We examine the completeness of these new assessments. The knowledge areas experts selected, and the textual feedback they gave provide us with a unique opportunity to answer the question: "How well are we doing at the expert profiling task?" We perform a detailed error analysis of the generated profiles and a content analysis of experts' feedback, leading to new insights on what aspects make expertise retrieval difficult for current systems.

In the second part, we take a step back and ask: "Does benchmarking a set of expertise retrieval systems with the judged system-generated profiles lead to different conclusions, compared to benchmarking with the self-selected profiles?" We benchmark eight state-of-the-art expertise retrieval systems with both sets of ground truth and investigate changes in absolute system scores, system ranking, and the number of significant differences detected between systems. We find that there are differences in evaluation outcomes, and are able to isolate factors that contribute to these differences. Based on our findings, we provide recommendations for researchers and practitioners who want to evaluate their own profiling systems. The research questions in this chapter are summarized in RQ3:

chapter.

[2] http://www.tilburguniversity.edu/webwijs/

RQ3 We ask experts to judge profiles we generate for them. What is the quality of our generated profiles? Which experts are hard to generate a profile for and why? Previously, evaluation of expert profiling algorithms had been done by using profiles of knowledge areas that experts selected from an alphabetical list. If we use judged system-generated profiles for evaluation, what, if any, are the differences in evaluation outcomes?

The main contributions in this chapter are:

- The release of a test collection for assessing expert profiling (the TU expert collection) plus a critical assessment and analysis of this test collection. Test collections support the continuous evaluation and improvement of retrieval models by researchers and practitioners: in this case in the field of expertise retrieval.

- Insights into the performance of current expert profiling systems through an extensive error analysis, plus a content analysis of feedback of experts on the generated profiles. These insights lead to recommendations for improving expertise profiling systems.

- Insights in the differences in evaluation outcomes between evaluating the two sets of ground truth released with this chapter. This will allow researchers and practitioners in the field of expertise retrieval to understand the performance of their own systems better.

Before we delve in, we give a small recap of some terminology. Expert profiles, or topical profiles, in this chapter consist of a set of knowledge areas from a thesaurus. Throughout the chapter, we will be focusing on two kinds of expert profile that we use as ground truth:

Self-selected These profiles consist of knowledge areas that experts originally selected from an alphabetical list of knowledge areas.

Judged system-generated These profiles consist of those knowledge areas that experts judged relevant from *system-generated profiles*: a ranked list of (up to) a hundred knowledge areas that we generated for them.

The rest of this chapter is structured as follows. In Section 5.1, we define the topical profiling task. Next, we describe the assessment experiment in Section 5.2: the profiling models used to generate the profiles, and the assessment interface experts used to judge these profiles. The evaluation task we study in this chapter is a multi-faceted one, giving rise to multiple research questions. We organize these in Section 5.3, where we list the questions and the methods used to answer them. We present and analyze the results of our assessment experiment in Section 5.4, followed by an analysis of benchmarking differences between two sets of relevance assessments in Section 5.5: self-selected vs. judged system-generated knowledge areas. In Section 5.6, we wrap up with a discussion, conclusion, and look-ahead.

5.1 The topical profiling task

The TU expert collection is meant to help assess topical profiling systems in the setting of a multilingual intranet of a knowledge intensive organization. One can answer the question "What topics does an expert know about?" by returning a topical profile of that expert: a record of the types of areas of skills and knowledge of that individual, and a level of proficiency in each [24]. The task consists of two steps, (1) identifying possible knowledge areas, and (2) assigning a score to each knowledge area [19]. In an enterprise search environment, there often exists a list of knowledge areas in which an organization has expertise. In our test collection this is indeed the case, therefore we focus on the second step. We assume that a list of knowledge areas $\{a_1, \ldots, a_n\}$ is given, and state the problem of assigning a score to each knowledge area (given an expert) as follows:

> What is the probability of a knowledge area (a) being part of the expert's (e) topical profile?

We approach this task as one where we have to rank knowledge areas by this probability $P(a|e)$.

In the TU expert collection, for this task systems receive the following ingredients as input:

- A query consisting of an expert ID (that is, an organization-wide unique identifier for the person).
- A collection consisting of publications, supervised student theses, course descriptions, and research descriptions crawled from the *Webwijs* system of Tilburg University. All documents are either Dutch or English. The language is known for research and course descriptions and is unknown for publications and student theses.
- Explicit associations between the expert ID and documents in the corpus.
- A thesaurus of knowledge areas. Knowledge areas are available in two languages, Dutch and English. All areas have a Dutch representation, for most of them an English translation is available as well.

Given this input, the requested system output is a ranked list of knowledge areas from the thesaurus.

We note a small subtlety concerning the language of documents in the collection. In previous work [18], systems were evaluated on the subset of knowledge areas for which both a Dutch and an English translation were available; if an expert had selected a knowledge area without an English translation, for evaluation purposes this knowledge area would be considered as non-relevant. In this work, if an expert selects a knowledge area, we consider it as relevant, regardless of whether or not it has an English translation.

5.2 The assessment experiment

We first describe the models we used to produce the *system-generated profiles*. Then, we describe the assessment interface that experts used to judge these profiles.

5.2.1 Automatically generating profiles

The system-generated profiles are generated by combining the results of eight state-of-the-art expert profiling systems in a straightforward way. In this subsection we describe the eight systems, the combination method, and we list the parameter settings we used in this chapter. The eight expertise profiling systems that we used differ in three dimensions. First, two different retrieval models are employed. Second, systems use either the Dutch or the English translations of the knowledge areas. Third, half of the systems treat knowledge areas as independent of each other, while the other half use a thesaurus of knowledge areas to capture the similarity between them. We shortly describe the models here.

The two retrieval models considered below take a generative probabilistic approach and rank knowledge areas a by the probability that they are generated by expert e: $P(a|e)$. In the first model, called Model 1 in [21], we construct a multinomial language model θ_e for each expert e over the vocabulary of terms from the documents associated with the expert. We model knowledge areas as bags of words, created from their textual labels (either Dutch or English). It is assumed that knowledge area terms t are sampled independently from this multinomial distribution, with replacement. Then, for Model 1 we have:

$$P(a|e) = P(a|\theta_e) = \prod_{t \in a} P(t|\theta_e)^{n(t,a)}, \tag{5.1}$$

where $n(t, a)$ is the number of times term t occurs in a. In estimating $P(t|\theta_e)$, we apply smoothing using collection term probabilities, with unsupervised estimation of smoothing parameters. Specifically, we employ Dirichlet smoothing and use the average representation length (i.e., the average number of terms associated with experts) as the smoothing parameter. In the second model, called Model 2 in [21], we estimate a language model θ_d for each document associated with an expert (we assume binary associations between experts and documents). Let this set of documents be D_e. We sum up the probabilities of each of these documents generating the knowledge area. The terms in a are sampled independently from each document. Then, for Model 2 we have:

$$P(a|e) = \frac{1}{|D_e|} \sum_{d \in D_e} P(a|\theta_d) = \frac{1}{|D_e|} \sum_{d \in D_e} \prod_{t \in a} P(t|\theta_d)^{n(t,a)}. \tag{5.2}$$

To estimate $P(t|\theta_d)$ we smooth using collection term probabilities as before, estimating smoothing parameters in an unsupervised way. As before, we use Dirichlet smoothing, but here we set the smoothing parameter to the average document length in the collection.

As for the language dimension, recall that knowledge areas come in two languages: $a = \{a_{\text{Dutch}}, a_{\text{English}}\}$. The Dutch retrieval models estimate $P(a_{\text{Dutch}}|e)$, the English systems estimate $P(a_{\text{English}}|e)$.

Systems that utilize the thesaurus rely on a similarity metric between a pair of knowledge areas, $\text{sim}(a, a')$. This similarity is taken to be the reciprocal of the length of the shortest path $\text{SP}(a, a')$ between a and a' in the thesaurus. If two knowledge areas are not connected, their similarity is set to zero. In addition, we use a parameter m for the maximal length of the shortest path for which we allow knowledge areas to have a non-zero

probability. Formally,

$$\mathrm{sim}(a, a') = \begin{cases} 1/\,\mathrm{SP}(a, a'), & 0 < \mathrm{SP}(a, a') \leq m \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

We describe the thesaurus graph in detail in Appendix A, Section A.4. Note that we do not distinguish between different types of relation in the graph and use all of them when searching for the shortest path. Next, we use $\mathrm{sim}(a, a')$ to measure the likelihood of seeing knowledge area a given the presence of another knowledge area a':

$$P(a|a') = \frac{\mathrm{sim}(a, a')}{\sum_{a'' \neq a'} \mathrm{sim}(a'', a')}. \tag{5.4}$$

The idea is that a knowledge area is more likely to be included in a person's expertise profile if the person is knowledgeable on related knowledge areas. This support from other knowledge areas is linearly interpolated with $P(a|e)$ using a parameter λ to obtain an updated probability estimate $P'(a|e)$:

$$P'(a|e) = \lambda P(a|e) + (1 - \lambda) \left(\sum_{a' \neq a} P(a|a') P(a'|e) \right). \tag{5.5}$$

For all systems, once $P(a|e)$ has been estimated, we rank knowledge areas according to this probability and return the top 100 knowledge areas for a given user; as we only retrieve knowledge areas a where $P(a|e) > 0$, the result list may contain fewer than 100 items.

Merging systems' outputs To arrive at the set of judged system-generated knowledge areas, we proceed as follows. We use the eight profiling systems just described (that is, {Model 1, Model 2} × {Dutch, English} × {with thesaurus, without thesaurus}) to estimate the probabilities of knowledge areas for each expert. Let us denote this as $P_i(a|e)$ ($i = \{1, \ldots, 8\}$). These probabilities are then combined linearly to obtain a combined score $P(a|e)$:

$$P(a|e) = \sum_i \alpha_i P_i(a|e). \tag{5.6}$$

Additionally, the top three knowledge areas retrieved by each profiling system receive an extra boost to ensure that they get judged. This is done by adding a sufficiently large constant C to $P(a|e)$.

Parameter settings For the systems that use the thesaurus, we let $m = 3$ (Eq. 5.3) and $\lambda = 0.6$ (Eq. 5.5). For the combination algorithm (Eq. 5.6) we let $\alpha_i = 1/8$ for all i. Further, we set $C = 10$ and, again, we only retrieve knowledge areas for which $P(a|e) > 0$.

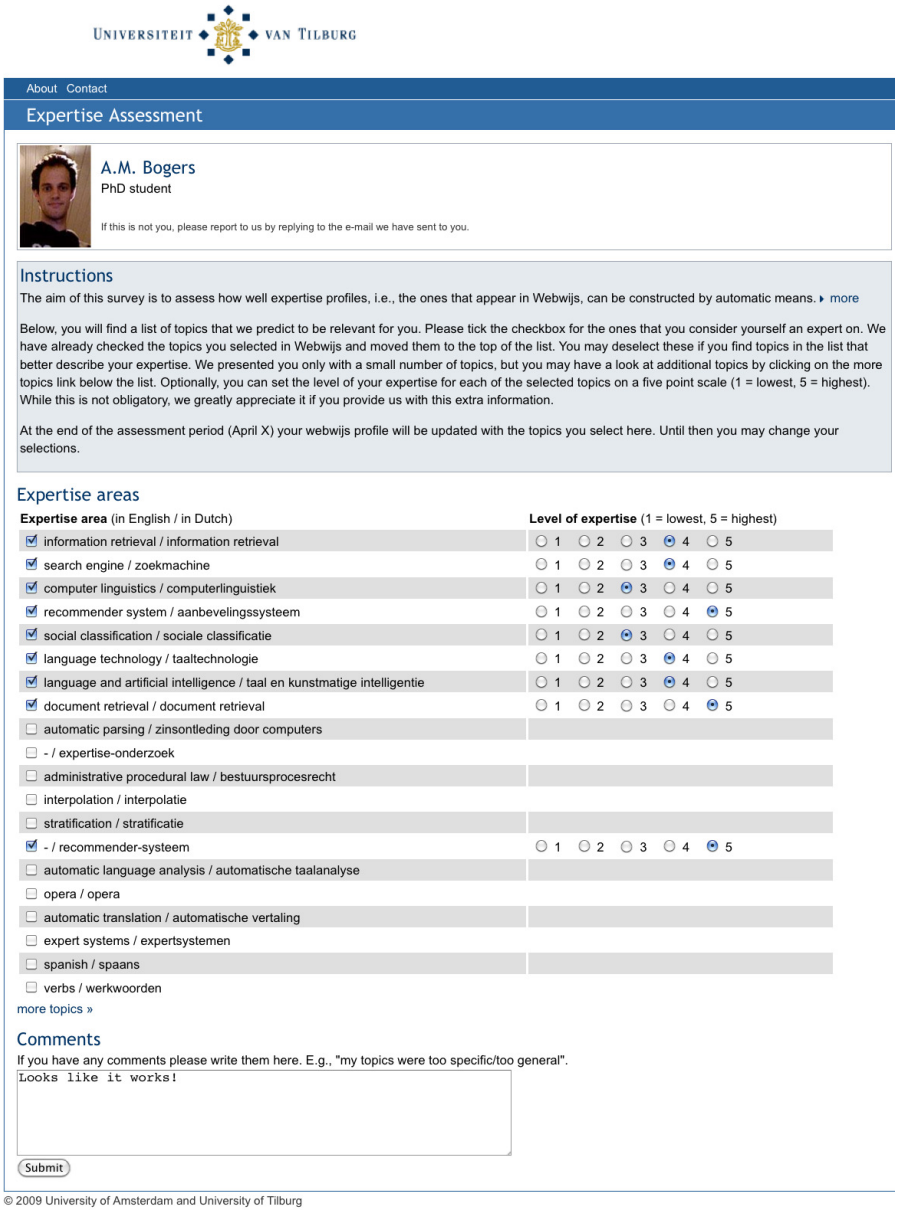


Figure 5.1: A screenshot of the interface for judging system-generated knowledge areas. At the top, instructions for the expert are given. In the middle, the expert can select knowledge areas. For selected knowledge areas, a level of expertise may be indicated. At the bottom, there is a text field for any comments the expert might have.

5.2.2 Judging the generated profiles

The assessment interface used in the assessment experiment is shown in Figure 5.1. At the top of the page, instructions for the expert are given. In the middle, the expert can indicate the knowledge areas (called "Expertise areas" in the interface) regarded as relevant by ticking them. Immediately below the top twenty knowledge areas listed by default, the expert has the option to view and assess additional knowledge areas. The expert may or may not have examined all (up to hundred) retrieved knowledge areas in the generated profile; this information was not recorded. System-generated knowledge areas that were in the original (self-selected) profile of the expert are pushed to the top of the list and are ticked by default in the interface, but the expert may deselect them, thereby judging them as non-relevant. For the ticked knowledge areas, experts have the option to indicate a level of expertise. If they do not do this, we still include these knowledge areas in the graded self-assessments, with a level of expertise of three ("somewhere in the middle"). At the bottom of the interface, experts can leave any comments they might have on the generated profile.

5.3 Research questions and methodology

In the previous two sections we have introduced the topical profiling task, and our assessment experiment. In this section, we split out RQ3 from the introduction in several more specific research questions, so as to help structure the analysis in the upcoming sections. Recall that RQ3 concerns the quality of the profiles we generated for experts in the assessment experiment, as well as evaluation outcomes when the judgments from that experiment are used. We organize the specific research questions into two corresponding subsections. The first subsection is concerned with the results of the assessment experiment. We study the completeness of the judgments gathered and the quality of the generated profiles; we answer these questions in Section 5.4. The second subsection deals with the impact of using two sets of ground truth on evaluation outcomes; we answer these questions in Section 5.5. We briefly motivate each research question and outline the methods used to answer it.

5.3.1 Results and analysis of the assessment experiment

The TU expert collection includes two sets of assessments: self-selected knowledge areas and judged system-generated knowledge areas. Our first research question concerns these two test sets of relevance assessments:

RQ3.1 *Which of the two sets of ground truth is more complete?*

> **Methods used** We construct the set of all knowledge areas that an expert judged relevant at some point in time, *either* by including it in the self-selected profile, *or* by judging it relevant in the self assessment interface. We then look which of the sets of ground truth contains more of these knowledge areas.

Remember that the judged profiles were generated by a combination of state-of-the-art systems. Our next three research questions address the informal question: "How well are we doing?"

RQ3.2 *What are the characteristics of "difficult" experts?*

For example, does the number of relevant knowledge areas correlate with performance? Does the number of documents associated with an expert matter? Is there a significant difference between mean performance over different groups of experts, for example, PhD students vs. professors?

Methods used We look for correlations by visual inspection. We group experts by their position (job title) and look for significant performance differences using Welch's two sample t-test [232]. Because we perform a number of comparisons we use an alpha value of $\alpha = 0.01$, to keep the overall Type I error under control.

RQ3.3 *What are the characteristics of "difficult" knowledge areas?*

Methods used We identify knowledge areas that are often included in experts' self-selected profiles but are rarely retrieved in the system-generated profiles. In addition, we identify knowledge areas that are often retrieved in the top ten ranks of system-generated profiles but never judged relevant by experts.

RQ3.4 *What are important aspects in the feedback experts gave on their system-generated profiles?*

Methods used In a content analysis, performed by two researchers, aspects are identified in a first pass over the data. In a second pass over the data, occurrences of these aspects are counted.

5.3.2 Self-selected vs. judged system-generated areas

Next, we analyze the differences in evaluation outcomes that arise when our two sets of relevance assessments are applied to assess expert profiling systems. Our main research question is the following:

RQ3.5 *Does using the set of judged system-generated knowledge areas lead to differences in system evaluation outcomes compared to using the self-selected knowledge areas?*

When answering this question, we consider four differences between the two sets of relevance assessments: (1) only a subset of experts has judged the system-generated knowledge areas, (2) self-selected knowledge areas that were not in the set of system-generated knowledge areas are considered non-relevant in the judged system-generated profiles, (3) experts selected new knowledge areas from the system-generated profile, and (4) experts provided a level of expertise for most judged system-generated knowledge areas. We isolate the effect of each difference by constructing five sets of ground truth (self-selected profiles, judged system-generated profiles, and three intermediate ones), which we will detail later. We consider the effect of each difference on three dimensions; these are handled as separate sub-questions.

RQ3.5a *How do the differences between the set of self-selected knowledge areas and the set of judged system-generated knowledgea areas affect absolute system scores?*

Methods used We analyze nDCG@100 performance for each of the five sets of ground truth. nDCG@100 is a metric that rewards both high precision, high recall, and—in the case of graded relevance assessments—correct ordering of relevant knowledge areas.

RQ3.5b *How do the differences between the set of self-selected knowledge areas and the set of judged system-generated knowledge areas affect system ranking?*

Methods used We analyze differences in ranking with the five sets of ground truth. Following [224], we use Kendall's tau. Like [194], we use the formula

$$\tau = \frac{P - Q}{\sqrt{(P + Q + T)(P + Q + U)}}, \tag{5.7}$$

where P is the number of concordant pairs, Q is the number of discordant pairs, T is the number of ties in the first list, and U is the number of ties in the second list. If there is a tie in at least one of the lists for a pair, the pair is neither correctly nor incorrectly ordered. When there are no ties, this formula is equivalent to the original formula as proposed by [119]. For two evaluation metrics which take into account graded relevance assessments (nDCG@100 and nDCG@10), we rank the eight systems described in Section 5.2 with each of our five sets of ground truth. We then compute Kendall's tau between each pair of these five system rankings, a total of ten comparisons per metric. For two evaluation metrics that do not take into account graded relevance (MAP, MRR), we rank the eight systems with four sets of ground truth, all of which contain only binary relevance judgments. We compute Kendall's tau between all pairs of system rankings, a total of six comparisons per metric. Thus, to answer this research question, we compute Kendall's tau for $20 + 12 = 32$ pairs of system rankings. We accept a probability of Type I error $\alpha = 0.01$ for each comparison. Then, the probability of at least one Type I error in all comparisons if they would be independent equals $1 - (1 - 0.01)^{32} = 0.28$. For eight systems, Kendall's tau has to be greater than or equal to 0.79 in order to reject the null hypothesis (two-tailed) [204].[3] To compute MAP and MRR scores, `trec_eval` was used for evaluation; for implementing nDCG, we followed [54]. We took all experts for the given test set into account during evaluation, even if systems did not retrieve any knowledge areas for them (these experts get zero score on all evaluation metrics).

RQ3.5c *How do the differences between the set of self-selected knowledge areas and the set of judged system-generated knowledge areas affect the average number of systems a system differs significantly from?*

Methods used We compare the five sets of ground truth on the basis of the number of significant differences in MAP, nDCG@100, MRR, and nDCG@10 that they detect between pairs of systems. A pair of systems differ significantly if their

[3] We observed no ties in our data, thus equation 5.7 reduces to the original Kendall's tau formula, to which the critical value 0.79 corresponds.

difference is expected to generalize to unseen queries. We use Fisher's pairwise randomization test, following [209], and set $\alpha = 0.001$. We repeat this test for five sets of ground truth, four evaluation metrics (except that we have no MAP or MRR scores for the graded relevance assessments), and all possible ($\frac{1}{2}8(8-1) = 28$) pairs of systems: a total of $18 \times 28 = 504$ comparisons. Assuming all these comparisons are independent, this means accepting a Type I error of $1 - (1 - 0.001)^{504} = 0.40$. It is no problem for the interpretation of our results if there are a few spurious rejections of the null hypothesis; we mean to give an indication of the sensitivity of each set of ground truth, i.e., the average number of systems that a system differs significantly from.

5.4 Results and analysis of the assessment experiment

In this section we report on the results of the assessment experiment defined in Section 5.2. We start with an examination of the completeness of the main tangible outcome of this experiment, the so-called judged system-generated knowledge areas. Then, we analyze the quality of the generated profiles.

5.4.1 Completeness of the two sets of ground truth for expert profiling

To answer the question how complete each set of ground truth is (**RQ3.1**) we start out with some basic descriptive statistics. Our first set of ground truth contains 761 self-selected profiles of experts who are associated with at least one document in the collection. Together, these experts selected a total of 1,662 unique knowledge areas. On average, a self-selected profile contains 6.4 knowledge areas. The second set of ground truth contains 239 judged system-generated profiles. These experts together selected a total of 1,266 unique knowledge areas. On average, a judged system-generated profile contains 8.6 knowledge areas.

In Figure 5.2, the left two histograms show the distribution of experts over their number of relevant knowledge areas for the self-selected profiles (top) and for the judged system-generated profiles (bottom). The latter distribution is shifted to the right. The histograms on the right show the distribution of knowledge areas over the profiles that include them; top-right the self-selected profiles and bottom-right the judged system-generated profiles. The latter histogram is more skewed to the left; half of the knowledge areas have been judged relevant by a single expert only.

As an aside, we now check for how many of the graded judged system-generated knowledge areas we assigned our "somewhere in the middle" value of three, because the expert judged the knowledge area relevant without indicating a level of expertise. On average, this occurred for 0.6 of the 8.8 knowledge areas in each expert's profile. We conclude that the effect of this is negligible.

Now, to quantify the completeness of each set of ground truth in a single number, we proceed as follows. Let the set of all relevant knowledge areas associated with an expert be the union of the self-selected profile and the judged system-generated profile. Then, subtract the knowledge areas that the expert deselected during the assessment interface

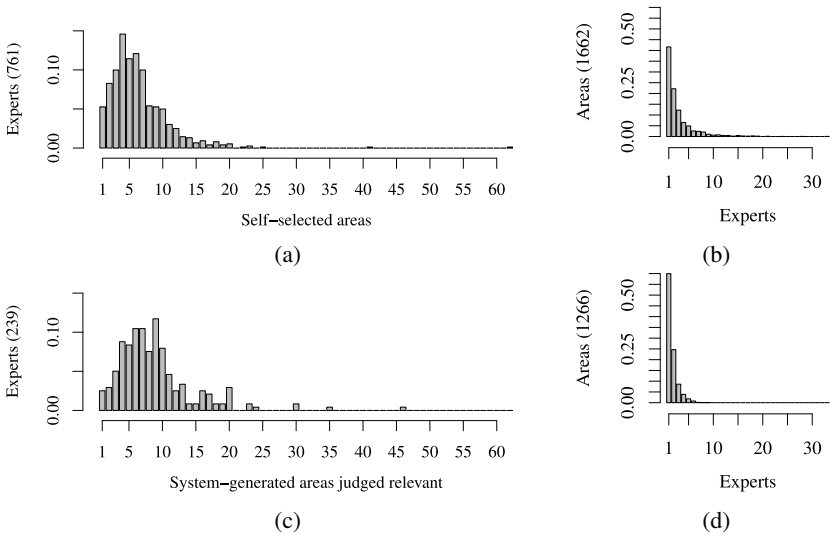


Figure 5.2: Distribution of experts over their number of relevant knowledge areas (left) and distribution of knowledge areas over the profiles that include them (right). The top graphs are based on the self-selected profiles, the bottom graphs are based on the judged system-generated knowledge areas.

Table 5.1: Average percentage of the total number of relevant knowledge areas found for experts only by the automatic expert profilers, only by the experts when they self-selected knowledge areas, or by both. The sample standard deviation over experts is shown in the "Sample std" column.

	Average	Sample std
Only found by systems	35%	24%
Only found by experts	19%	19%
Found by both	46%	24%

Table 5.2: Retrieval performance of the combined profiling algorithm on the self-selected and on the judged system-generated knowledge areas.

Ground truth	MAP	MRR	nDCG@10	nDCG@100
Self-selected (761)	0.16	0.40	0.21	0.36
Judged system-generated (239)	0.43	0.71	0.44	0.66

(on average, experts removed two percent of the knowledge areas originally included in their self-selected profiles). We divide the resulting list of knowledge areas into three categories:

Only found by systems. These knowledge areas were not in the self-selected profile, but they were in the system-generated profile and judged relevant by the experts.

Only found by experts. These knowledge areas were in the self-selected profile, but not in the system-generated profile.

Found by both These knowledge areas were in both the self-selected and system-generated profiles, and the experts did not deselect them during the assessment experiment.

In Table 5.1 we see the percentage of relevant knowledge areas that fall into each category, per profile, averaged over profiles. To answer **RQ3.1**, we find that the judged system-generated profiles are more complete. On average, a judged system-generated profile contains 81% (46% + 35%, see Table 5.1), while a self-selected profile contains only 65% (46% + 19%, see Table 5.1) of all relevant knowledge areas.

This leads to the following recommendation: because the judged system-generated profiles are more complete, we expect this set of ground truth to give a more accurate picture of system performance, even if fewer assessed expert profiles are available. We will elaborate on this when we answer **RQ3.5** later.

5.4.2 Difficult experts and difficult knowledge areas

We investigate the characteristics of "difficult" experts (**RQ3.2**) and knowledge areas (**RQ3.3**). Before we begin with the analysis at the expert and knowledge area level, we report on the overall quality of the combined profiling algorithm in Table 5.2. We measure performance against the self-selected and the judged system-generated knowledge areas, respectively. All metrics are averaged over all profiles. Note that MAP and MRR treat relevance as a binary decision and the level of expertise indicated is not taken into account. Also note that there are no graded assessments available for the self-selected profiles, hence nDCG@10 and nDCG@100 in the first row of Table 5.2 are computed using the same relevance level for all self-selected knowledge areas. We find that considerably higher absolute scores are obtained on the judged system-generated profiles than on the self-selected ones. This finding holds for all metrics. Later, when we answer **RQ3.5**, we will identify four factors that contribute to this large difference. In our detailed error analysis of the system-generated profiles that follows next, we focus

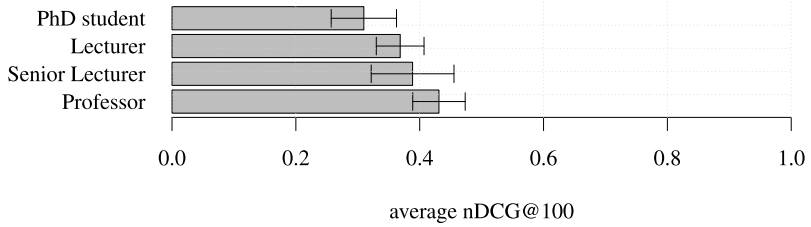


Figure 5.3: Average nDCG@100 on the self-selected profiles for the four most common positions, with 99% confidence intervals.

on nDCG@100 as it is a metric that captures the quality of the entire system-generated profile.

Difficult experts Here, we aim to find properties of experts that can explain some of the variance in performance. We use the self-selected profiles of all 761 experts; this allows us to incorporate self-selected knowledge areas that were missing from the system-generated profiles in our analysis. We investigate a number of characteristics: the number of relevant knowledge areas for the expert, the number of documents associated with experts, and the position (job title) of an expert.

First, we attempt to find a correlation between these properties and nDCG@100 performance by visual inspection. We find no correlation between the number of relevant knowledge areas selected and nDCG@100, and no correlation between the number of documents associated with an expert and nDCG@100 either. Intuitively, the relationship between the ratio of relevant knowledge areas and number of documents associated with the expert is also interesting. For example, achieving high recall may be hard when one has to find many knowledge areas in a few documents. Achieving high precision may be hard if one has to find a few knowledge areas in a lot of documents. However, between the ratio of relevant knowledge areas and number of documents associated with an expert we also find no correlation.

Next, we investigate a variable that may have different effects on performance indirectly: the position of an expert. In Figure 5.3 we see average nDCG@100 scores for the four most common positions among the 761 experts that self-selected a profile: lecturers (210), professors (168), PhD students (129), and senior lecturers (77); 99% confidence intervals on the estimated means are shown. These are calculated as $\overline{X} \pm 2.704 * \sigma/\sqrt{n}$, where σ is the sample standard deviation and n is the sample size. The value 2.704 gives a 99% confidence interval for samples larger than forty. For professors, higher nDCG scores are achieved than for lecturers and PhD students; both of these differences are significant at the $\alpha = 0.01$ level (Welch Two Sample t-test).

An intuitive explanation for the fact that it seems easier to find relevant knowledge areas for professors than for PhD students is that professors have more publications. We just noted, however, that the number of documents associated with experts does not correlate with nDCG@100 performance. But, if we look a bit deeper into the different kinds of document that can be associated with an expert, we find that it matters whether or not an expert has a research description. Experts can have no research description,

only a Dutch one, only an English one, or both a Dutch and an English one. We find that for the 282 experts without a research description we achieve significantly lower average nDCG@100 performance than for the 479 experts who have at least one (Welch Two Sample t-test, $p < 0.001$). The difference, in absolute terms, is also substantial: 0.39 vs. 0.30 for experts with and without a research description, respectively. It is not surprising that these research descriptions are important; they constitute a concise summary of a person's qualifications and expertise, written by the expert himself/herself. Of the professors, 73% have a research description against 53% of the PhD students, so this property explains part of the difference in performance between these two groups.

Missing knowledge areas Next we provide insights into relevant knowledge areas that we failed to retrieve in the system-generated profiles. To capture the fact that some knowledge areas are missing in more system-generated profiles than other knowledge areas, we define recall and precision measures for knowledge areas in a very straightforward and intuitive way. We say that knowledge areas that are missing in many system-generated profiles are *difficult*: they have low recall. Letting O_a be the set of self-selected profiles that contain knowledge area a and G_a the set of system-generated profiles that contain a, we can define recall as:

$$R(a) = \frac{|O_a \cap G_a|}{|O_a|}. \tag{5.8}$$

We are interested in knowledge areas a with low recall $R(a)$ here. Given equal recall, the more difficult knowledge areas are those that have lower precision:

$$P(a) = \frac{|O_a \cap G_a|}{|G_a|}. \tag{5.9}$$

We discard knowledge areas from our error analysis for which we cannot compute reliable recall and precision values. First, for computing recall, we exclude knowledge areas that are not in any self-selected profile. Also, we discard knowledge areas that are present in less than five self-selected profiles; the reason for doing so is to avoid large differences in recall for knowledge areas that may occur only by chance. Second, we cannot compute precision for knowledge areas that were not retrieved for any expert; this applies for only 14 (out of the 2,509) knowledge areas, 8 of which were also not in any self-selected profile. In this error analysis, then, we analyze only 361 of all 2,509 knowledge areas.

Figure 5.4 displays the abovementioned 361 knowledge areas on a precision-recall plot. We added some jitter to the points for visualization purposes. In the bottom left corner of the figure, there are 17 knowledge areas with zero recall and precision. We list these "problematic" knowledge areas in Table 5.3, ranked by the number of system-generated profiles that contain them. This may be seen as an ordering by difficulty, where we consider knowledge areas that are more often retrieved incorrectly to be more difficult. In this list, we find some very general knowledge areas like *computer science* and *language*; there are also very specific knowledge areas like *dutch for foreigners* and *income distribution*. Looking further down to the knowledge areas that are retrieved less often, we see many have no English translation. The English language profiling systems will never contribute these knowledge areas.

Table 5.3: Problematic knowledge areas in terms of precision and recall. For each knowledge area, we list the number of system-generated profiles where it is missing ("Missing") and where it is (incorrectly) retrieved ("Retrieved"). In a small number of cases, experts added these knowledge areas to their profile during the assessment experiment ("Added").

Dutch	English	Missing	Retrieved	Added
informatica	computer science	8	71	0
inkomensverdeling	income distribution	5	66	1
taal	language	5	61	2
nederlands voor buitenlanders	dutch for foreigners	5	55	0
automatisering	automation	5	49	1
culturele verscheidenheid	cultural diversity	5	41	2
e-government	e-government	6	39	0
evaluatie onderzoek	-	8	38	0
bedrijfsbeleid en -strategie	corporate policy and strategy	6	38	2
welzijn	well-being	6	26	1
ontwikkelingsvraagstukken	-	5	23	0
methoden en technieken, sociaal-wetenschappelijke	-	8	22	0
programmeren voor internet	-	7	20	0
beleidsonderzoek	-	8	18	1
cognitieve informatieverwerking	-	6	12	0
Kant, Immanuel (1724-1804)	Kant, Immanuel (1724-1804)	5	9	0
cultuurparticipatie	-	5	9	0

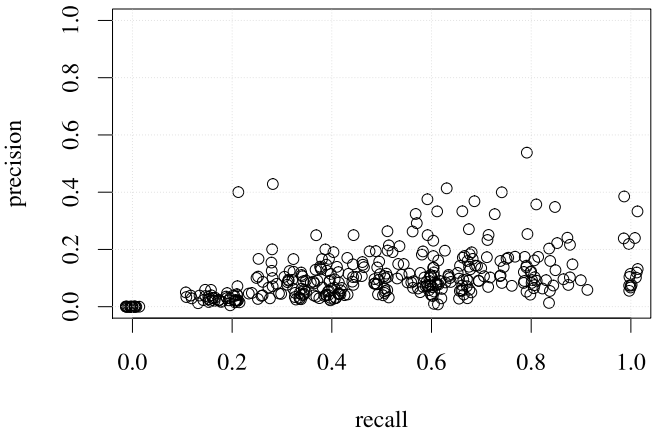


Figure 5.4: Precision and recall of 361 knowledge areas that were in at least five self-selected profiles.

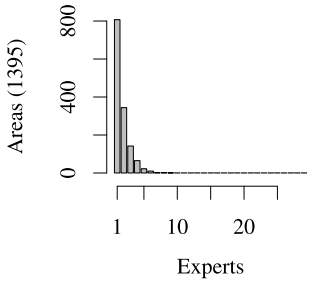


Figure 5.5: Knowledge areas over the number of experts for whom they were retrieved in the top ten.

Knowledge areas often retrieved but never selected Here, we are interested in finding knowledge areas that are ranked high (say, in the top 10) for many experts and yet are always judged non-relevant by these experts. For this analysis, we limit ourselves to the 239 system-generated profiles that have been judged in the assessment experiment.

In Figure 5.5 we show the distribution of knowledge areas over the number of experts they were retrieved for in the top ten. Note that this distribution resembles the distribution of knowledge areas over the number of experts that judged them relevant in the assessment experiment (Figure 5.2d); this is a good property to have. We see that 1395 knowledge areas are retrieved for at least one expert in the top ten, this is about 60% of all knowledge areas. Of these 1395 knowledge areas, 773 were not judged relevant by any of the experts for whom they were retrieved in the top ten. We order these areas by decreasing number of system-generated profiles in which they were incorrectly included in the top ten, and show the top twenty in Table 5.4. Most of these knowledge areas appear to be quite specific.

Summing up, the main findings of this subsection are as follows. With regard to char-

Table 5.4: Knowledge areas shown in the top ten, but never selected. The list is ordered by the number of times the knowledge area was retrieved in the top ten.

Dutch	English	In top 10
alfabetisering in nederlands als tweede taal nt2	–	9
godsdienstpedagogiek	pedagogy of religion	8
optietheorie	option pricing	8
behendigheidsspelen	dexterity games	7
sociolingustiek	sociolinguistics	7
asset liability management	asset liability management	6
productiemanagement	production management	6
dienstenmarketing	services marketing	6
werkloosheidsduur	–	6
mediarecht	–	6
cognitieve lingustiek	cognitive linguistics	6
handelsmerken	trademarks	6
organisatiebewustzijn	–	6
belastingrecht	tax law	5
bestuursrecht	administrative law	5
geldwezen	money	5
instructieve teksten	instructive texts	5
oefenrechtbank	moot court	5
onderwijs- en opleidingspsychologie	educational and training psychology	5
openbaar bestuur	public administration	5

acteristics of difficult experts (**RQ3.2**): (1) difficulty is *not* correlated simply with the number of relevant knowledge areas or with the number of documents associated with experts; (2) performance is significantly higher for experts who have a research description (in Dutch and/or in English). With regard to characteristics of difficult knowledge areas (**RQ3.3**), we find that knowledge areas that we often fail to retrieve (see Table 5.3) (1) often lack an English translation, making them impossible to find for our English language profiling algorithms; (2) can be both general and specific knowledge areas. Knowledge areas that we often retrieve in the top ten while they were not judged relevant by experts (see Table 5.4) (1) appear to be quite specific knowledge areas; (2) sometimes lack an English translation.

5.4.3 A content analysis of expert feedback

We now address our research question about important aspects in the feedback experts gave by carrying out a content analysis (**RQ3.4**). During the assessment experiment, 91 experts left comments in the text area field at the bottom of the assessment interface. These comments were coded based on a coding scheme developed by a first pass over the data. The coding was performed by two of the authors of the paper on which this chapter is based [29]. A statement could be assigned multiple aspects. After all aspect types were identified, the participants' comments were coded in a second pass over the data. Upon completion, the two annotators resolved differences through discussion. We report on inter-annotator agreement after discussion, reflecting cases where there remained a difference in opinion. We use two measures of inter-annotator agreement:

Micro-averaged inter-annotator agreement The number of times both annotators coded a comment with the same aspect, divided by the total number of codings: $146/150 \approx 0.97$.

Macro-averaged inter-annotator agreement For each aspect inter-annotator agreement is calculated: the number of times both annotators coded a comment with this aspect, divided by the total number of codings with this aspect. Then the average of these aspect inter-annotator agreements is calculated: ≈ 0.98.

Both measures show very high inter-annotator agreement.

Table 5.5 lists all aspects with the count and the percentage of the comments in which they appeared. First, we address the most common aspects in the experts' comments about the system-generated knowledge areas. The most common complaint is that a key knowledge area is missing. These missing knowledge areas were in *Webwijs*, and consequently they were in the input list from which the retrieval systems select knowledge areas. This means the profiling algorithm is perceived to have insufficient recall. The second most frequently mentioned aspect is a request to add a new knowledge area to *Webwijs*. These do not reflect a failure on the part of our profiling algorithms. Rather, it is a request to the administrators of *Webwijs* to expand the thesaurus of knowledge areas. The third most common aspect is that the profile consists entirely of non-relevant knowledge areas. This is a complaint about low precision. If there are relevant knowledge areas for the expert in the thesaurus, it also implies low recall.

Next, we examine the four categories of aspects in Table 5.5. Looking at the aspects relating to the quality of recommendations, we see that experts tend to be dissatisfied.

Table 5.5: Results of a content analysis of expert feedback.

Aspects	Count	Percentage
Quality of recommendations		
Excellent recommendations	7	7.9
Partially correct	10	11.2
All nonsense recommendations	15	16.9
Comments about individual knowledge areas		
Too much focused on one knowledge area	3	3.4
Missing key knowledge area (present in *Webwijs*, but not recommended)	32	36.0
Mix-up between different fields	2	2.2
One single nonsense recommendation	8	9.0
Comments about list as a whole		
Big overlap in recommended knowledge areas	10	11.2
Lack of consistency in recommended knowledge areas	1	1.1
Knowledge areas are too specific	4	4.5
Knowledge areas are too broad/general	10	11.2
No clear ordering of list	2	2.2
Upper limit of 10 knowledge areas	2	2.2
Knowledge areas taken from only one source (i.e., publications vs. theses)	5	5.6
Administrative comments		
Request to add expertise term to *Webwijs* itself.	20	22.5
Missing *Webwijs* terms due to time difference between dump and survey	1	1.1
Complaint about incorrect or outdated *Webwijs* metadata	5	5.6
Rating expertise seen as ineffective	1	1.1
Complaint about spelling or translation of *Webwijs* knowledge areas	12	13.5

We cannot directly relate this to average performance over all experts, because we do not know the reasons why one expert chooses to leave a comment while another decides not to. For some, dissatisfaction with the result list may be a motivation to comment, while others might find the results satisfactory enough and see no reason to add further feedback.

From comments about the lists as a whole, one of the main complaints is that there is much overlap in recommended knowledge areas. On the one hand, this means that our algorithm finds "near misses:" knowledge areas that are not relevant, but are very similar to relevant knowledge areas. On the other hand, it is clear that retrieving multiple very similar knowledge areas is not appreciated. De Rijke et al. [68] propose a new metric that simultaneously rewards near misses and penalizes redundancy in a result list; we leave it as future work to actually implement and use this metric.

A second main complaint about the list as a whole is that results are too general. Interestingly, the opposite complaint also occurs: results are too specific. De Rijke et al. [68] suggest that experts higher up in the organization tend to prefer more specific knowledge areas, while teachers and research assistants prefer broader terms. In our comments, complaints about a result list being too specific come from a professor, a lecturer, a researcher, and someone with an unknown function description. Complaints about the generated list being too general come from professors (5), senior lecturers (2), lecturers (2), and someone with no function description: mostly from senior staff.

In the administrative comments, it is interesting to note that almost no experts view rating knowledge areas as ineffective or unnecessary. Of course, experts were not explicitly asked about what they thought of rating knowledge areas, but still this was a big difference between our assessment interface and the *Webwijs* interface experts originally used for selecting knowledge areas.

To answer (**RQ3.4**), the main aspects in the feedback of experts are (1) missing a key knowledge area in the generated profile (36%); (2) only non-relevant knowledge areas in the profile (16.9%); (3) redundancy in the generated profiles (11.2%); (4) knowledge areas being too general (11.2%).

In sum, it is clear that there is room for improvement in terms of both precision and recall. Since experts complain about redundancy in their profiles, in future work the diversity of profiles deserves attention. The desired level of specificity/generality is to a large extent a matter of personal preference. There are more complaints, however, about knowledge areas being too general; this is an indication that algorithms in overall may score better by preferring specific knowledge areas.

5.5 Self-selected vs. judged system-generated areas

In this section we look at another aspect of the TU expert collection as a measurement device. We study the differences between evaluating profiling systems with the self-selected knowledge areas and evaluating them with the judged system-generated knowledge areas (**RQ3.5**). The differences between the two types of assessment are isolated using five sets of ground truth, which we detail in the first subsection. In the remaining subsections we study the changes between evaluating with the two types of assessment along three dimensions: absolute system scores (**RQ3.5a**), system ranking (**RQ3.5b**), and the aver-

age number of systems a system performs significantly different from (**RQ3.5c**). All this is meant to help understand the merits of the TU expert collection.

5.5.1 Five sets of assessments

In Section 5.4, we have studied the differences between self-selected and judged system-generated profiles; the corresponding ground truth that was used for evaluation (cf. Table 5.2) will be referred to as GT1 and GT5, respectively, throughout this section. These two sets of assessments differ in a number of dimensions: the number of profiles evaluated, the knowledge areas considered relevant within the profiles, and the grades of relevance. To help better understand the impact these differences might have on system evaluation, we introduce three more intermediate sets of assessments (GT2, ..., GT4). Next, we briefly discuss each of the five sets.

GT1: Self-selected profiles. The self-selected profiles of all experts for whom we generated a profile. Experts had previously selected these knowledge areas in the *Webwijs* system of Tilburg University; this set contains 761 experts.

GT2: Self-selected profiles of participants in assessment experiment. The self-selected profiles of only those experts who completed the assessment experiment. To be able to realize all subsequent evaluation conditions with the same set of experts, we limit this set of experts to those:

- who completed the assessment experiment, selecting (or keeping) at least one knowledge area;
- who had a non-empty self-selected profile;
- for whom at least one of the knowledge areas in their self-selected profile was retrieved by the automatic profiling systems. (This condition is required to be able to analyze, for the same set of experts, what evaluation differences there are when we evaluate only on the pooled subsets of their self-selected profiles.)

As noted in Section 5.4.1, this set comprises of 239 experts; for ease of reference, we sometimes refer to them as "our assessors."

GT3: Pooled subsets of self-selected profiles. For each self-selected profile of an assessor, we only use knowledge areas that were in the system-generated profile. This means that knowledge areas that are not in the system-generated profile are treated as non-relevant.

GT4: Judged system-generated profiles (binary). The knowledge areas judged relevant during the assessment experiment. We only consider binary relevance; if a knowledge area was selected it is considered as relevant, otherwise it is taken to be non-relevant.

GT5: Judged system-generated profiles (graded). The same as GT4, but now with graded relevance. Experts could optionally indicate their level of expertise on each knowledge area they selected. Recall that when experts have selected a knowledge

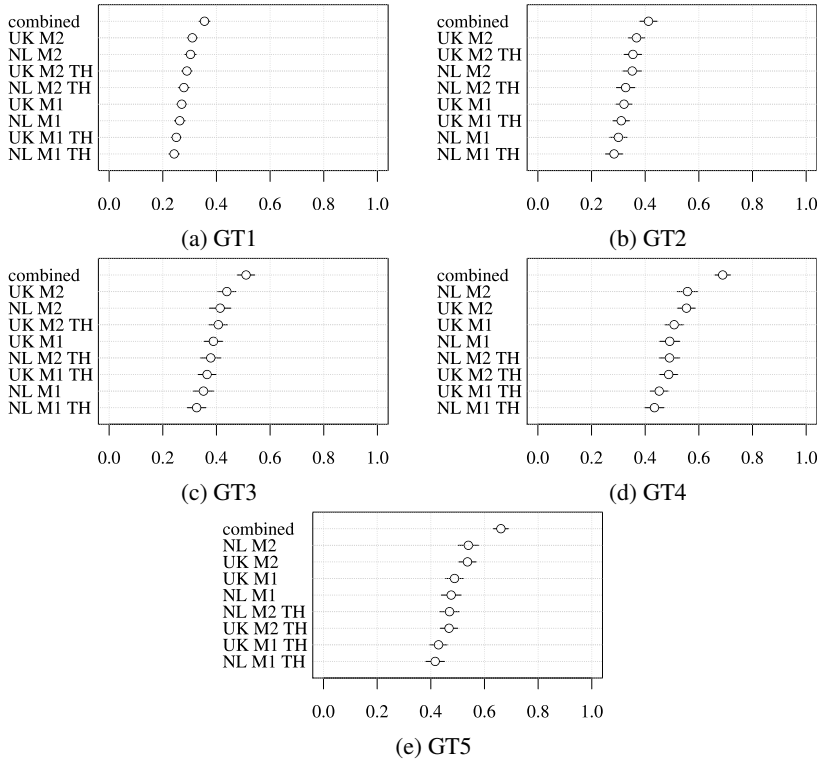


Figure 5.6: Average nDCG@100 for each profiling system defined in Section 5.2, with 99% confidence intervals, for the five sets of assessments GT1, ..., GT5 examined in this section.

area but indicated no level, we assume they would have indicated a level "somewhere in the middle": level three out of five.

In the next subsection we go through these five sets of ground truth, looking only at nDCG@100. We will show how absolute system scores change from set to set.

5.5.2 Contrasting GT1–GT5

Previously, in our error analysis of system-generated profiles, we have seen that the combined profiling algorithm achieved higher scores on GT5 than on GT1. Here, we investigate the influence of the four differences between GT1 and GT5, in a step-by-step fashion, by considering each of GT1, ..., GT5 and evaluating our profiling using those sets of assessments. To remain focused, we evaluate all systems with nDCG@100; nDCG is a well understood metric that can be used with both binary and graded relevance assessments.

In Figure 5.6a we show the nDCG@100 scores obtained with GT1, for all systems.[4]

[4]We use the following convention to name the profiling systems defined in Section 5.2: XYZ, where

Confidence intervals on the means $\overline{X}$ are shown. These are based on the assumption that nDCG@100 scores are normally distributed. We show a 99% confidence interval, calculated as $\overline{X} \pm 2.576\hat{\sigma}/\sqrt{n}$, where $\hat{\sigma}$ is the sample standard deviation and n is the sample size. The scores of individual systems are close to each other. Model 2 outperforms Model 1, the English language systems tend to perform marginally better than their Dutch counterparts, and using the thesaurus does not appear to offer any benefits. The combined algorithm, which generated the profile shown to our assessors, outperforms all individual systems.

In Figure 5.6b we plot the results based on GT2, on the self-selected profiles of assessors only. Confidence intervals are larger here, this is because the sample size is smaller. System scores are also a bit higher across the board.

In Figure 5.6c we show the results obtained using GT3, i.e., only using knowledge areas that the assessors may have seen during the assessment experiment. Recall that initially experts see only the top twenty of the generated profile, but they can see up to a hundred knowledge areas if they request more results. When we view the combined algorithm that generated the profile as a pooling algorithm, we can study the effect of pooling here. Absolute scores again increase for all systems. This is not surprising; un-pooled knowledge areas are hard for all systems and regarding them as non-relevant reduces the problem difficulty. However, we can also see that relative system ranking hardly changes.

In Figure 5.6d we evaluate profiling with GT4, i.e., with the knowledge areas selected during the assessment experiment. We see a substantial performance increase in absolute scores for all systems, compared to evaluating with only the pooled knowledge areas from the original profiles. This increase is caused by knowledge areas experts chose to *add* to their profiles. It is an indication that the original self-selected profiles were often incomplete, and systems are actually doing a better job than evaluating with the self-selected profiles would suggest. We also see changes in system rankings here that are a bit stronger than between other sets of ground truth. Systems that employ Model 2 clearly outperform the ones that work with Model 1. Also, systems without the thesaurus are distinctly better than those with it. No language is preferred over the other.

For selected knowledge areas experts could optionally indicate a level of expertise on a scale of one to five. In cases where they did not indicate a level of expertise for a selected knowledge area we assigned a default level (3). In Figure 5.6e we see how the move from binary to multiple levels of relevance changes nDCG@100: absolute scores slightly decrease for all systems. This means that all systems retrieve knowledge areas in a suboptimal order. The relative ordering of systems, however, remains unchanged.

Answering **RQ3.5a**: (1) Scores obtained on our assessors only are higher than on all experts. (2) Scores on only the pooled knowledge areas are higher than on the complete self-selected profiles; this is because self-selected knowledge areas that were un-pooled are apparently hard for these systems, and regarding them as non-relevant reduces problem difficulty. (3) Scores on the binary judged system-generated knowledge areas are substantially higher than on the self-selected knowledge areas; this is an indication that the system-generated profiles were better than the original self-selected knowledge areas would give them credit for. (4) When we consider multiple relevance levels for assess-

X ∈ {NL, UK}, Y ∈ {M1, M2}, and Z ∈ { , TH}.

Table 5.6: Kendall's tau between system rankings on two sets of assessments with MAP (lower triangle) and average nDCG@100 (upper triangle).

	GT1	GT2	GT3	GT4	GT5
GT1 Self-selected profiles of all experts	–	0.86	0.86	0.57	0.57
GT2 Self-selected profiles of assessors	0.79	–	0.86	0.43	0.43
GT3 Pooled subsets of self-selected profiles	0.71	0.93	–	0.57	0.57
GT4 Judged system-generated profiles (binary)	0.57	0.50	0.57	–	1.00
GT5 Judged system-generated profiles (graded)	–	–	–	–	–

Table 5.7: Kendall's tau between system rankings on two sets of assessments with MRR (lower triangle) and average nDCG@10 (upper triangle).

	GT1	GT2	GT3	GT4	GT5
GT1 Self-selected profiles of all experts	–	0.79	0.79	0.93	0.86
GT2 Self-selected profiles of assessors	0.71	–	1.00	0.71	0.79
GT3 Pooled subsets of self-selected profiles	0.71	1.00	–	0.71	0.79
GT4 Judged system-generated profiles (binary)	0.93	0.79	0.79	–	0.93
GT5 Judged system-generated profiles (graded)	–	–	–	–	–

ment, absolute performance decreases a bit across the board, showing that to some extent all systems rank knowledge areas suboptimally.

5.5.3 Changes in system ranking

We take a closer look at differences between GT1–GT5 and analyze whether, and if so, how, they rank profiling systems differently. In the previous section we observed changes in system ranking in terms of nDCG@100, due to knowledge areas experts had added to their self-selected profile during the assessment experiment. In this section we study how system rankings change on each set of GT1, ..., GT5 for other metrics too: MAP, MRR, and nDCG@10. We exclude the combined algorithm from our analysis here, because it produced the actual rankings that experts judged (experts are likely biased by the order in which suggestions were presented to them).

Tables 5.6 and 5.7 report Kendall's tau for four evaluation metrics, computed between all pairs of sets of assessments. (The last rows of the tables are empty, since the MAP and MRR measures only consider binary relevance.) Table 5.6 shows the tau values for MAP and nDCG@100 in the lower and upper triangles, respectively. Both of these evaluation metrics capture precision as well as recall. Because all systems retrieve at most 100 documents, they both consider the complete list of results retrieved.

Let us consider the five sets of assessments GT1, ..., GT5 for MAP and nDCG@10 and walk through Table 5.6. First, system ranking correlation between evaluating with the self-selected profiles of all 761 experts (GT1) and evaluating with the self-selected profiles of only the 239 assessors (GT2) is reasonable for both MAP and nDCG@100. Compared to GT2, considering un-pooled knowledge areas as non-relevant (GT3) ranks systems similarly for both metrics as well. While we saw in the previous section that

absolute scores increased substantially when un-pooled knowledge areas are assumed to be non-relevant, this has little effect on relative performance. The next step is including knowledge areas added during the assessment experiment (GT4). This does change the picture for both MAP and nDCG@100. For neither of the two metrics we can reject the null hypothesis which states that there is no monotone relationship between the two rankings. Finally, taking into account the level of expertise (GT5) does not affect system ranking at all.

Next, we look at Table 5.7 and two measures that focus on the top ranks: MRR and nDCG@10. Table 5.7 shows Kendall's tau values for MRR and nDCG@10 in the lower and upper triangles, respectively. Again, we step through the four changes that lead from GT1 to GT5. When evaluating with self-selected profiles from assessors only (GT2) instead of from all experts (GT1), rankings change a bit for both metrics. For MRR, there is no significant correlation. Regarding un-pooled knowledge areas as non-relevant (GT3) does not affect system ranking at all for these two metrics. Using the judged system-generated knowledge areas (GT4) instead of the self-selected knowledge areas changes the ranking a bit, again; for nDCG@10 there is no significant correlation. Finally, we find that the level of expertise (GT5) leads to only minor changes in system ranking for nDCG@10.

In answer to **RQ3.5b**, our findings are: (1) Comparing GT1 with GT2, the only difference being that GT2 evaluates with a subset of experts, we see that system rankings change a bit; nevertheless, for all metrics but nDCG@10 we can reject the null hypothesis, which states that system rankings do not correlate. (2) Regarding un-pooled knowledge areas as non-relevant hardly affects system rankings for the eight systems that contributed to the pool. Kendall tau values are high, ranging from 0.86 (nDCG@100) to 0.93 (MAP) to 1.00 (nDCG@10 and MRR) when comparing GT2 with GT3. (3) The knowledge areas that experts added to their self-selected profile during the assessment experiment have an effect on system rankings. When comparing GT1–3 with GT4, in all but two cases, we cannot reject the null hypothesis stating that there is no monotone relationship between system rankings obtained when evaluating with the self-selected vs. judged system-generated profiles. (4) Comparing GT4 with GT5, we see that taking into account the level of expertise does not change system ranking for nDCG@10 or nDCG@100.

5.5.4 Pairwise significant differences

The final analysis we conduct concerns a high-level perspective: the sensitivity of our evaluation methodology. The measurement that serves as a rough estimate here is the average number of systems each system differs from; we compute this for each of the five sets of assessments and for four different metrics. We use Fisher's pairwise randomization test with $\alpha = 0.001$ to establish the average number of systems each system differs from in each condition. Table 5.8 lists these averages for MAP, MRR, nDCG@10, and nDCG@100. We start out with original profiles of all experts (GT1). If we limit ourselves to the 239 self assessors (GT2), we see that the number of significant differences detected decreases for all four metrics. This is expected as the power of significance tests decreases with sample size. If we disregard non-pooled knowledge areas (GT3) we do not witness much change in the number of significant differences. Regarding non-pooled

Table 5.8: Average number of systems each system differs significantly from, using Fisher's pairwise randomization test with $\alpha = 0.001$.

		MAP	MRR	nDCG@10	nDCG@100
Self-selected profiles	GT1 all experts	3.75	4.50	4.25	4.75
	GT2 assessors	2.75	2.25	2.75	3.00
	GT3 pooled subsets	2.75	2.25	3.25	2.75
Judged system-generated	GT4 binary	4.00	2.75	4.00	3.50
	GT5 graded	–	–	4.25	4.00

knowledge areas as non-relevant does not change our insights about the relative performance of the profiling systems being examined. Comparing the self-selected profiles with the judged system-generated profiles (interpreted as binary judgments, GT4), there is a noticeable difference in the number of significant pairwise differences detected. For MAP and nDCG@10 we are roughly at the same level as for the self-selected profiles of all experts (GT1). If we use graded relevance (GT5), there is a slight increase for nDCG@10 and nDCG@100.

Answering **RQ3.5c**, we find that: (1) smaller test collections (fewer experts) implies fewer significant differences; (2) regarding un-pooled knowledge areas as non-relevant does not have much effect on sensitivity; (3) knowledge areas that experts added to their profile during the assessment experiment lead to more detected significant differences; (4) taking into account the level of relevance can lead to some further increase in sensitivity.

The two main findings for **RQ3.5** overall are: (1) GT4 (the judged system-generated knowledge areas, with binary relevance) is different from GT3, with much higher absolute scores, a different system ranking, and more detected pairwise significant differences between systems. (2) For our eight systems, regarding the un-pooled knowledge areas as non-relevant does lead to higher absolute scores, but not to different system rankings or more detected pairwise significance differences.

Our findings lead to the following recommendations for researchers who would like to evaluate their expert profiling systems on the TU expert collection. Since the judged system-generated profiles are more complete (Section 5.4.1), they form the preferred ground truth for expert profiling. Compared to evaluating on the self-selected profiles, system ranking can change. Taking into account the level of expertise is useful because it does have an effect on absolute scores, even if it is not expected to lead to very different insights into relative system performance. If researchers are concerned that their methods are not rewarded for some retrieved knowledge areas that were not in the system-generated profiles, we recommend to repeat our analysis contrasting GT2 and GT3; this comparison allows for studying that factor in isolation.

5.6 Discussion and conclusions

In this section we provide a recap of our findings with regard to RQ3 as formulated in the introduction. We released, described and analyzed the TU expert collection for assessing automatic expert profiling systems. The collection building process was detailed and we provided a critical assessment and analysis of this test collection. We started with an analysis of the completeness of self-selected vs. judged system-generated knowledge areas as ground truth, an error analysis of system-generated expertise profiles, and a content analysis of feedback given by experts on system-generated expertise profiles. Together, this analysis covers the first part of RQ3, which concerns the quality of the system-generated knowledge areas. Then we took a step back and contrasted findings by benchmarking eight state-of-the-art expert profiling systems with the two different sets of ground truth. This analysis answers the second part of RQ3 about evaluation outcomes using the judged system-generated knowledge areas. In Section 5.3 we reformulated RQ3 in terms of five more specific research questions. Here, we do not repeat all answers to these detailed questions, but instead list the main findings for each, and with these main findings we give recommendations for the development and evaluation of expert profiling systems. After that, we discuss directions for future work for which the TU expert collection could be of use.

5.6.1 Main findings with recommendations

In this subsection we repeat our research questions and list the main findings and recommendations.

RQ3.1 *Which of the two sets of ground truth is more complete?*

Judged system-generated profiles are more complete, on average. When we regard as relevant for an expert the union of knowledge areas in the self-selected profile and the judged system-generated profile (minus those knowledge areas that were judged non-relevant), the average judged system-generated profile contains 81%, the self-selected profile 65% of all relevant knowledge areas.

Recommendation To evaluate expert profiling systems, it is preferred to use the system-generated profiles because they are more complete.

RQ3.2 *What are the characteristics of "difficult" experts?*

Our main finding here is that experts that do not have a research description are significantly harder to profile accurately than experts that do.

Recommendation Have experts in a knowledge-intensive organization maintain an up-to-date natural language description of their own expertise to facilitate better expert profiling.

RQ3.3 *What are the characteristics of "difficult" knowledge areas?*

Our main finding here is that knowledge areas that lack an English translation are harder to retrieve, and they are also among those knowledge areas that are most often retrieved without being relevant.

Recommendation In a multilingual setting, maintain a complete translation of the list of knowledge areas in all languages to facilitate better expert profiling.

RQ3.4 *What are important aspects in the feedback experts gave on their system-generated profiles?*

Experts mainly complain about missing a key knowledge area, about generated profiles consisting of all nonsense knowledge areas, redundancy in the generated profiles, and about retrieved knowledge areas being too general.

Recommendation An interesting direction for future work is to go beyond ranking knowledge areas for an expert and to build coherent, complete, concise, diverse expertise profiles at the right level of specificity.

RQ3.5 *Does the set of judged system-generated knowledge areas lead to differences in system evaluation outcomes compared to using the self-selected knowledge areas?*

We found that the knowledge areas experts added to their self-selected profile by judging them relevant do have an influence on system ranking, and we observe more significant differences between systems compared to evaluating with the self-selected profiles of these experts. Even though in the judged system-generated profiles some of the knowledge areas that experts had self-selected before are missing, this hardly affects the relative ranking of our eight systems.

Recommendation It is preferred to use the judged system-generated profiles for benchmarking expert profiling systems because these profiles are more complete. The missing knowledge areas from the self-selected knowledge areas hardly had an effect on relative performance from our systems, but if researchers wish to evaluate new and very different systems, we recommend to repeat our analysis contrasting the sets of ground truth GT2 and GT3 (we release all sets of ground truth used in this chapter).

5.6.2 Directions for future work

One conclusion that can be drawn from the error analysis and the content analysis of expert feedback is that there is still much room for improvement in the area of expertise retrieval. In addition to improving system performance on the task we studied in this chapter, we believe there are interesting possibilities to study tasks that differ subtly from it.

Expert profiling and expert finding The expert profiling task is closely related to the expert finding task. Very similar algorithms may be used to approach the expert finding and profiling tasks; in both cases the extent to which an expert and a knowledge area are associated have to be estimated [24]. It has also been shown that expert finding algorithms can benefit from the output of expert profiling algorithms [19]. In addition to benchmarking expert profiling systems, the TU expert collection can also be used for benchmarking expert finding systems. In this case, using the self-selected profiles would suit fine. Since the self-selected profiles are available for more experts, the number of relevant experts per knowledge area is somewhat larger in them. In addition, the graded

relevance assessments were collected with the task of expert profiling in mind. Relevance levels are not guaranteed to be comparable across experts.

Diversity, redundancy and specificity The evaluation metrics used in this chapter treat the relevance of knowledge areas in the ranked list independent from each other. We have seen that experts complained about redundancy in their generated profiles; something, our evaluation metrics cannot capture. De Rijke et al. [68] propose a metric that would reward diversity and near misses in a topical profile. Benchmarking with metrics like this is an interesting direction for future work. Experts complained about profiles being too general, and a few about profiles being too specific. One step to accommodating adjusting the level of specificity in expertise profiles would be to require systems to organize knowledge areas in a hierarchy. A follow-up step could be to develop an assessment interface where experts can judge: (1) if grouped knowledge areas are indeed similar; (2) if hierarchical orderings are correct; (3) if retrieved knowledge areas are of the right specificity. The curated thesaurus that comes with the TU expert collection can be of help for work in this direction.

A learning assessment interface The retrieval systems we evaluated in this chapter did not use knowledge areas that had already been self-selected by experts as evidence. This means our findings on their relative performance generalize to other settings where such self-selected ground truth is not available. Still, in settings where such ground truth is available, using it to locate additional relevant items is a powerful way of expanding a set of relevant items fast, with limited annotation effort. An assessment interface that would be fed by a learning retrieval model and would be continuously available for experts to update their profile is an interesting direction for future work.

In this long chapter we have taken a thorough look at evaluating expert profiling algorithms with the TU expert collection. We have used techniques commonly used to evaluate evaluation methodology, as discussed in Chapter 2, Section 2.2, including using Kendall's tau to compare system rankings produced by different evaluation methodologies, and the use of significance tests to say something about the sensitivity of an evaluation methodology. The TU test collection was created manually, and a significant amount of effort of employees of Tilburg University went into the assessment experiment. In the next two chapters, we create Cranfield style test collections automatically, referring to them as pseudo test collections (PTCs). In the first of these two chapters, we will consider benchmarking on a PTC, comparing system rankings produced by a PTC to that of a hand-crafted test collection by means of Kendall's tau again. However, the main focus will shift to optimizing retrieval algorithms on PTCs.

6

Pseudo Test Collections for Scientific Literature Search

In the previous chapter we treated the problem of profiling an expert in the setting of a university. In this chapter we look at another task in the domain of science: scientific literature retrieval. More importantly, perhaps, this chapter is the first of two chapters about using annotations to generate pseudo test collections (PTCs) automatically. As a first exploration of this idea, this chapter is relatively short. Recent years have seen increasing interest in generating pseudo test collections for training and evaluation purposes. This is primarily motivated by the costs associated with obtaining manual relevance assessments, and by the hunger of learning based ranking methods for large volumes of training material. Most approaches to generating ground truth leverage some kind of human behavior, such as annotation, hyperlinking, or simply using a search engine. Beitzel et al. [27] use the Open Directory Project, a large scale annotation effort targeting web pages in general. They assume relevance of documents to the title of the category they are listed under to generate relevance judgments. More recently, Asadi et al. [14] use anchor texts as queries and assume linked-to documents are potentially relevant documents. Web search is characterized by heterogeneous and high volume content and usage data. We investigate the generation of pseudo test collections in the less studied and more specialized domain of digital libraries.

Digital libraries are increasingly publishing their content online allowing people to access, browse, and search the archives. This type of content is typically semi-structured and manually annotated using rich descriptors. These characteristics differentiate it from web documents, and many retrieval methods have been developed to exploit them, improving retrieval effectiveness [70]. Modern Information Retrieval (IR) algorithms—especially in the form of learning to rank (LTR) methods—are able to learn to combine relatively uncertain evidence from individual features and typically improve retrieval effectiveness when large amounts of training data are available [138].

In this chapter, we focus on generating pseudo test collections that can be used to optimize retrieval algorithms for ad-hoc search on domain-specific, semi-structured documents. The most commonly used method for generating pseudo test collections is to sample and group documents in a collection by a certain criterion, and generate queries for these groups [14, 27]. In the domain of digital libraries, rich annotations are often available in the form of thesaurus terms, classification codes, or other descriptors that can

Algorithm 6.1: Algorithm for creating pseudo test collections for semi-structured domain-specific collections

1: Select one or more annotation types.
2: Sample one or more annotations from each type.
3: Sample documents associated with these annotations, populating R_q.
4: Generate a query q from the annotations and R_q.

be used as grouping criteria. Our leading intuition is that people provide this metadata in order to make documents better findable with regard to certain information needs. In this chapter, we use such annotations to group documents in *topics*, and generate simulated queries (*pseudo queries*) for and from these topics. The set of documents assigned to a topic is considered to be the relevant set of documents for the topic, where relevance is taken to be binary.

In the pseudo test collection generation process there are three key challenges that shape our research questions and contributions: (1) how to use annotations for grouping documents, (2) which documents to allow in the groups, and (3) how to simulate queries. The common ingredient among these challenges is the sampling of annotations. Not all annotations are equally specific (compare, e.g., "United States of America" and "workaholism"). Developing methods for sampling descriptors from different metadata fields can help to manipulate the generality and specificity of the resulting groups and therefore the resulting performance of LTR. In this chapter we tackle each of these challenges, using the domain-specific characteristics of ad-hoc search in scientific articles.

The research question that we address in this chapter is:

RQ4 Collections of scientific literature are sometimes indexed with different kinds of annotations. Can the result of these annotation efforts be used to generate a pseudo test collection (PTC)? When retrieval algorithms are evaluated using a PTC, are they ranked as they would be by a hand-crafted test collection? And when we test a L2R algorithm on a hand-crafted test collection, what is the best way to train it: on a different hand-crafted test collection, or on a PTC?

We detail our problem statement in Section 6.1. We present our methods in Section 6.2, conduct experiments in Section 6.3, report on our results in Section 6.4, discuss our findings in Section 6.5, and conclude in Section 6.6.

6.1 Problem statement

We first define the problem of generating pseudo test collections for semi-structured documents, and then describe our approach to this problem. A pseudo test collection is defined here as consisting of a set of generated queries Q and, for each query $q \in Q$, a set of documents assumed to be relevant, R_q (all documents within R_q are assumed to be equally good results). Given this definition, there are two main steps involved: (1) simulating the query and (2) simulating the relevant documents.

Our idea is to use document annotations for this. Let a document be annotated using several annotation types, each corresponding to a separate descriptor field, such as a field containing keywords from a thesaurus of research areas, or a field containing specific research methods. We can select one or more of these types of annotation as a starting point for creating pseudo-topics. Each document has a set of annotations of each type. We can select one or more of these annotations, e.g., the annotation "information retrieval" from a research area descriptor field. We can estimate a relevant set of documents R_q from the set of annotations we selected. Once we have sampled the set of annotations and their associated documents R_q, we can simulate a query. This way of thinking about the problem breaks it down to the subproblems listed in Algorithm 6.1.

In the following section, we will detail and explain the choices that we make for each of the steps in Algorithm 6.1. At each turn, it is useful to keep in mind that our main goal is to develop sampling methods that optimize the effectiveness of a learning to rank system in the setting of domain-specific retrieval. In contrast to other pseudo test collection research, we are not primarily interested in developing methods that produce pseudo test collections similar to manually crafted test collections. We evaluate our methods on the end-to-end performance of an LTR system, i.e., train on pseudo test collections generated by our methods, and test on manually crafted collections. For a test collection to be suitable as training material, it is important that the queries should be diverse, covering a wide range of topics. The scientific article collection we use has been annotated with a wide range of descriptors, which form a good source for obtaining a rich and diverse topic set. For a training query, it is desirable that there should be a reasonable amount of relevant documents to act as positive training examples. In other words, a training query should not be overly specific. On the other hand, training queries should not be overly general, either. In the following section, we discuss how we take these desiderata into account.

6.2 Sampling methods

Below we discuss instantiations for all the steps in Algorithm 6.1; we begin by sampling an annotation type (Step 1), then move on to sampling annotations (Step 2). All our PTC generation methods opt for a very straightforward option in sampling documents for R_q (Step 3): we require each document to have *all* the annotations we sampled. This means we view sampling more annotations as narrowing down the scope of the pseudo topic. Relaxing this requirement is an interesting possibility for future work. Finally, we discuss simulating queries (Step 4).

Step 1: Sampling annotation type We start with Step 1 in Algorithm 6.1. In our scientific literature collection, there are three main annotation types:

- `METHOD` can be any of 40 research methods, e.g., "descriptive study,"
- `CLASSIFICATION` is a classification code, e.g., "Labor Market Policy," and
- `CONTROLLED` is a thesaurus term, e.g., "social partnership."

The first two types (METHOD, CLASSIFICATION) cover broad topics, while annotations from CONTROLLED range from very broad to very narrow topics.

We sample annotations in two ways: from each type individually, and from all types simultaneously. In the first case, we generate pseudo test collections using only annotations from one type, CONTROLLED, because it offers a range of more general and more specific coverage, just like we would expect in queries. In the second case, we take the cross product $A_{METHOD} \times A_{CLASSIFICATION} \times A_{CONTROLLED}$ and the relevant sets of documents consist of documents that are annotated with the triple of annotations over the three types.

Step 2: Sampling annotations For Step 2 of Algorithm 6.1 we use two techniques for sampling annotations from annotation type CONTROLLED: randomly sampling single annotations, and randomly sampling pairs of annotations (sampling from $A_{CONTROLLED} \times A_{CONTROLLED}$), where the relevant sets of documents have both annotations. In the first case we observed that annotations ranged from broad to specific. Very specific annotations were associated with a very small number of documents, while some others were found very broad and were associated with a large fraction of the documents in the collection. Our second sampling method using pairs of annotations aims at accounting for this phenomenon: documents that have both annotations are intuitively more on topic than documents that have only one of the two.[1] Our third sampling strategy samples annotations from $A_{METHOD} \times A_{CLASSIFICATION} \times A_{CONTROLLED}$, as already noted above. In all three cases, to ensure that our sampled annotations are neither too broad or too specific, we select single annotations or pairs of annotations that are associated with between 100 and 1000 documents. The lower bound warrants enough positive training examples for a learning to rank algorithm. Another consideration for the lower bound is that we expect the log-likelihood ratio-based query generation technique described in the next section to work better if it can select terms using estimates based on a reasonable number of relevant documents. The upper bound on the number of documents associated with annotations discards very broad annotations. The scientific article collections we will search over have sum total of about 170K documents, and we do not want pseudo topics with a number of relevant documents that consists a too large fraction of the document collection.

Step 4: Generating queries For simulating the queries in Step 3 of Algorithm 6.1, we use two approaches: (1) use the annotations as queries, and (2) extract query terms from the simulated relevant set of documents (R_q).

Our first query simulation method is straightforward. All terms from the sampled annotations are used as query terms; we refer to this method as "Keywords". This method is also somewhat naive. Intuitively, the set of relevant documents for our keyword queries would be easy to locate for any retrieval algorithm. Our second query generation method samples terms only from the title and abstract fields of the scientific articles in the set of relevant documents. This second method is inspired by [15], who sample discriminative terms from a given document to generate a known-item query for that document. Be-

[1] We also experimented with sampling requiring a larger number of annotations to be present. The number of documents associated with them was found small, therefore of little use for training LTR systems.

cause, in our setting, we want to sample terms from the *set* of documents R_q associated with our sampled annotations, we cannot directly apply their methods. A technique that has been used in the literature to find discriminative terms in one corpus with regards to another corpus is based on a log-likelihood ratio, cf. [73, 182]; we use this query generation technique also in the next chapter. We briefly explain how this log-likelihood ratio (LLR) technique works. The idea behind the technique is to perform a statistical test for every term in the vocabulary, which uses term frequency information. We only consider terms that occur in at least a minimal amount of documents, set to 10. This is done to avoid selecting extremely rare terms, and to ensure the statistical test has some information to work with. The null hypothesis in the LLR test states that the observed frequency for a term in a set of corpora is what we would expect it to be given the size of the corpora. More formally, let O_i be the observed frequency for a term in corpus i, and let N_i denote the number of tokens in corpus i. The *expected frequency* E_i of the term in corpus i is then computed as follows:

$$E_i = O_i \frac{\sum_i O_i}{\sum_i N_i}.$$

In words: the expected frequency of the term is the number of tokens in corpus i multiplied by the probability of encountering the term in the concatenation of all corpora. According to the null hypothesis, that single probability explains the observed frequency in each corpus. Formally, the null hypothesis states that

$$H_0 : O_i = E_i, \text{ for all corpora i.}$$

The *relative term frequency* in a corpus is given by $\frac{O_i}{N_i}$. If the null hypothesis holds, then the relative term frequency of w is the same for all corpora. In our setting, we have only two corpora of interest: the document set associated with our sampled annotations (R_q), and the rest of the scientific article collection, let us denote this by C. If the null hypothesis does not hold, either a term is *over-represented* in R_q, or it is under-represented; we are interested in over-represented terms. To test the null hypothesis in this experimental design, the chi-square statistic could be used, but Dunning [73] propose the log-likelihood ratio as an alternative, because the chi-square test becomes unreliable when E_i is smaller than five, and it possibly overestimates significance for terms with high frequency and when one of the corpora is much smaller than the other [182]. Note that in our setting indeed $|R_q| \ll |C|$, and in the small corpus R_q, expected term frequencies will often be smaller than five. The *log-likehilood ratio* (LLR) is computed as

$$\text{LLR} = \sum_i O_i \ln \frac{O_i}{E_i},$$

and the likelihood ratio statistic G as $G = 2 \times \text{LLR}$. At this point, we could lookup a p-value for G corresponding to the probability of our observation under the null hypothesis. Since we are interested in the most over-represented terms, we instead simply rank terms by G, decreasing (as in e.g., [182]). Then, we select the *top ten* terms from this ranking as query terms.[2] Setting the length of the query to a fixed length of ten terms is a somewhat

[2]Incidentally, in our experiments, we did not discard under-represented terms, but, since $|R_q| \ll |T|$, and we only consider terms that occur at least once in R_q, under-represented terms are rare.

arbitrary choice. In the next chapter, in the context of microblog search, we report on experiments with generated queries that have fewer terms. Finally, note the similarity of query generation to query expansion in pseudo relevance feedback. Consequently, there are many alternatives to LLR to explore, e.g., applying methods from [15] by concatenating the documents in R_q into a single document, applying relevance modeling [130], and so on.

6.3 Experimental setup

To answer our research question RQ4, we evaluate our methods for constructing pseudo test collections with regard to their effectiveness for training an LTR system. We do this by testing PTC-trained L2R models on the test collections used in the CLEF Domain Specific track [173]. We compare their test performance to L2R models that have been trained on hand-crafted test collections. In addition, we benchmark widely used retrieval algorithms on each PTC and on each hand-crafted test collection, and we compare how systems are ranked. Below, we describe in more detail the datasets that we use, the PTCs we generate, the experiments we perform, the learning to rank (L2R) algorithm we employ, and the evaluation metric and significance tests we use.

Dataset We use the test collections used in the CLEF Domain Specific track in 2007 (25 topics) and 2008 (25 topics) in our experiments. Of the topics, we only used the title fields. The topics were created in German, and translated into English and Russian; we used the English topics. Documents for the Domain Specific track are scientific articles, for which titles, author names, annotations, and, often, abstracts are available. There were German, English, and Russian corpora. We used the two English corpora: (1) the English GIRT corpus, which is a translation of a German GIRT corpus, and (2) the CSA SA corpus. The English GIRT corpus contains just over 150,000 documents, roughly 17% of which contain an abstract. The smaller CSA SA corpus contains 20,000 documents, 94% of which contain an abstract [174]. The English GIRT and the CSA SA collections were available both in 2007 and in 2008.

Generated PTCs For generating PTCs, we use only the GIRT corpus. We generate pseudo test collections that use both single annotations and pairs of annotations from the `CONTROLLED` type, and triples of annotations over all three types (i.e., `METHOD`, `CLASSIFICATION`, `CONTROLLED`). In each case, we keep only topics with between a hundred and a thousand documents, resulting in the following numbers of pseudo topics: 2,073, 7,039 and 4,161, respectively. Each of these three sampling methods is coupled with two query simulation methods: using keywords and using LLR. This yields six PTC generation recipes in total. For generating queries using our LLR technique, we indexed the collection with Lucene,[3] a comprehensive search engine library. We did not perform stemming. This yields queries that, like realistic queries, consist of unstemmed words. Since our main aim is not to generate realistic queries, however, there is no reason not to experiment with stemming for query generation, if this improves the usefulness of the resulting PTC for training purposes. We did remove a standard set of stopwords.

[3] `https://lucene.apache.org/core/`

Experiments We are interested in how training on our pseudo test collections compares to training on hand-crafted test collections. To put this more concretely: How does training on the 2007 topics compare to training on the PTCs when the learned models are tested on the 2008 topics?; and vice versa for the 2008 topics. In addition, we report the performance obtained on the training data of the models trained on hand-crafted test collections. We also compare evaluation outcomes of using our PTCs and hand-crafted test collections for ranking eleven widely used retrieval algorithms. Between each pair of (pseudo) test collections, we compute the correlation of the two system rankings using Kendall's τ, following [224]. We report significant correlations (two-tailed) at the $\alpha = 0.01$ level ($\tau \geq 0.600$) in boldface [204]. The relatively conservative alpha level is used to reduce the chance of seeing a Type I error (rejecting the null hypothesis while it actually holds), as in Chapter 5.

Learning to rank For retrieval we use a learning to rank approach. We use a pairwise learning to rank approach based on a support vector machine [198, 202],[4] with default values for its hyperparameters, except for the way it samples training examples during training, which was set to optimize performance for the area under the ROC curve, which was found to work well in preliminary experiments. Using a single L2R algorithm to estimate the usefulness of our PTCs as training material raises the question to what extent our findings will translate to different retrieval algorithms. In the next chapter, where we pursue a similar PTC generation idea, we will tune parameters of a variety of retrieval models (e.g., language modeling, BM25) on our PTCS. Moreover, we will experiment with three L2R algorithms. For L2R, we use two feature sets: (1) query-independent, and (2) query dependent. Table 6.1 lists eleven query-dependent features (top-half), which are the outputs of off-the-shelf and widely used retrieval algorithms, and nine query-independent features.

For the query-dependent features, which we discussed in Chapter 2, we use the Indri and Terrier retrieval software packages. We do not tune the parameters of the retrieval models, e.g., μ for language modeling with Dirichlet smoothing, but use default values for every model. It is true that by tuning such parameters we may expect to increase our performance, but in this chapter our main focus is on comparing the experimental condition of training on a PTC with training on a hand-crafted test collection. For Indri indexing, we use a Porter stemmer, but no stopword removal. For Terrier indexing, we do remove stopwords, and use Porter stemming. Preprocessing choices like removing stopwords or leaving them in the index can affect retrieval performance. Rather than experimenting what works best and opting for exactly the same setup in Indri and Terrier, we choose different preprocessing options for Indri and Terrier, and trust our L2R algorithm to favor the features that work best. Diversity in features, intuitively, can benefit L2R performance, even if some individual features are weak. Both with Indri and Terrier we index all fields, also the fields containing the annotations. There are no fielded retrieval models among our query dependent features, e.g., BM25F, so we build a single index that contains text from all fields. In future work, it would be interesting to examine the tuning and training of fielded models on our PTCs. Intuitively, while training a L2R model on a PTC, it would be complicated to estimate feature importance for features that

[4] `http://code.google.com/p/sofia-ml/`

Table 6.1: Query-dependent and query-independent features for learning to rank. For the features that use properties of authors, we calculate four different values, one based on the first author, and three calculated based on all authors: the maximal, minimal and average value.

	Abbr	Description and parameters
Query-dependent features		
Indri 5.1	Tf-idf	Tf-idf run, with $k_1 = 1.2$ and $b = 0.75$.
	BM25	BM25 run, with $k_1 = 1.2$, $b = 0.75$, and $k_3 = 7$.
	LM	Language modeling, with Dirichlet smoothing, $\mu = 2500$.
	BOW	Boolean ordered window, with unlimited window size.
	BUW	Boolean unordered window, with unlimited window size.
	PRF	Pseudo-relevance feedback [130], we use the 10 top pseudo-relevant documents, we extract 10 terms, we give the original query 0.5 weight and use $\mu = 0$.
Terrier 3.5	Tf-idf	Tf-idf run, with $k_1 = 1.2$, $b = 0.75$.
	DFRee	A parameter free DFR (Divergence from Randomness) model.
	PL2	Another DFR run, with $c = 1.0$.
	PRF	A query expansion run, with DFR model Bose-Einstein 1. Query is expanded with the top 10 terms, obtained from the top 3 documents.
	DFR-FD	A full dependence DFR proximity dependence model [171], with proximity ngram length of 2.
Query-independent features		
	DocLength	Number of terms in title and abstract.
	Authors	Number of authors of article.
	Age	Age of publication (2008 - publication year).
	Publications	Number of publications by authors {max, first, avg, min}.
	Co-authors	Number of co-authors of authors {max, first, avg, min}.
	Degree	Degree-centrality of authors {max, first, avg, min}.
	Closeness	Closeness-centrality of authors {max, first, avg, min}.
	Pagerank	Pagerank of authors {max, first, avg, min}.

capitalize strongly on the same annotation fields used for generating a PTC. If this proves true in practice, one can experiment with training on a variety of PTCs and, if available, handcrafted ground truth. We normalize features as follows. For the Indri language modeling runs (Indri-LM, Indri-BOW, Indri-BUW, Indri-PRF) we take the exponential of the scores. Then, for each feature, we normalize by dividing by the maximal value for that feature over all documents. In addition to the query dependent features listed in Table 6.1, we use the query clarity feature by [62].

Our query-independent features include degree-centrality and closeness-centrality. These are properties of nodes in an undirected graph that can be used as measures of influence or centrality in a collaboration network [74]. We calculated them on the co-author graph where nodes are authors and edges exist between authors who co-authored at least one paper, using NetworkX.[5] We assumed that two author fields refer to the same author if, and only if, the strings match exactly. On the one hand, this approach can spread evidence for an author over different surface forms of his name in publications. On the other hand, it can conflate evidence for different authors who use the same surface form of their name in publications. A research area concerned with this problem is *author name disambiguation*. For the purposes of our experiments, a full-blown author name disambiguation approach is out of scope. Query-independent features have values equal to or greater than zero. We normalize each feature by dividing it through its maximal value over all documents.

To label the training set of pseudo topics for a L2R algorithm, we need positive and negative training examples. As positive examples, we use documents in R_q, the set of documents associated with the annotations of the pseudo-topic. As negative training examples, following [14], we sample documents from the bottom (starting from rank 1000, going up) of the ranking of a retrieval algorithm, in particular, the Indri language modeling (LM) algorithm, sampling twice as many negative training examples as there are positive examples in R_q.

Evaluation For our retrieval experiments, we report on mean average precision (MAP). Statistical significance testing is done using Fisher's pairwise randomization test [209], with $\alpha = 0.001$. We use a conservative α level to keep Type I errors under control, as we are making many pairwise comparisons. Significant differences are marked with $^{\blacktriangle}$(better) or $^{\blacktriangledown}$(worse).

6.4 Results

Our first experiment is aimed at answering the question whether training on pseudo test collections leads to different performance from training on hand-crafted test collections. In the rightmost two columns of Table 6.2, we list the MAP performance scores of our learning to rank algorithm on the 2007 and 2008 topic sets from the CLEF Domain-Specific track, respectively. For scores on the 2007 topics, we list in boldface the runs that are significantly different from the run that was trained on the 2008 queries. For the 2008 topics, we list in boldface the runs that differ significantly from the run trained on

[5] `http://networkx.lanl.gov`

Table 6.2: MAP performance scores of our learning to rank approach on the CLEF Domain-Specific 2007 and 2008 topics (title only). The annotation types A_M, A_{CL} and A_{CO} are short for A_{METHOD}, $A_{CLASSIFICATION}$ and $A_{CONTROLLED}$, respectively. The query generation method is given in brackets. For performance scores on the 2007 topics, it is indicated if scores differ significantly ($\alpha = 0.001$) from scores obtained by training on the 2008 topics; and vice versa for scores on the 2008 topics. The training performance for each of the 2007 and 2008 topic sets is given in italics. The best performance (excluding training performances) are given in boldface.

	Test on CLEF DS	
Train on	2007	2008
CLEF DS 2008	**0.2347**	*0.3158*
CLEF DS 2007	*0.2226*	**0.2970**
A_{CO} (Keywords)	0.1985$^{\blacktriangledown}$	0.2734
A_{CO} (LLR)	0.1155$^{\blacktriangledown}$	0.1869$^{\blacktriangledown}$
$A_{CO} \times A_{CO}$ (Keywords)	0.2091$^{\blacktriangledown}$	0.2866
$A_{CO} \times A_{CO}$ (LLR)	0.1240$^{\blacktriangledown}$	0.1959$^{\blacktriangledown}$
$A_M \times A_{CL} \times A_{CO}$ (Keywords)	0.1329$^{\blacktriangledown}$	0.1609$^{\blacktriangledown}$
$A_M \times A_{CL} \times A_{CL}$ (LLR)	0.1979$^{\blacktriangledown}$	0.2602

the 2007 topics. Training performance scores—where training and testing is done on the same set of queries—are given in italics. Each row corresponds to one test collection. The top two rows contain scores for training on the two hand-crafted test collections (the 2007 and 2008 topic sets). The bottom six rows contain scores for our six PTC generation recipes.

When we evaluate on the 2008 test topics, we see that three of our six methods of generating a pseudo test collection yield performance that is similar to training on the 2007 test topics: the differences are not statistically significant. This result provides first evidence for the utility of our pseudo test collection generation methods.

Looking at which methods perform well, we see that for $A_{CONTROLLED}$, it is best to use terms occurring in the annotation as query terms, rather than generating a query with LLR, which is worse on both the 2007 and 2008 topics, even though the difference is only significant on the 2007 topics. We observe a similar result for $A_{CONTROLLED} \times A_{CONTROLLED}$; in this case using LLR is significantly worse for both 2007 and 2008. However, for $A_{METHOD} \times A_{CLASSIFICATION} \times A_{CONTROLLED}$, generating the query with LLR is more successful, significantly so for the 2008 topics.

Evaluating on the 2007 test topics yields a different picture. In this case all our methods are significantly outperformed by a learning to rank system trained on the 2008 topics.

Comparing scores between training performances (listed in italics) and test performances is useful mostly to check that a model does not overfit. Complex learning algorithms may obtain a perfect score on any training set, but generalize very poorly to a test set. In these results, we see that our learning to rank algorithm does not seem to overfit.

Table 6.3: MAP performance of our individual query-dependent features. Best scores for both CLEF DS topic sets are given in bold.

		CLEF DS	
	Algorithm	2007	2008
Indri 5.1	Tf-idf	0.2028	0.2723
	BM25	0.1821	0.2707
	LM	0.1835	0.2051
	PRF	0.1854	0.1984
	BUW	0.0733	0.1678
	BOW	0.0531	0.1344
Terrier 3.5	PRF	**0.2599**	**0.3360**
	DFRee	0.2183	0.3107
	DFR-FD	0.2355	0.3085
	Tf-idf	0.2381	0.2941
	PL2	0.2277	0.2794

On the 2007 topics, the training performance is actually lower than the score obtained by training on the 2008 topics. On the 2008 topics, the training performance is highest, but it is not significantly higher than the score obtained by training on the 2007 topics.

6.4.1 Performance of individual features

For completeness, we list scores of the individual retrieval algorithms, which were used as query-dependent features in our L2R setup, in Table 6.3. The Indri and Terrier runs are ordered decreasingly by MAP on 2008 topics. The best query-dependent feature is Terrier PRF. However, for 2007, it does not improve significantly over the other Terrier features. Also, with regard to the learning to rank runs: for 2007, it does not significantly outperform the runs that trained on 2008 topics, the 2007 topics, or $A_{CONTROLLED} \times A_{CONTROLLED}$ (Keywords). It is significantly better than all other runs for 2007. For 2008, Terrier PRF does not significantly outperform Indri-tf-idf, nor the other Terrier features. With regard to the learning to rank runs, it does not significantly outperform the runs that train on the 2007 topics, the 2008 topics, or $A_{CONTROLLED}$ (Keywords). All other runs are significantly outperformed. This is strong evidence that PRF can be very succesful in this domain-specific setting, as was found also in, e.g., [83, 110]. Interestingly, the Indri PRF implementation has lower performance scores than the Indri Tf-idf and BM25 runs. Possibly, this is a side-effect of our choice not to remove stopwords for the Indri runs.

Some query-dependent feature scores are very high, and even outperform our learning to rank approach in some cases. This is a sign that there is room for improvement in our L2R features and, possibly, in the choice of L2R algorithm. Ideally, L2R algorithms should outperform all individual features. Our main focus, however, is not on showing that we can outperform the best query-dependent feature. Rather, it is to show that we can use pseudo test collections for training retrieval algorithms, with the same effectiveness

Table 6.4: Kendall's tau values between system rankings produced by hand-crafted and pseudo test collections. The annotation types A_M, A_{CL} and A_{CO} are short for A_{METHOD}, $A_{CLASSIFICATION}$ and $A_{CONTROLLED}$, respectively. The query generation method is given in brackets. Significant correlations ($\alpha = 0.01$, $\tau \geq 0.600$) are given in boldface.

	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) 2007	**0.745**	0.309	0.294	0.382	0.294	**0.636**	0.404
(2) 2008		0.418	0.110	0.564	0.110	**0.891**	0.220
(3) A_{CO} (Keywords)			-0.147	0.564	-0.147	0.382	-0.037
(4) A_{CO} (LLR)				-0.110	**0.982**	0.000	**0.800**
(5) $A_{CO} \times A_{CO}$ (Keywords)					-0.110	**0.600**	0.000
(6) $A_{CO} \times A_{CO}$ (LLR)						0.000	**0.800**
(7) $A_M \times A_{CL} \times A_{CO}$ (Keywords)							0.110
(8) $A_M \times A_{CL} \times A_{CO}$ (LLR)							

as using hand-crafted test collections for training.

6.4.2 Using pseudo test collections for evaluation

In principle, pseudo test collections can be used for *evaluation* purposes. In Table 6.4 we list Kendall's tau values between system rankings produced by the hand-crafted and pseudo test collections. The systems ranked here are the same eleven retrieval algorithms we used for our query dependent features (see Table 6.1).

The first two rows contain system ranking correlations between the two hand-crafted test collections and all other test collections. There is a significant correlation between how the 2007 and 2008 topics rank the eleven retrieval algorithms. There are no negative correlations between the system rankings produced by the hand-crafted test collections and any other collection. It is interesting to note that the pseudo test collection with the strongest correlation with a hand-crafted test collection is $A_M \times A_{CL} \times A_{CT}$ (Keywords); the method that uses documents associated with an annotation triple (`METHOD,CLASSIFICATION,CONTROLLED`). This is in stark contrast with our previous observation that this pseudo test collection should not be used to train a learning to rank system on.

6.5 Discussion

We have shown that it is possible to use the rich annotations available in digital libraries collections for training a learning to rank system. We assumed that people annotate documents to make them better findable for certain information needs. We identified four main steps, addressing what kind of annotations to use, how to sample annotations, how to sample relevant documents, and how to generate queries. We tackled all four steps and showed that it is possible to generate pseudo test collections in the digital library domain

on which a learning to rank system can be trained, such that in some cases performance is indistinguishable from training on editorial topics and judgments. In particular, when testing on the 2008 topics, for three pseudo test collections it holds that training on them yields performance on par with training on 2007 editorial judgments. There is room for improvement with regard to training on the 2008 topics: this strategy outperforms our methods when tested on the 2007 topics.

There are some limitations in our work, which we aim to address in future work. One of them is that our learning to rank algorithm is unable to outperform our best query-dependent feature. We plan to experiment with other learning to rank algorithms and to go beyond using such an algorithm as a black-box. In Chapter 7, where we generate pseudo test collections for microblog search, we experiment with several L2R algorithms, which outperform individual retrieval algorithms.

Another limitation is that we used off-the-shelf retrieval algorithms, and did not tune their parameters. This may limit the quality of our features. In Chapter 7, we do tune parameters for abovementioned retrieval models. Moreover, we do so on pseudo test collections for microblog search, showing another way to put pseudo test collections to good use.

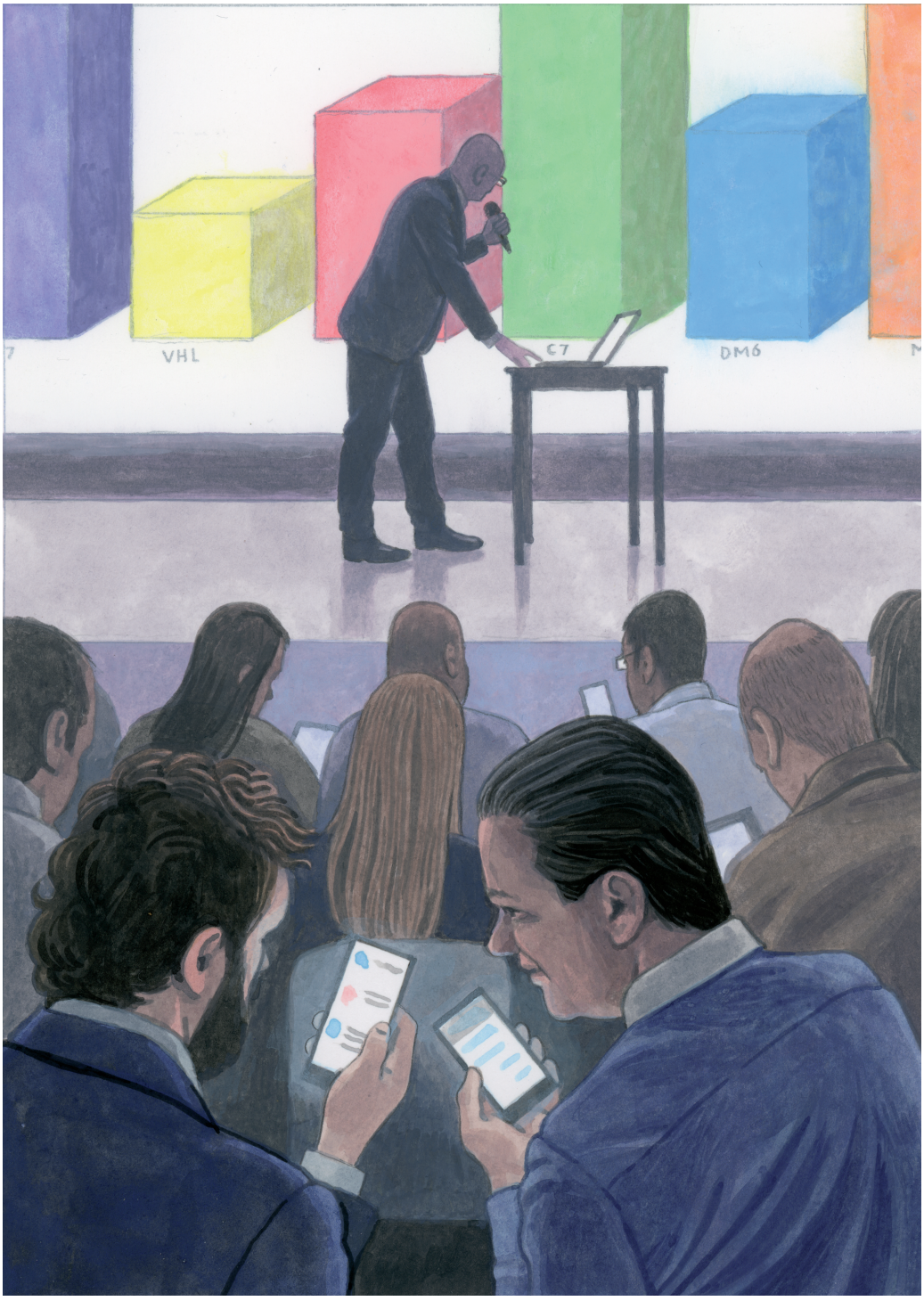
There are some interactions that we do not yet fully understand. One of them is the following. Recall that Asadi et al. [14] sample non-relevant documents from the bottom of a retrieval algorithm ranked list, and we followed this procedure. We chose Indri LM, but noticed that the choice of algorithm to use has a considerable impact on performance. For example, selecting the Indri tf-idf algorithm instead of Indri-LM made oracle run performance drop from about MAP 0.30 to MAP 0.25 for 2008 topics. Our choice of the Indri LM retrieval function was arbitrary, as of yet we have a limited understanding of the properties such a retrieval function should have. In Chapter 7, we sample negative training examples randomly, which turns out to work well, and introduces surprisingly little variance.

The performance of our query-independent features was disappointing. The Pegasos [202] algorithm we used for learning to rank learns a linear model, and the weights for all our query-independent features were close to zero. We used 24 query-independent features in this chapter, but none of them seemed promising enough in a learning to rank setting in order to use them in the query generation process. In future work, we plan to use richer collections which give us the opportunity to test stronger query-independent features based, e.g., on the citation graph.

Our results raise interesting questions that may be addressed in future work. For example, why does the success of the two query term sampling strategies depend so strongly on the ground truth sampling strategy that is being used and vice versa? And, consider the correlations between system rankings produced by our PTCs and system rankings produced by hand-crafted test collections. These correlations were positive, but mostly insignificant. The only significant correlation observed was for a PTC that performed poorly when used to train a L2R algorithm on. Apparently using a PTC for benchmarking is different from using it as training material. What are important characteristics that a PTC should have for either of these use cases? Another interesting direction for future work is training on a variety of PTCs, to increase diversity in training material.

6.6 Conclusion

We have shown that it is feasible to generate pseudo test collections for training a learning to rank system on scientific document collections. We proposed three pseudo test collection generation methods for which we could show that for one of our test sets, training on these collections is just as effective as training on editorial topics and judgments. We pointed to interesting directions for future work and areas where we need to deepen our understanding. In Chapter 7, we provide another study on the idea of generating pseudo test collections for training L2R algorithms, addressing several of the limitations raised in the previous section.


VHL
C7
DM6

7

Pseudo Test Collections for Microblog Search

In the previous chapter we discussed generating pseudo test collections (PTCs) for scientific article retrieval, for both optimization and evaluation purposes. In this chapter we develop a similar approach to generate PTCs for the much more noisy setting of microblog search. We focus exclusively on the optimization of retrieval algorithms. Modern information retrieval (IR) systems have evolved from single model based systems to intelligent systems that learn to combine uncertain evidence from multiple individual models [61, 138]. The effectiveness and flexibility of such systems has led to wide adoptation in IR research. A key contributor to the success of such systems is the learning phase, i.e., the training set they are given for learning. Training sets have to be tailored to the task at hand and, in contrast to the systems themselves, do not generalize to other tasks. This characteristic requires compiling task-specific training sets, which is a time consuming and resource intensive process, as it usually involves human labor. Automating the process of compiling training sets has obvious advantages in reducing costs, while it simultaneously increases the size of the training set. This observation has led to a persistent interest in finding ways for generating so-called *pseudo test collections*, which consist of a set of queries, and for each query a set of relevant documents (given some document set).

Microblog search is the task of finding information in microblogs, such as Facebook status updates, Twitter posts, etc. The task became popular with the advent of social media and is distinct from web search and from blog search due mainly to its real-time nature, the very limited length of microblog posts and the use of "microblog language," e.g., hashtags, mentions, which can provide useful information for retrieval purposes. In 2011 the Text REtrieval Conference (TREC) launched the Microblog track aimed at developing a test collection from Twitter data and evaluating systems' performance on retrieving—given a query and time-stamp—relevant and interesting tweets in a simulated real-time scenario. Several participants approach the task using learning to rank methods for combining evidence from multiple rankers [166]. This approach to microblog search comes natural because of the many dimensions available for ranking microblog posts, e.g., recency, user authority, content, existence of hyperlinks, hashtags, retweets. For training a learning to rank (L2R) based system at the TREC 2011 Microblog track, participants use a traditional supervised method: many manually labeled data for com-

piling a training set. What if we could generate the required training sets automatically? In 2012 a second edition of the Microblog track was organized. This gives us the opportunity to compare what yields better learning to rank performance: training on the 2011 relevance assessments, or training on automatically generated ground truth?

Our starting point is the following intuition, based upon the observation that hashtags tend to represent a topic in the Twitter domain: *From tweets T_h associated with a hashtag h, select a subset of tweets $R_h \subseteq T_h$ that are relevant to an unknown query q_h related to h.* We build on this intuition for creating a training set for microblog rankers. To this end, we take several steps, each giving rise to multiple options. First, we select hashtags h and associated relevant tweets R_h. Can we just select all hashtags and use all their associated tweets? In microblog search, time is important: what is considered relevant to a query may change rapidly over time. A microblog query, then, has a timestamp, and relevant tweets must occur prior to this timestamp. As for a query, the topic a *hashtag* is associated with may change over time. Can we exploit this analogy, and label hashtags with a timestamp, regarding tweets prior to this timestamp as relevant? Another well-known aspect of microblog posts is that they often contain casual conversation that is unlikely to be relevant to a query. Can we improve generated training sets by selecting interesting tweets and hashtags associated with such tweets? Once we have selected a hashtag h and a set of tweets R_h, how do we generate a query q_h related to h? The research question that guides our work in this chapter is:

RQ5 Hashtags are used in microblog search to indicate that a tweet is part of a larger discussion. We assume that tweets sharing the same hashtag share the same topic, by and large. Can we build on this assumption to generate a PTC? And when we test an L2R algorithm on a hand-crafted microblog search collection, what yields better performance: training on a different hand-crafted collection, or training on a PTC? We consider three recipes for generating a PTC. What is their relative merit as training material? We consider three strong L2R algorithms. How will our findings vary with a different choice of L2R algorithms? And, how succesful can free parameters of individual retrieval algorithms such as language modeling be tuned on our PTCs?

The main contribution in this chapter is a set of methods for creating pseudo test collections for microblog search. These collections are shown to be useful as training material for tuning well-known retrieval methods from the literature, and for optimizing a learning to rank method. In particular, we contribute: (1) unsupervised pseudo test collection generation methods; (2) a supervised pseudo test collection generation method, where we learn what are interesting tweets from TREC Microblog track assessments; (3) insights into the sensitivity of our methods to parameter settings.

7.1 Problem definition

Below, we consider a number of instantiations of our pseudo test collection generator. For the purposes of the TREC Microblog track, a *test collection for microblog search* consists of queries with timestamps and a set of relevant documents for these queries.

A *pseudo test collection for microblog search* consists of a set of queries $\mathcal{Q}$, in which each query $q \in \mathcal{Q}$ is associated with a timestamp q_t and a set of relevant documents R_q. Given this definition, there are three main steps for generating a pseudo test collection for microblog search: generating (1) the query; (2) the query timestamp; and (3) a set of relevant tweets for the query.

We start from the following intuition: *From the tweets T_h that contain a hashtag h, we can select tweets R_h that are relevant to an unknown query q_h related to h.* In the next section, we present three methods to generate a pseudo test collection. Each method selects hashtags and for every hashtag h it selects tweets R_h from T_h that will act as relevant tweets to a suitable query related to h. In Section 7.3, we present a technique for generating queries from R_h.

7.2 Selecting hashtags and tweets

We propose four solutions to selecting hashtags and tweets for inclusion in a pseudo test collection:

Random A sanity check baseline against our hypothesis that hashtags are good sources for generating pseudo test collections. Collections are created by randomly sampling a set of relevant tweets for each topic, without replacement. All these random collections are of a fixed size, equal to our largest hashtag-based pseudo test collection.

Hashtags A naive method that serves as baseline in our experiments and that considers all hashtags and tweets to be equally important (Section 7.2.1).

Hashtags-T A method that creates a test collection in the microblog retrieval sense, in which queries have timestamps (Section 7.2.2).

Hashtags-TI A method that aims at capturing interestingness in tweets. Interesting tweets should contain good candidate terms for a query. We present a method with which we can estimate from example queries and relevant tweets the probability of interestingness of a tweet (Section 7.2.3).

7.2.1 Hashtags: all hashtags, tweets are equal

We select all hashtags subject to a single requirement: that they are mentioned in a reasonable amount of tweets, m (Algorithm 7.1). There are three reasons for this lower bound: (1) it reflects a certain consensus about the meaning of a hashtag; (2) we generate our queries based on word distributions in these tweets: for this to work reliably, we need a reasonable amount of tweets, see Section 7.3; (3) we train a learning to rank retrieval algorithm on our pseudo test collection; we hypothesize that it would benefit from a relative large set of positive training examples, see Section 7.4. We normalize hashtags by lowercasing them and removing any non-alphanumeric characters. In all our experiments, we set $m = 50$. The pseudo test collection generated by Algorithm 7.1 is called *Hashtags*.

Algorithm 7.1: Generating collection *Hashtags*

```
H ⟵ {h : |T_h| >= m};
for h ∈ H do
    R_h ⟵ T_h;
    Generate query q_h from R_h;                    // See Section 7.3
end
```

7.2.2 Hashtags-T: generating timestamps

Microblog search is sensitive to the query issue time because of the real time nature of tweets. To generate a timestamp for a query related to a hashtag h, we make an analogy between search volume over time for a query and publishing volume over time for tweets that contain h. Our assumption is that users often issue queries for trending topics because they want to monitor developments, similar to certain types of blog search [153]. We generate a timestamp for hashtag h just after peaks in publishing volume. In this way, our generated queries will be about trending topics. In addition, we keep a large amount of tweets from T_h, while discarding a limited number, after h stops trending. In collections that span a considerable period of time, re-occurring topics, such as Christmas or Super Bowl, may quite likely be observed. In this case, one may want to assign multiple issue times for a query, depending on the number of observed peaks. Our corpus (see Section 7.4) covers a relatively short period, and we assign only one issue time to every query sampled.

In detail, our query issue time generation works as follows. First, we group the time span of the collection in 8-hours bins. Then, for each hashtag, we count how many relevant documents belong to each bin; this results in generating the hashtag's timeseries. In our setting, timeseries are short and sparse; our peak detection method aims at coping with this challenge. We find the bin with the most counts and resolve ties by taking the earliest date. This approach allows us to return a peak even for very sparse timeseries. We call the pseudo test collection generated by Algorithm 7.2: *Hashtags-T*.

Algorithm 7.2: Generating collection *Hashtags-T*

```
H' ⟵ {h : |T_h| ≥ m};
for h ∈ H' do
    Generate timestamp t(h);                        // See Section 7.2.2
    R_h ⟵ {τ : τ ∈ T_h and t(τ) ≤ t(h)};
end
H ⟵ {h : h ∈ H' and |R_h| ≥ m};
for h ∈ H do
    Generate query q_h from R_h;                    // See Section 7.3
end
```

7.2.3 Hashtags-TI: selecting interesting tweets

Consider the following tweet: "Hey follow me here #teamfollowback #justinbieber." We hypothesize that this tweet would not be useful for sampling terms for topics labeled #teamfollowback or #justinbieber or as a relevant document for these topics. To avoid selecting such tweets, we rank tweets by their probability of interestingness and keep the best X percent. We think of a tweet as *interesting* if it carries some information and could be relevant to a query. We use a set of criteria to capture interestingness and present a method to learn from example queries and relevant documents from an editorial collection.

Let $C_1, C_2, \ldots, C_n$ be random variables associated with the criteria and let $I = 1$ denote the event that a tweet is interesting. We estimate the probability that a tweet τ is interesting, given the observed values for the criteria: $P(I = 1|C_1 = c_1, \ldots, C_n = c_n)$, or, shorthand: $P(I|c_1, \ldots, c_n)$. Following Bayes' rule, we have

$$P(I|c_1, \ldots, c_n) = \frac{P(c_1, \ldots, c_n|I)P(I)}{P(c_1, \ldots, c_n)}, \tag{7.1}$$

where $P(I)$ is the a-priori probability that a tweet is interesting, $P(c_1, \ldots, c_n|I)$ is the likelihood of observing the evidence given that a tweet is interesting, and $P(c_1, \ldots, c_n)$ is the probability of observing the evidence. The crucial step is to estimate $P(c_1, \ldots, c_n|I)$. We hypothesize that tweets that are known to be relevant to a query are interesting and estimate $P(c_1, \ldots, c_n|I)$ with $P(c_1, \ldots, c_n|R)$, where R is the event that a tweet is relevant to a query in an editorial collection. We use the TREC Microblog 2011 qrels for this estimation. Since we do not have enough relevant tweets to estimate the full joint probability, we assume conditional independence of c_i given that a tweet is relevant:

$$P(I|c_1, \ldots, c_n) \approx \frac{\left(\prod_i P(c_i|R)\right) P(I)}{P(c_1, \ldots c_n)}. \tag{7.2}$$

Since we rank tweets by interestingness we do not have to estimate $P(I)$: it is the same for all tweets. On the other hand, we have to keep and estimate $P(c_1, \ldots, c_n)$, because it is different for different tweets. Therefore, we have:

$$\text{rank}_{\text{tweets}}\left(\frac{\left(\prod_i P(c_i|R)\right) P(I)}{P(c_1, \ldots c_n)}\right) = \text{rank}_{\text{tweets}}\left(\frac{\sum_i \log(P(c_i|R))}{P(c_1, \ldots, c_n)}\right). \tag{7.3}$$

Most of the criteria we use have discrete distributions. For those that do not, we bin their values in B bins; we set $B = 10$. To avoid rejecting a tweet on the basis of one measurement that did not occur in any of the relevant tweets, we add one observation to every bin of every $P(c_i|R)$ distribution. For estimating $P(c_1, \ldots, c_n)$ we use the empirical distribution of all tweets in the collection T. To do this, we bin feature values for the criteria $C_1, \ldots C_n$. Again, most of the criteria have discrete distributions, and for those that do not, we create $B = 10$ bins. After binning, $C_1, \ldots, C_n$ all assume discrete values $c_1, \ldots, c_n$. Then,

$$P(c_1, \ldots, c_n) = \frac{|\{\tau \mid C_i = c_i \text{ for all } i\}|}{|T|}. \tag{7.4}$$

After selecting the best X percent of tweets, we again filter out hashtags that have less than 50 interesting tweets. We build this method on top of our pseudo test collection *Hashtags-T*, only ranking the tweets in this collection and keeping the best 50% of them. We call the pseudo test collection generated by Algorithm 7.3 *Hashtags-TI*.

Algorithm 7.3: Generating collection *Hashtags-TI*

```
H'' ⟵ {h : |T_h| ≥ m};
for h ∈ H'' do
    Generate timestamp t(h);                          // See Section 7.2.2
    T_{h,t} ⟵ {τ : τ ∈ T_h and t(τ) ≤ t(h)};
end
H' ⟵ {h : h ∈ H'' and |T_{h,t}| ≥ m};
T ⟵ ⋃_{h∈H'} T_{h,t};
// Rank tweets by probability of being interesting
Rank T by P(I|c_1, ..., c_n);                         // See eq. 7.2
Let T_I be the top X percent of this ranking;
for h ∈ H' do
    R_h ⟵ T_{h,t} ∩ T_I;
end
H ⟵ {h : h ∈ H' and |R_h| ≥ m};
for h ∈ H do
    Generate query q from R_h;                        // See Section 7.3
end
```

The criteria we use build on textual features (*density* and *capitalization*) and microblog features (*links*, *mentions*, *recency*). Each criterion is discussed below. The marginal distributions $P(c_i|R_\tau)$ of three criteria are shown in Figure 7.1 as white histograms. They overlap with black histograms of all tweets in our *Hashtags* pseudo test collection. These criteria have different distributions over relevant tweets and over tweets that have a hashtag, which motivates our idea to keep tweets with high probability of interestingness.

Links The existence of a hyperlink is a good indicator of the content value of a tweet. If the referenced web page adds useful information, it increases the potential relevance of a Tweet. TREC Microblog 2012 assesors followed outlinks to check this. Also, a large fraction of tweets are pointers to online news [129]. Tweets with links are likely to include terms that describe the linked web page, rendering them good surrogates for query terms [39].

Mentions Tweets with mentions (@username) signify discussions about the hashtag's topic. This type of tweet is likely to be noisy because of their personal character. They may, however, bring in query terms used by a niche of people.

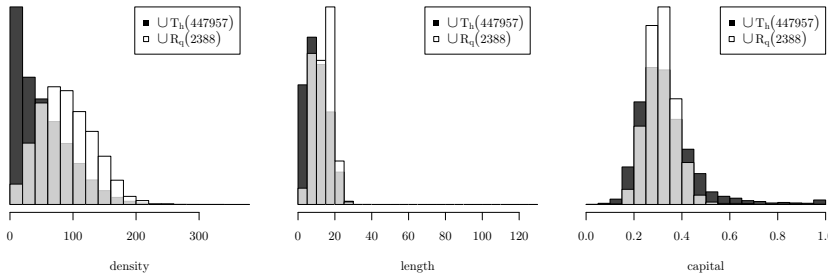


Figure 7.1: **Distribution of (Left) density scores, (Center) tweet length, and (Right) capitalization. Where the histograms for the tweets associated with hashtags (dark grey) and for the TREC MB 2011 relevant tweets (white) overlap, the color is light grey.**

Tweet length Document length has been shown to matter in retrieval scenarios [186]. Short tweets are less likely to contain terms useful for query simulation, see Figure 7.1 (Center) for the distribution of tweet length. Tweet length is measured in words. Note that even though a tweet should contain at most 140 characters, we observe a few outlier tweets with many words.

Density A direct measure for probing a tweet's content quality is the density score [132]. Density is defined as the sum of tf-idf values of non-stopwords, divided by the number of stopwords they are apart, squared:

$$Density(\tau) = \frac{K}{K-1} \sum_{k=1}^{K-1} \frac{weight(w_k) + weight(w_{k+1})}{distance(w_k, w_{k+1})^2},$$

where K is the total number of non-stopwords terms in tweet τ, w_k and w_{k+1} are two adjacent keywords in τ. $weight(\cdot)$ denotes the term's tf-idf score, and $distance(w_k, w_{k+1})$ denotes the distance between w_k and w_{k+1} in number of stopwords. Figure 7.1 (Left) shows the distribution of density scores of tweets.

Capitalization The textual quality of tweets can partially be captured through the use of capitalization [229]. Words in all capitals are considered shouting and an indication of low quality. The ratio of capitals may indicate the quality of the text. Figure 7.1 (Right) shows the distribution of the fraction of capital letters over tweet length.

Direct A tweet is *direct* if it is meant to be a "private" message to another user (i.e., the tweets starts with *@user*).

7.3 Generating queries

In web search, Asadi et al. [14] use anchor texts of in-links of documents as a source for query terms. In our microblog setting, such anchor texts are not available. Instead we resort to sampling query terms from R_h, the set of relevant tweets sampled from T_h, which holds the tweets that contain hashtag h. This is a challenging step in the process of automatically generating pseudo test collections. Considerations that play a role in microblog search are the presence of many spelling mistakes and intentional variations of word spellings indicative of online chatter (e.g., "looooooool"). As a first step to ensure a basic level of quality for our query terms, we consider only terms that occur in at least ten documents. As a second step, we discard terms which are equal to the hashtag h up to the '#' character. This is because all tweets in R_h contain the hashtag, making the query very easy for all rankers, which then leads to a learning to rank method having a hard time distinguishing between rankers.

Next, we need a method to determine which of the remaining candidate terms to sample. Azzopardi et al. [15] propose several methods for sampling query terms from single documents, for known-item search. Because in our setting we sample from a set of documets, their methods are not directly applicable. From the three approaches they proposed: (1) sampling random terms, (2) sampling popular terms, and (3) sampling discriminative terms, the latter method produced best results. Their discriminative sampling method rewards terms that have a high inverse document frequency (idf). In our setting, it is not clear that we want to favor terms with a high idf. Such terms could include very rare terms, which are rare even in R_h. Rather, we want to find terms which are over-represented in R_h with respect to the rest of the document collection. One technique which has been used for this purpose is the log-likelihood ratio test (LLR) [182]. We discussed this technique in detail in the previous chapter. It has the nice property that it is robust in situations where one of the corpora in which term frequencies are compared is much smaller than the other, which is the case in our setting ($|R_h| \ll |T \setminus R_h|$). It is also robust to low expected term frequencies, which also arise in the small set of tweets R_h [182].

For every hashtag h, we rank the terms occurring in R_h in descending order of their log-likelihood ratio score. We generate queries that consist of the top-K ranked terms. For all pseudo test collections described in Section 7.2 we set $K = 10$. For our most promising method, we examine the impact of this parameter by generating queries of length 1, 2, 3, 5, and 20.

Table 7.1: The retrieval algorithms we tuned, along with the parameter values we used. The ranges are based on similar experiments in the literature when possible, cf. the last column.

	Ranker	Description	Parameter	Values	cf. Literature
Indri 5.1	LM	Language modeling with Dirichlet smoothing	μ	$\{50, 150, \ldots, 10050\}$	[242]
	Tf-idf	Indri's implementation of tf-idf	k1 b	$\{0.2, 0.4, \ldots, 3\}$ $\{0, 0.05 \ldots, 1\}$	[183]
	BM25	BM25 [186]	k1 b k3	$\{0, 0.2, \ldots, 3\}$ $\{0, 0.05 \ldots, 1\}$ $\{0\}$	[183]
	BOW	Boolean ordered window	Window size	$\{1, 2 \ldots, 15, \inf\}$	
	BUW	Boolean unordered window	Window size	$\{1, 2 \ldots, 15, \inf\}$	
Terrier 3.5	Tf-idf	Terrier's implementation of tf-idf	b	$\{0, 0.05 \ldots, 1\}$	[183]
	PL2	A Divergence from Randomness (DFR) model [5]	c	$\{0.5, 1, 5, 10\}$	[55]
	DFR-FD	Terrier's DFRee model with a document score modifier that takes into account co-occurence within a window	Window size	$\{2, 3 \ldots, 15\}$	[171]
	PRF	Terrier PRF implementation	No. documents No. terms	$\{1, 5, 10, 20, 30, 50\}$ $\{1, 5, 10, 20, 30, 50\}$	[139]

7.4 Experimental setup

In this section, we first describe which experiments we perform. We then follow with a description of the ingredients used in these experiments: the dataset and preprocessing, the L2R setup, and the evaluation metrics.

7.4.1 Experiments

To answer RQ5, we perform a number of experiments. These are separated in two main sets: parameter tuning experiments and L2R experiments. In a third set of experiments we vary two parameters of our PTC generation recipes to observe the sensitivity of our methods with regard to these parameter settings.

Parameter tuning Recall that in RQ5 we asked how succesful free parameters of individual (non L2R) retrieval algorithms, such as language modeling, can be tuned on our PTCs. To answer this question we do parameter sweeps for some retrieval runs.[1] On different (pseudo) test collections, see Table 7.1 for details. More specifically, we ask: (1) What is better in terms of retrieval performance: tuning on a different hand-crafted test collection or tuning on a pseudo test collection? We answer by calculating how far performance obtained by tuning on either collection is from optimal performance for each retrieval model. We refer to this quantity as the *expected loss*. (2) Do scores between a pseudo test collection and a hand-crafted test collection correlate better than scores between editorial test collections? For each retrieval algorithm, for each parameter setting (cf. Table 7.1), we compute a retrieval performance score on both the PTC and the hand-crafted test collection. This gives two lists of observations for each retrieval algorithm. We use Kendall's tau to compute a correlation between performance scores in these two lists. Then, finally, we average the correlations of all retrieval algorithms to estimate the degree to which a PTC gives similar parameter tuning results as a hand-crafted test collection. Because the number of observations in the lists differs between retrieval algorithms, we refrain from reporting on p-values here. To handle ties, we opt for Kendall's tau-b. Given two lists of observations, it is defined as:

$$\frac{P - Q}{\sqrt{(N - T)(N - U)}}, \tag{7.5}$$

where P is the number of concordant (identically ordered) pairs in the two lists, Q is the number of discordant pairs in them, N is the total number of pairs, T is the number of tied pairs in the first list, and U is the number of tied pairs in the second list. This version of Kendall's tau has the property that if one of the lists contains only ties, the fraction is undefined. In such cases, we report the correlation is not available (NA).

A subtle point concering the goal of our parameter tuning experiments is the following. The ranges of each of the parameters in Table 7.1 are based on reasonable values tried in the literature in various domains. From the perspective of the microblog search domain, some ranges are quite naive, e.g., already in 2011 it was shown that document

[1] The retrieval models are the same as in the previous chapter, except for the missing Indri PRF implementation, which, for some PTCs, terminated with an error

Algorithm 7.4: Training an LTR system on a pseudo test collection, and testing it on a hand-crafted test collection

```
for i = 1 → N do
    // --- Training phase: ---
    Generate the pseudo test collection;
    for each ranker do
        Sweep parameters on the pseudo test collection;
        Randomly sample parameter vector to use from winners;
    end
    for q ∈ Q do
        Merge the ranked lists into M_q;
        Let positive training examples ← R_{q,h} ∩ M_q;
        Randomly sample |R_{q,h}| negative training examples from M_q \ R_{q,h};
    end
    Learn a LTR model on the training set;
    // --- Testing phase: ---
    for each ranker do
        Run on test topics using the sampled parameter vector;
    end
    Run the learned LTR model on the test set;
end
```

length normalization is not beneficial for microblog search [161], and consequently we could have restricted values for the b parameter of the Tf-idf and BM25 models to values close to zero. However, the point here is not to maximize microblog search performance, primarily. The main point is to show that parameters of general retrieval models can be tuned on a PTC generated for the microblog search domain. To show this, we have to include "bad" values for parameters in the tuning ranges.

Learning to rank In RQ5, we ask what yields best performance on a hand-crafted test collection: training on a different hand-crafted test collection or training on a PTC? We also ask what the relative merit of our PTC generation methods is, and how the choice of L2R algorithm affects our findings. To answer these questions, we train three different L2R algorithms. For each algorithm, we optimize a model on each PTC and on each of two hand-crafted test collections. We then compare the test performance of all these models on the same two hand-crafted test collections.

Parameter sensitivity. We also analyze parameter sensitivity of our methods, focusing on two parameters. First, in generating *Hashtag-TI* (Section 7.2), we keep the best $X = 50\%$ percent of tweets. How sensitive are our results to this method to different values for X? We try these values: 20, 40, 60, and 80. Second, in all our experiments, when we generate queries, we keep the top 10 terms of the ranking produced by LLR (Section 7.3). For our best PTC, we ask how parameter tuning results and learning to rank performance is influenced by different query length. We try query lengths 1, 2, 3, 5, and 20.

Table 7.2: Statistics for pseudo test collection generated from our methods and the TREC Microblog 2011 track.

Collection	Topics				Relevant documents			
	#	Max	Min	Avg. length	#	Max	Min	Avg.
TREC MB 2011	49	6	1	3.4	2,965	178	1	60.5
TREC MB 2012	59	7	1	2.9	6,286	572	1	106.5
Random-1	1,888	10	10	10	462560	245	245	245.0
Hashtags	1,888	10	10	10	462,013	16,105	50	244.7
Hashtags-T	891	10	10	10	212,377	9,164	50	238.4
Hashtags-TI	481	10	10	10	98,586	4,949	50	205.0
H-TI-X20	175	10	10	10	32,221	3661	50	184.1
H-TI-X40	392	10	10	10	75,287	4804	50	192.1
H-TI-X60	576	10	10	10	121,639	5277	50	211.2
H-TI-X80	740	10	10	10	166,489	6956	50	225.0

7.4.2 Dataset and preprocessing

We use the publicly available dataset from the TREC 2011 and 2012 Microblog tracks, which have 49 and 59 queries, respectively. The document collection covers two weeks of Twitter data, from January 24, 2011–February 8, 2011, consisting of approximately 16 million tweets. We perform a series of preprocessing steps on the content of tweets. We discard non-English tweets using a language identification method for microblogs [44]. Exact duplicates are removed; among a set of duplicates the oldest tweet is kept. Retweets are discarded; in ambiguous cases, e.g., where comments were added to a retweet, we keep the tweet. Punctuation and stop words are removed using a collection-based stop word list, but we keep hashtags without the '#' character. After preprocessing we are left with 4,459,840 tweets, roughly 27% of all tweets. Due to our aggressive preprocessing, we miss 19% of the relevant tweets per topic, on average. Our 10 retrieval models avoid using future evidence by using per topic indexes. For completeness, we note that our stopword list and the idf-weights in the density feature were computed on the entire collection. Pseudo test collections and both TREC microblog test collections also contain tweets from the entire collection. For generating queries, we index the collection with Lucene without stemming or stopword removal.

Table 7.2 lists statistics of the pseudo test collections generated with the methods described in Section 7.2, as well as statistics of the collections generated by choosing different values for X. The Hashtags-TI-QL$\{1, 2, 3, 5, 20\}$ pseudo test collections are of the same proportions as *Hashtags-TI*, apart from the query length. We list only one of fifteen Random collections, but these are all of the same proportions: about as large as the *Hashtags* collection.

7.4.3 Learning to rank

We follow a two-step approach to learning to rank, outlined in Algorithm 7.4. First, we run several retrieval algorithms, then we re-rank all retrieved tweets. For retrieval, we use the retrieval algorithms listed in Table 7.1, optimized for MAP [143, 184] after tuning on a training collection. In case of ties among parameter vectors for a ranker, we randomly sample a parameter vector. We also use a parameter free retrieval algorithm, DFRee [5]. For re-ranking, we compute three groups of features.

Query-tweet features These are features that have different values for each query-tweet pair. We subdivide these as follows. *Rankers*: the raw output of each retrieval algorithm. For LM, BOW and BUW we transform the raw output X by taking the exponent: $\exp(X)$. *Ranker meta features*: the number of rankers that retrieved the tweet, the maximal, average, and median reciprocal rank of the tweet over all rankers. *Recency*: query-tweet time difference decay, computed as $\exp(t(\tau) - q_t)$, where $t(\tau)$ is the timestamp of the tweet and q_t the timestamp of the query. We linearly normalize query-tweet features over all retrieved tweets for the query.

Query features These are features which have the same value for every retrieved tweet within the same query. We use *Query clarity*, a method for probing the semantic distance between the query and the collection [63]. We linearly normalize query features over the set of retrieved tweets for all queries.

Tweet features These are features that have the same value for each tweet independent of the query. We use the *Quality criteria* listed in Section 7.2: *link*, *mentions*, *tweet length*, *density*, *capitalization*, and *direct*. We transform the feature values for tweet length and capitalization, replacing them with the absolute difference from the median value for those features. This was done to obtain features more suitable for a linear classifier. Intuitively, good values for tweet length and capitalization are close to the median: not too long and not too short (see also Figure 7.1). We linearly normalize tweet features over the set of retrieved tweets for all queries.

To build a training set, one needs positive and negative training examples. Let $q \in Q$ be a query from the training collection, R_q the set of relevant tweets for query q, and M_q the set of all retrieved tweets for q. Then, for each query in the training collection we use $R_q \cap M_q$ as positive examples. To have a balanced training set, we randomly sample $|R_q|$ tweets as negative training examples from $M_q \setminus R_q$.

Next, we feed the training set to three state of the art learners: (1) Pegasos SVM [197, 202],[2] (2) Coordinate ascent [151], (3) RankSVM [111].[3][4] We used Pegasos with the same hyperparameter settings as in Chapter 6. We set coordinate ascent to optimize for MAP with ϵ = 0.0001 (recommended in [151]), and a maximum step size of 3. We use line search to optimize each feature with uniform initialization and consider only positive

[2] `http://code.google.com/p/sofia-ml/`

[3] `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

[4] We also considered (4) RT-Rank [157], but since it performed poorly in preliminary experiments, we leave it out from our report.

feature weights without projecting points on the manifold, cf. [151]. We used RankSVM with default settings for its hyperparameters. It is possible to tune hyperparameters for the L2R algorithms as well, e.g., by subdividing training sets further, performing internal cross validation. This may lead to further improvements.

Recall that training sets are compiled using tuned rankers and that in case of ties between different parameter vectors for a ranker, a random vector is selected. When compiling test sets for TREC MB 2011 and TREC MB 2012 to evaluate the utility of a training set, we use the exact same parameter vectors, so that the same set of features are used for training and testing.

Algorithm 7.4 has randomness in several stages: (1) when generating the pseudo test collection (only in the case of the Random collections), (2) when sampling a winning parameter setting for each feature, (3) when randomly sampling negative training examples, and (4) during model learning. To obtain a reliable estimate of of the performance when training on a pseudo test collection, this procedure is repeated $N = 10$ times, each time generating a new pseudo test collection (in the case of the Random test collection), selecting random parameter vectors, selecting random negative training examples, and training an LTR model.

7.4.4 Evaluation

We report on precision at 30 (P30) on binary relevance judgments. We choose P30 because it has been one of the main metrics in both the TREC 2011 and 2012 Microblog track. We also report on MAP, as it is a well understood and commonly used evaluation metric in information retrieval, allowing us to better understand the behavior of our pseudo test collections. Note that in the 2011 task, tweets had to be ordered by their publication date instead of by their relevance. Many top performing systems treated the task as normal relevance ranking and cut off their ranked lists at rank 30 [166]. In the 2012 track organizers decided to focus on ranking by relevance again, which is what we will focus on.

Testing for statistical significance For each training collection, we run Algorithm 7.4 $N = 10$ times, giving rise to N scores for each topic, for each collection. We report average performance and sample standard deviation over these iterations. To also gain insight if any differences between a pair of training collections would be observed on different microblog topics from the same hypothetical population of topics, we proceed as follows. We pick for each collection the iteration of Algorithm 7.4 which had the smallest *training* error on that collection. In a benchmark such as the TREC microblog track, this is the run one would most likely submit. Then, we do a paired t-test over differences per topic and report the obtained p-values. Statistically significant differences are marked as $^{\blacktriangle}$ (or $^{\blacktriangledown}$) for significant differences for $\alpha = .001$, or $^{\triangle}$ (and $^{\triangledown}$) for $\alpha = .05$.

7.5 Results and analysis

First, we report on our parameter tuning results; then on our learning to rank results. We also analyze parameter sensitivity with regard to the percentage of interesting tweets kept

Table 7.3: For Tf-idf (Indri), and P30, Kendall's tau correlations of parameter sweeps on several pseudo test collections with sweeps on TREC MB 2011 and 2012 collections. Kendall's tau between sweeps over TREC MB 2011 and 2012 is 0.85.

Tune on	TREC MB 2011	TREC MB 2012
Random-1	0.18	0.20
Hashtags	0.87	0.86
Hashtags-T	0.90	0.86
Hashtags-TI	0.90	0.86

Table 7.4: For language modeling (LM), and P30, Kendall's tau correlations of parameter sweeps on several pseudo test collections with sweeps on TREC MB 2011 and 2012 collections. Kendall's tau between sweeps over TREC MB 2011 and 2012 is 0.61.

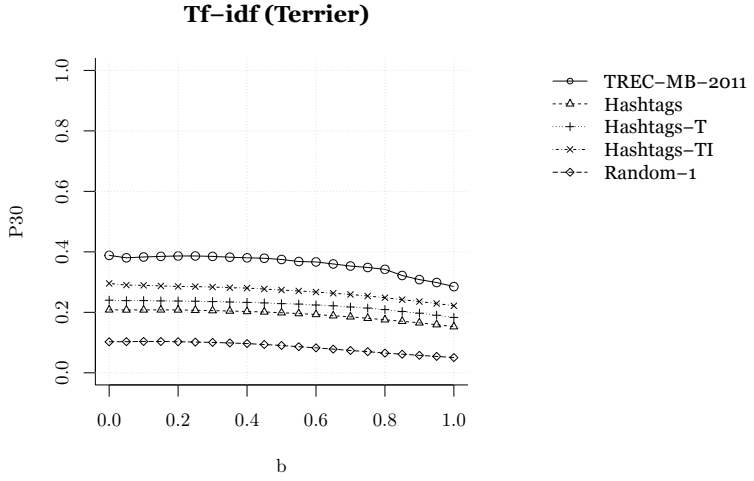
Tune on	TREC MB 2011	TREC MB 2012
Random-1	-0.87	-0.65
Hashtags	0.65	0.78
Hashtags-T	0.69	0.77
Hashtags-TI	0.58	0.79

and query length.

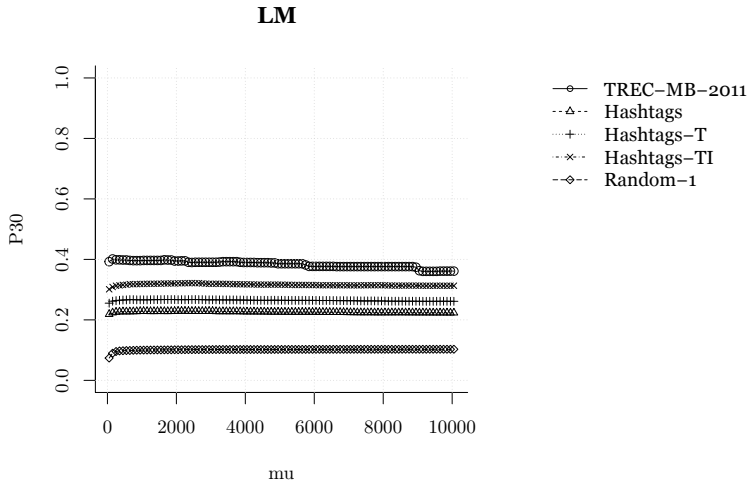
7.5.1 Parameter tuning results

The main outcomes in this section will be correlations, to answer the question whether relative performance of parameter values on pseudo test collections correlates with relative performance of the same values on a hand-crafted collection. We begin with two case studies to gain a better understanding of the behavior of our pseudo text collections. We sweep (a) the document length normalization parameter b for Terrier's tf-idf implementation (Figure 7.2a), and (b) μ for Indri's implementation of language modeling with Dirichlet smoothing (Figure 7.2b). We only include one of our ten random pseudo test collections; all random collections behave similarly, for all retrieval systems and metrics.

Figure 7.2a shows that on the TREC MB 2011 collection there is a general trend to prefer lower values of b, possibly because of the very small average document length, which, in turn, renders the deviation from the average length close to one. All pseudo test collections capture this trend, including the Random-1 pseudo test collection. The curves of the pseudo test collections are smoother than the curve obtained when tuning on the TREC MB 2011 topics; this is because the pseudo test collections have far more test topics. Pseudo test collections show differences in absolute scores, but most importantly, we are interested in whether pseudo test collection predictions that one parameter vector is better than another correlate to such predictions of hand-crafted collections. Kendall's tau expresses exactly that correlation, see Table 7.3: correlations are high across the board. In addition, we want to know the following: if we sample a random parameter vector from those predicted to yield optimal performance on a pseudo test collection,



(a)



(b)

Figure 7.2: Sweeping the b parameter of Tf-idf (Terrier) (7.2a) and the μ parameter of LM (7.2b). The x-axes have parameter values, the y-axes average P30 over the topics of the respective tuning collection.

Table 7.5: For Tf-idf (Indri), expected loss in P30 performance of parameter sweeps on several training collections compared to optimal performance on TREC MB 2011 and 2012 collections. When only one parameter setting achieves optimal performance on a PTC, we list the loss with regard to the optimal setting on TREC MB 2011 or 2012. If there are multiple, we list the expected (average) loss, followed by the sample standard deviation, and finally their number, in brackets.

Tune on	TREC MB 2011	TREC MB 2012
TREC MB 2011	-	0.003
TREC MB 2012	0.006±0.005 (3)	-
Random-1	0.029	0.024
Hashtags	0.012±0.002 (2)	0.002±0.002 (2)
Hashtags-T	0.002	0.000
Hashtags-TI	0.002	0.000

Table 7.6: For language modeling (LM), expected loss in P30 performance of parameter sweeps on several training collections compared to optimal performance on TREC MB 2011 and 2012 collections. When only one parameter setting achieves optimal performance on a PTC, we list the loss with regard to the optimal setting on TREC MB 2011 or 2012. If there are multiple, we list the expected (average) loss, followed by the sample standard deviation, and finally their number, in brackets.

Tune on	TREC MB 2011	TREC MB 2012
TREC MB 2011	-	0.010
TREC MB 2012	0.006	-
Random-1	0.039±0.001 (11)	0.014±0.001 (11)
Hashtags	0.010	0.001
Hashtags-T	0.013±0.004 (2)	0.001±0.001 (2)
Hashtags-TI	0.014	0.002

Table 7.7: Over all retrieval models, the average Kendall's tau correlations between P30 performance scores of parameter sweeps on several pseudo test collections and P30 performance scores of parameter sweeps on TREC MB 2011 and 2012 collections. Kendall's tau between sweeps over TREC MB 2011 and 2012 is 0.80. The average tau over retrieval algorithms is given, followed by the sample standard deviation, and the number of retrieval algorithms for which tau was undefined.

Tune on	TREC MB 2011	TREC MB 2012
Random-1	0.27±0.62 (2 NA)	0.32±0.56 (1 NA)
Hashtags	0.78±0.20 (1 NA)	0.70±0.35 (1 NA)
Hashtags-T	0.80±0.20 (1 NA)	0.78±0.30 (1 NA)
Hashtags-TI	0.75±0.26 (1 NA)	0.81±0.23 (1 NA)

Table 7.8: Over all retrieval models, average expected loss in P30 performance of parameter sweeps on several training collections compared to optimal performance on TREC MB 2011 and 2012 collections.

Tune on	TREC MB 2011	TREC MB 2012
TREC MB 2011	-	0.003±0.006
TREC MB 2012	0.003±0.003	-
Random-1	0.021±0.021	0.013±0.012
Hashtags	0.005±0.006	0.004±0.006
Hashtags-T	0.004±0.005	0.002±0.005
Hashtags-TI	0.004±0.005	0.002±0.005

what will be the expected loss with regard to optimal performance on a hand-crafted collection? Table 7.5 provides these quantities. If there are multiple parameter settings with the same optimal performance, the sample standard deviation is reported along with the expected loss, and the number of optimal parameter settings is given in brackets. We observe that the expected loss is low. All pseudo test collections can be used to reliably tune the b parameter of Terrier's TF-IDF, even the Random-1 collection.

Turning to a second case study, Indri's language modeling algorithm with Dirichlet smoothing, we see a different picture (Figure 7.2b). The TREC MB 2011 topics show a slightly decreasing trend for larger μ values. We believe this is due to the short document length; tweets after processing are few terms long, and therefore even small μ values overshadow the document term probability with the background probability. This trend is not entirely captured by the pseudo test collections. All have a short increase for low values of μ which is much less pronounced in the TREC MB 2011 curve. After that, all except the Random-1 collection show a decline, if only in the third digit. Correlations are fair (Table 7.4), but the Random-1 collection fails here. Expected loss is low across the board (Table 7.6).

Looking at the big picture, we average the correlations and expected loss figures in Tables 7.7 and 7.8 over all nine retrieval models from Table 7.1. For the hashtag based

collections, correlations are high, with a large variance over systems. Expected loss is low. This indicates that pseudo test collections can be used to reliably and profitably tune parameters for a variety of well established retrieval algorithms, with more or less success depending on which model is being tuned. Tuning retrieval models on all hashtag based pseudo test collections is about as reliable as tuning on hand-crafted test collections. For most retrieval algorithms, a Random collection cannot be recommended for tuning. Thus, the idea of grouping tweets by hashtag has value for creating pseudo test collections for tuning retrieval algorithms.

7.5.2 Learning to rank results

In this section we evaluate the usefulness of our pseudo test collections by training several learning to rank algorithms on them. In Tables 7.9 and 7.10, we report P30 and MAP performance on the TREC 2011 Microblog track topics. We compare training on our PTCs with training on the TREC 2012 Microblog track topics and indicate significant differences. In Tables 7.11 and 7.12, we report P30 and MAP performance on the TREC 2012 Microblog track topics, and compare training on our PTCs with training on the TREC 2011 Microblog track topics.

A first brief glance at all four tables tells us that training on a hand-crafted test collection is in most cases (but not all) the best strategy. Still, if training on a PTC is not substantially and significantly worse, we may conclude that in the abscence of training data, pseudo test collections are a viable alternative.

A second glance at all four tables shows us that training on the Random PTC is always significantly outperformed by training on hand-crafted collections. Still, in most cases, there is a hashtag based PTC on which training yields performance on par with training on hand-crafted collections. This shows that there is indeed considerable merit in our idea of using hashtags to group tweets by topic.

Another phenomenon we can observe in all tables is that Pegasos has remarkably stable performance over pseudo test collections, compared to the other two learning to rank algorithms. With the exception of Hashtags-TI-QL1 and Random, it seems to be able to exploit the structure in any PTC to learn a function that yields performance comparable to a function learned on hand-crafted training data. RankSVM, on the other hand, has unstable performance. Especially in Table 7.12 it refuses to work on anything but manually obtained ground truth. Coordinate Ascent, a remarkably simple learning to rank algorithm holds the middle ground. When queries are too short, as in Hashtags-TI-QL1, Hashtags-QL2 and Hashtags-TI-QL3 performance detoriates. When subsamples of tweets become too small (Hashtags-TI-X20) the same happens.

So which PTC is the best? All PTCs are significantly outperformed by training on a hand-crafted collection in at least one of the conditions. Hashtags-TI-X60 and Hashtags-TI-X80 both yield best results in one case. Also, like Hashtags, Hashtags-T and Hashtags-TI are significantly outperformed in only a small number of cases. Once we zoom in on the best performing L2R algorithm, Pegasos, the situation becomes more clear. Hashtags-TI-QL5 is always among the top scorers, and it is never significantly beaten by training on hand-crafted test collections. Still, the differences among PTCs are small.

Learning to rank only makes sense if improvements over individual retrieval algo-

Table 7.9: P30 performance on TREC MB 2011 topics for various LTR algorithms, trained on different collections. Features are tuned on MAP. Best run in bold. Indicated statistically significant differences are with regard to training on TREC-MB-2012. The best individual tuned ranker, DFR-FD, achieved 0.416.

Train on	Pegasos	RankSVM	CA
TREC-MB-2012	**0.446±0.001**	**0.436±0.004**	**0.447±0.004**
Random	0.401±0.004$^{\triangledown}$	0.356±0.011$^{\blacktriangledown}$	0.378±0.006$^{\blacktriangledown}$
Hashtags	0.437±0.002	0.430±0.004	0.434±0.002
Hashtags-T	0.438±0.001	0.426±0.002	0.434±0.003
Hashtags-TI	0.437±0.001	0.406±0.007$^{\triangledown}$	0.429±0.002$^{\triangledown}$
Hashtags-TI-X20	0.432±0.001	0.265±0.016$^{\blacktriangledown}$	0.383±0.016$^{\blacktriangledown}$
Hashtags-TI-X40	0.435±0.001	0.361±0.006$^{\blacktriangledown}$	0.427±0.001$^{\triangledown}$
Hashtags-TI-X60	0.438±0.001	0.415±0.004$^{\triangledown}$	0.431±0.003$^{\triangledown}$
Hashtags-TI-X80	0.438±0.001	0.427±0.003	0.435±0.002
Hashtags-TI-QL1	0.189±0.135$^{\blacktriangledown}$	0.034±0.015$^{\blacktriangledown}$	0.293±0.107 $^{\blacktriangledown}$
Hashtags-TI-QL2	0.435±0.002	0.244±0.024$^{\blacktriangledown}$	0.421±0.047 $^{\blacktriangledown}$
Hashtags-TI-QL3	0.443±0.001	0.240±0.012$^{\blacktriangledown}$	0.427±0.023 $^{\blacktriangledown}$
Hashtags-TI-QL5	0.443±0.001	0.259±0.008$^{\blacktriangledown}$	0.428±0.005 $^{\triangledown}$
Hashtags-TI-QL20	0.438±0.001	0.320±0.007$^{\blacktriangledown}$	0.432±0.004

rithms can be obtained. Tables 7.13 and 7.14 show performance of the retrieval models we use as features. In the great majority of cases our hashtag based PTCs outperform the best feature. The Random collection never achieves this.

7.6 Discussion

Following the results of our experiments we list three main observations: (1) The Random pseudo test collection performs significantly worse than hand-crafted collections in the retrieval experiments and shows low correlation to these collections in the tuning phase. (2) The top pseudo test collections are not significantly worse than hand-crafted collections and show high correlation when tuning parameters. (3) Differences between various pseudo test collections on retrieval effectiveness are small.

Combining the top two observations leads us to conclude that our approach to constructing pseudo test collections works. We can successfully use pseudo test collections, as long as we find appropriate surrogate relevance labels. Why are these findings important? To train learning to rank methods on microblog retrieval tasks we do not have to invest in manual annotations but can use hashtags for creating training examples. Pseudo test collections can also be used successfully for tuning parameters of retrieval models.

The third observation is that the differences between our pseudo test collections are limited. More advanced methods for selecting tweets and hashtags result in performance that is only sporadically better than the naive baseline method, which treats all tweets and hashtags equally. We can look at this from two angles. (1) Collection construction:

Table 7.10: MAP performance on TREC MB 2011 topics for various LTR algorithms, trained on different collections. Features are tuned on MAP. Best performance per LTR algorithm in bold. Indicated statistically significant differences are with regard to training on TREC-MB-2012. The best individual tuned ranker, Tf-idf (Indri), achieved 0.357.

Train on	Pegasos	RankSVM	CA
TREC-MB-2012	**0.388±0.001**	0.362±0.003	**0.387±0.003**
Random	0.348±0.004▼	0.294±0.009▼	0.319±0.009▼
Hashtags	0.374±0.001	0.360±0.005	0.385±0.001
Hashtags-T	0.379±0.000	0.362±0.005	0.386±0.000
Hashtags-TI	0.383±0.001	0.344±0.003	0.377±0.002
Hashtags-TI-X20	0.376±0.001▽	0.198±0.015▼	0.332±0.013▼
Hashtags-TI-X40	0.380±0.001	0.300±0.007▽	0.373±0.001▽
Hashtags-TI-X60	0.383±0.001	0.352±0.005	0.381±0.003
Hashtags-TI-X80	0.381±0.000	**0.363±0.002**	0.384±0.002
Hashtags-TI-QL1	0.141±0.125▼	0.020±0.009▼	0.202±0.119▼
Hashtags-TI-QL2	0.370±0.001▽	0.185±0.019▼	0.360±0.046▼
Hashtags-TI-QL3	0.379±0.000	0.166±0.011▼	0.366±0.026▼
Hashtags-TI-QL5	0.383±0.001	0.165±0.007▼	0.372±0.006▽
Hashtags-TI-QL20	0.383±0.001	0.231±0.008▼	0.381±0.002

Table 7.11: P30 performance on TREC MB 2012 topics for various LTR algorithms, trained on different collections. Features are tuned on MAP. Best performance per LTR algorithm in bold. Indicated statistically significant differences are with regard to training on TREC-MB-2011. The best individual tuned ranker, DFR-FD, achieved 0.351.

Train on	Pegasos	RankSVM	CA
TREC-MB-2011	**0.392±0.001**	**0.391±0.007**	0.392±0.007
Random	0.341±0.003▼	0.289±0.008▼	0.314±0.006▼
Hashtags	0.379±0.001	0.330±0.011▽	0.378±0.001▽
Hashtags-T	0.372±0.001▽	0.336±0.005▽	0.382±0.001
Hashtags-TI	0.381±0.001	0.326±0.007▽	0.393±0.002
Hashtags-TI-X20	0.377±0.001	0.248±0.013▼	0.353±0.008
Hashtags-TI-X40	0.379±0.001▽	0.294±0.009▼	0.389±0.001
Hashtags-TI-X60	0.379±0.001	0.348±0.003▽	**0.394±0.006**
Hashtags-TI-X80	0.373±0.001▽	0.337±0.006▽	0.388±0.007
Hashtags-TI-QL1	0.198±0.089▼	0.047±0.016▼	0.291±0.059▼
Hashtags-TI-QL2	0.374±0.002▽	0.231±0.016▼	0.377±0.035▼
Hashtags-TI-QL3	0.381±0.001	0.218±0.007▼	0.382±0.016▼
Hashtags-TI-QL5	0.385±0.001	0.248±0.005▼	0.391±0.004
Hashtags-TI-QL20	0.380±0.001	0.302±0.005▼	0.388±0.002

Table 7.12: MAP performance on TREC MB 2012 topics for various LTR algorithms, trained on different collections. Features are tuned on MAP. Best performance per LTR algorithm in bold. Indicated statistically significant differences are with regard to training on TREC-MB-2011. The best individual tuned ranker, DFR-FD, achieved 0.220.

Train on	**Pegasos**	**RankSVM**	**CA**
TREC-MB-2011	**0.245±0.000**	**0.245±0.004**	**0.246±0.004**
Random	0.207±0.001$^{\blacktriangledown}$	0.159±0.005$^{\blacktriangledown}$	0.176±0.006$^{\blacktriangledown}$
Hashtags	0.231±0.000$^{\triangledown}$	0.181±0.008$^{\blacktriangledown}$	0.224±0.001$^{\triangledown}$
Hashtags-T	0.227±0.001$^{\triangledown}$	0.193±0.004$^{\blacktriangledown}$	0.227±0.001$^{\triangledown}$
Hashtags-TI	0.234±0.001$^{\triangledown}$	0.172±0.005$^{\blacktriangledown}$	0.233±0.001$^{\triangledown}$
Hashtags-TI-X20	0.233±0.001$^{\triangledown}$	0.143±0.007$^{\blacktriangledown}$	0.210±0.005$^{\blacktriangledown}$
Hashtags-TI-X40	0.233±0.000$^{\triangledown}$	0.158±0.005$^{\blacktriangledown}$	0.230±0.001$^{\triangledown}$
Hashtags-TI-X60	0.233±0.000	0.196±0.002$^{\blacktriangledown}$	0.234±0.003$^{\triangledown}$
Hashtags-TI-X80	0.230±0.000$^{\triangledown}$	0.189±0.004$^{\blacktriangledown}$	0.230±0.004$^{\triangledown}$
Hashtags-TI-QL1	0.106±0.060$^{\blacktriangledown}$	0.026±0.008$^{\blacktriangledown}$	0.165±0.048$^{\blacktriangledown}$
Hashtags-TI-QL2	0.229±0.001$^{\triangledown}$	0.123±0.011$^{\blacktriangledown}$	0.233±0.021$^{\blacktriangledown}$
Hashtags-TI-QL3	0.234±0.000	0.106±0.004$^{\blacktriangledown}$	0.231±0.015$^{\blacktriangledown}$
Hashtags-TI-QL5	0.235±0.000	0.118±0.004$^{\blacktriangledown}$	0.233±0.003$^{\triangledown}$
Hashtags-TI-QL20	0.231±0.000$^{\triangledown}$	0.162±0.003$^{\blacktriangledown}$	0.228±0.001$^{\triangledown}$

Table 7.13: P30 performance on 2011 and 2012 topics for retrieval models for which MAP was tuned on 2011 topics, ordered by 2012 performance. The only parameter free retrieval model, DFRee, achieved 0.416 on the 2011 topics, and 0.346 on the 2012 topics.

Model	2011	2012
DFR-FD (Terrier)	0.416	0.351
Tf-idf (Terrier)	0.397	0.348
PL2 (Terrier)	0.406	0.336
PRF (Terrier)	0.391	0.335
Tf-idf (Indri)	0.413	0.331
Okapi (Indri)	0.409	0.329
LM (Indri)	0.404	0.325
BUW (Indri)	0.128	0.154
BOW (Indri)	0.094	0.134

Table 7.14: MAP performance on 2011 and 2012 topics for retrieval models for which MAP was tuned on 2011 topics, ordered by 2012 performance. The only parameter free retrieval model, DFRee, achieved 0.351 on the 2011 topics, and 0.216 on the 2012 topics.

Model	2011	2012
DFR-FD (Terrier)	0.352	0.220
Tf-idf (Terrier)	0.352	0.215
PL2 (Terrier)	0.348	0.209
PRF (Terrier)	0.335	0.201
Tf-idf (Indri)	0.357	0.194
Okapi (Indri)	0.354	0.191
LM (Indri)	0.346	0.188
BUW (Indri)	0.075	0.069
BOW (Indri)	0.060	0.061

We can limit time spent on constructing a smooth and interesting pseudo test collection by substituting more advanced methods (Hashtags-T and -TI) with the naive Hashtags method. Using the naive method is faster and results in a larger collection, with similar results. (2) Training volume: Investing in obtaining more interesting tweets and hashtags using our more advanced methods substantially reduces the number of queries in our collections. While the naive Hashtags uses over 1,800 queries and 460,000 relevant tweets, the other methods use only 890 (Hashtags-T) and 480 (Hashtags-TI) queries and equally reduced sets of relevant tweets. Training efficiency improves substantially by limiting the number of queries in the collections. In other words, we can choose between spending more time on constructing our collections, while reducing training time, or take a naive collection construction approach that results in larger collections and thus longer training times. A similar observation holds for the hand-crafted collections, which are the smallest collections (50–60 queries), but (supposedly) with the highest quality. An interesting follow-up experiment to shed further light on this trade-off would be to generate pseudo test collections that are both naive and small.

7.7 Conclusion

Summarizing, we have studied the use of pseudo test collections for training and tuning LTR systems for microblog retrieval. We use hashtags as surrogate relevance labels and generate queries from tweets that contain the particular hashtag. These pseudo test collections are then used for (1) tuning parameters of various retrieval models, and (2) training learning to rank methods for microblog retrieval. We explore three ways of constructing pseudo test collections, (1) a naive method that treats all tweets and hashtags equally, (2) a method that takes timestamps into account, and (3) a method that uses timestamps and selects only interesting microblog posts. We compare their performance to those of a randomly generated pseudo test collection and two hand-crafted collections.

Our pseudo test collections have high correlation with the hand-crafted collections in the parameter tuning phase, whereas the random collection has a significantly lower

correlation. In the LTR phase we find that in most cases our collections do not perform significantly worse than the hand-crafted collections, while the random collection does perform significantly worse.

Compared to the previous chapter, on PTCs for scientific literature search, the results in this chapter are somewhat more convincing. The retrieval algorithms were tuned, the L2R methods outperform all individual features, and in many conditions training on a PTC was found to yield performance on par with training on hand-crafted test collections.

Looking forward, we are interested in training on a mixture of hand-crafted and generated ground truth. Our work is related to creating ground truth in a semi-supervised way and we also aim to further explore this relation.

8 Conclusions

In this chapter, we first summarize our main findings by answering the research questions we posed in Chapter 1. We briefly recall each question, answer it, interpret that answer, and discuss ramifications. Next, we discuss directions for future work.

8.1 Main findings

In this thesis we studied several vertical search applications. In each of our studies, background knowledge about information needs and information objects played an important role. In Chapter 3, on people search, we focused on information needs. In particular, we automatically categorized queries into three classses [231]. Our research question here was:

RQ1 Is it possible to classify person name queries submitted to a people search engine as either low-profile, event-based and regular high-profile classes such that reasonable agreement with human classifying decisions is achieved? Among a set of features capturing online context of person names and interactions of people search engine users, which features have the largest influence on classification decisions?

We found that it is indeed possible to separate low-profile from high-profile queries (either event-based or regular) with reasonable accuracy: roughly on par with inter-annotator agreement. Best results were obtained by using a decision tree. The most important features, i.e., the first splits in the decision tree (that was trained on all data), are click volume in the last seven days and the number of bursts. When we perform a three-way classification, we see that in particular regular high-profile queries are hard to classify correctly, both in terms of recall and precision. The first splits in a decision tree trained on all data are the same as above. The news volume in the last seven days plays a role in separating event-based and regular high-profile queries.

The feature analysis results are reasonably intuitive. Most low-profile queries are simply not much searched for and clicked on, explaining the importance of the click volume feature in both trees. The two decision trees, corresponding to two-way and three-way classification, shown in Figures 3.3 and 3.4 resemble each other. This makes sense, since high-profile queries are the union of event-based and regular high-profile queries. Perhaps surprisingly, less mentions in the news are an indication of event-based

queries. Indeed, such queries are often connected with a single event. For example, the most common subclass observed in event-based queries in a people search log are names of people who recently passed away. More frequent occurences of a name in the news are an indication of a regular high-profile query: somebody most likely looking for a well-known public figure with this name. The features we used in the experiment can be grouped into features that are about search volume, click volume or news volume. Features from all groups appeared in the first few splits of the above decision trees. Within each group, only one feature played such a promiment role. It appears then that using all different kinds of available evidence is a good strategy to obtain reasonable results, spending only limited resources in design and computation time. Some caution must be used when drawing conclusions from the feature analysis. Since the two trees analyzed in Chapter 3 are trained on all labeled data, their generalization performance has not been evaluated. The number of misclassifications in the two trees seems low, it is possible that they overfit training data somewhat. However, we would not expect the two trees to overfit more than the trees that we evaluated in the ten-fold cross-validation, since the two analyzed trees are trained on more data.

One implication of our finding is that people search engines could use a query classifier to treat low-profile queries differently from high profile queries, for example, since this is possible with accuracy close to human agreement. Event-based queries can also be recognized reliably. For such queries, for example, news results could be featured prominently.

In Chapter 4, also on people search, we found that two types of documents in the document collection were a decisive factor in the task of people search result disambiguation. Social media profiles are so different from other search results that they should be treated separately. Our research question here was:

RQ2 State-of-the-art hierarchical aglommerative clustering (HAC) methods for clustering web search results for person name queries break on search results of a people search engine, which contain many social media profiles. Can we remedy this problem by treating social media profiles differently from regular web documents, clustering the two types of documents separately and then merging the clusterings back together?

We find that indeed a dramatic performance increase is achieved when social media profiles and other web documents are clustered separately from each other, with the clusterings subsequently being merged. Improving on this simple concept proves to be hard. The best performing clustering of social media profiles simply assigns each profile to a singleton cluster. Then simply adding these singleton clusters alongside the web-document clusters achieves strong performance. It is possible to improve over this method by iteratively assigning the singleton social media clusters to the most promising web-document clusters. The improvements are very small, but significant.

An underlying issue in all our results seems to be sparsity of features. The sparsity of text in social media profiles is one of the reasons their presence in a corpus harms clustering performance. The sparsity of many of the features we proposed for clustering profiles (cross-links, co-clicks, a match of a face recognition classifier in profile pictures, and so on) is part of the reason that the singleton clustering method was best for profiles.

Most of these features are strong evidence for clustering, intuitively. Indeed, they loose little precision to a singleton clustering. But they also do little for recall due to their sparsity.

The two-stage clustering method deals with the sparse text in profiles by refusing to cluster them based on textual evidence. While the method is conceptually simple, it was not obvious to us from the start. The magnitude of the improvement over treating all documents in the same way gives pause. It is likely that many document collections will contain distinct groups of documents. Treating different groups separately may be beneficial for various tasks, including clustering tasks, classification tasks and retrieval tasks. The method of iteratively merging social media clusters to web-document clusters—so as to gain more evidence on the way—yields only small improvements over simply taking the union of all clusters. But this need not discourage future research in this direction. Since we release a test collection, others are free to seek additional improvements. In time, many small, incremental improvements can be of practical value together.

In Chapter 5, on expert search, we performed a self-assessment experiment in which we asked experts to judge knowledge area profiles that we generated for them. The main aim of this study was to gain understanding of the performance of strong expert profiling algorithms. We asked:

RQ3 We ask experts to judge profiles we generate for them (judged system-generated profiles). What is the quality of our generated profiles? Which experts are hard to generate a profile for and why? Previously, evaluation of expert profiling algorithms had been done by using profiles of knowledge areas that experts selected from an alphabetical list (self-selected profiles). If we use judged system-generated profiles for evaluation, what are differences in evaluation outcomes? Why do these differences occur?

The quality of the generated profiles is indeed much larger than previous evaluation results would suggest [20]. During the self-assessment experiment experts judged relevant many knowledge areas that were not in their self-selected profile. Experts left useful free-text feedback on their generated profiles and by means of a content analysis we were able to find aspects that were highlighted by several experts. The most frequently reported issues all concerned the retrieved knowledge areas: missing a key one, retrieving nonsensical ones, redundant ones, and too general ones. An error analysis revealed that for experts who had written a short research profile, performance of expert profiling algorithms was significantly higher. Also, knowledge areas without an English translation are harder to retrieve. A study of differences in evaluation outcomes between the self-selected and the judged system-generated profiles showed that indeed systems are ranked rather differently. Most of this difference comes about through the additional knowledge areas that experts judged relevant in the self-assessment experiment.

The finding that experts judged as relevant many knowledge areas outside of their self-selected profile confirms that the self-selected profiles were sparse. Consequently, we recommend using the judged system-generated profiles for the evaluation of expert profiling systems. The system-generated profiles were generated by combining the results of eight variants of expert profiling algorithms which are all inspired by language modelling techniques. It is possible that new and very different algorithms may retrieve

knowledge area that were never judged by experts in our self-assessment experiment. To study the impact of this possible bias, researchers may use the self-selected profiles as well as the judged system-generated profiles: we release both sets of ground truth with the test collection. The finding that expert profiling is easier for researchers who have written a research profile, leads us to recommend experts to do this. The finding that knowledge areas without an English translation are harder to retrieve leads us to recommend keeping up-to-date a fully translated knowledge base.

It is interesting to note, finally, that our recommendations target different groups of people around knowledge intensive organizations: the experts (employees), the maintainers of the knowledge base, and researchers and designers of expertise retrieval algorithms. Ultimately, this should benefit the process of supporting knowledge exchange in an organization with expert profiling and expert finding algorithms.

In Chapter 6, on scientific literature retrieval, we used annotations of scientific articles, e.g., topical keywords. These had been added by curators of the literature collection we study, with the aim of fully indexing the collection. We used these annotations to generate an IR test collection automatically. IR test collections are used to evaluate or optimize retrieval algorithms, and contain at least queries, documents, and relevance assessments. We started by considering documents labeled with particular annotations relevant to a query that we subsequently generated. We called our automatically generated collections pseudo test collections (PTCs). We asked:

RQ4 Collections of scientific literature are sometimes indexed with different kinds of annotations. Can the result of these annotation efforts be used to generate a pseudo test collection (PTC)? When retrieval algorithms are evaluated using a PTC, are they ranked as they would be by a hand-crafted test collection? And when we test a L2R algorithm on a hand-crafted test collection, what is the best way to train it: on a different hand-crafted test collection, or on a PTC?

We proposed three strategies for sampling sets of documents to be considered relevant; these strategies use the different kinds of annotations. We formulated two approaches to generate a query for such a set of documents. The Cartesian product of these two leads to six different PTCs. We found that if we rank ten retrieval systems using each of our six PTCs, correlations with the system rankings obtained by evaluation on hand-crafted collections were all positive, but mostly small, with the exception of one PTC. When training a L2R algorithm on some PTCs and testing on one of our hand-crafted test collections, performance was not significantly worse than training on the other hand-crafted test collection. This constitutes no scientific proof that the two strategies lead to equivalent test performance, but it means we could not reject the null hypothesis that they are equivalent with strong certainty. Note that we looked for significance at a strict $\alpha = 0.001$ level, because many pairwise comparisons were made and otherwise the probability of Type I becomes large. When testing on the other hand-crafted test collection, training on the first hand-crafted collection was best.

The study was exploratory in nature. It would be too early to use the generated PTCs for benchmarking: the observed Kendall's tau correlations were too low and varied too much over PTCs. More analysis of how these differences come about would still be

required. But the results for using the PTCS for training were somewhat encouraging. Performance was not always significantly worse than training on hand-crafted collections. It is interesting in the L2R results that which query generation technique is best depends on what strategy is used for selecting documents as relevant documents. The opposite is also true. A final note on the results in Chapter 6 is that the L2R algorithm did not impress, be it trained on hand-crafted or pseudo test collections. The absolute performance of the algorithm was not our main concern here, however. The main concern was establishing which test collection was most suitable for training.

The implications of this work are that there is promise in the idea of generating training material in vertical search applications with document collections that have rich annotations. When targeting a smaller audience in a vertical search application, there are often few resources for creating training data by hand. Training on automatically generated training data can then be a sensible alternative, particularly if performance obtained by it can be expected to be close to what one could have achieved on hand-crafted ground truth.

In Chapter 7, on microblog search, we pursued a similar idea as above. The microblog search domain differs from the digital library domain: user generated content is much more noisy than the carefully written, edited, and indexed content in scientific literature collections. The annotations that we leveraged here were hashtags. This time, we focused exclusively on using PTCs for training L2R algorithms. We asked:

RQ5 Hashtags are used in microblog search to indicate that a tweet is part of a larger discussion. We assume that tweets sharing the same hashtag share the same topic, by and large. Can we build on this assumption to generate a PTC? And when we test an L2R algorithm on a hand-crafted microblog search collection, what yields better performance: training on a different hand-crafted collection, or training on a PTC? We consider three recipes for generating a PTC. What is their relative merit as training material? We consider three strong L2R algorithms. How will our findings vary with a different choice of L2R algorithms? And, how succesful can free parameters of individual retrieval algorithms such as language modeling be tuned on our PTCs?

We found that in many cases training on a PTC was not significantly worse than training on a hand-crafted test collection. This time we reported significance at both the $\alpha = 0.05$ and $\alpha = 0.001$ levels. In the former case, the null hypothesis (that performance is similar) is rejected much sooner, and we can expect some Type I errors to occur. But if we do not reject the null hypothesis, indeed the results must be quite similar. The three recipies for generating a PTC that we proposed yielded suprisingly similar results in terms of L2R test performance. With regard to the three L2R algorithms we performed our experiments with, one of them [202] was most robust with respect to which collection it was trained on. Ten of the features used in our L2R experiments were themselves retrieval algorithms which had one or more free parameters. These were tuned on the same collection on which the L2R algorithm was subsequently trained. Tuning these free parameters on our hashtag based PTCs gives these individual retrieval algorithms comparable performance to tuning them on hand-crafted test collections.

In this study, L2R results were strong, particulary on the 2011 topics, compared to results obtained in the competitive campaign in that year [166]. This perhaps gives a bit more street credibility to the answers to our main concern: the usefulness of our PTCs for training a strong L2R algorithm. It remains an interesting question why the other two L2R algorithms we tested were more sensitive to the type of material they were trained on. Since the three recipies we propose for PTC generation are similar in terms of test performance of algorithms optimized on them, one can consider other factors. Some recipies have a simple design, reducing development effort. Other recipes result in smaller PTCs, reducing computation time during training. It would be interesting to test if large PTCs generated with a simple recipe can be randomly subsampled with good results.

Overall, our results show that there is promise is the idea of generating PTCs for microblog search using hashtags. In the abscence of training data, a PTC is a reliable alternative for optimizing the ten retrieval algorithms we employed as features, and for optimizing at least one strong L2R algorithm. Thus, PTCs can be a cost-effective tool for optimizing retrieval algorithms in the microblog search domain.

In this dissertation, we studied several vertical search applications. For each, we have seen how background knowledge may be leveraged for the adaptation of algorithms and evaluation methodology. Sometimes, this can be as simple as recognizing that different types of information needs or different types of information objects should be handled differently, e.g., refraining from clustering social media profiles in people search result disambiguation. Sometimes, high quality, detailed domain knowledge is available, e.g., keyword annotations of scientific articles. In other applications there are more noisy signals, such as the use of hashtags in microblog environments. No matter the level of detail, or the level of noise in background knowledge, we have seen how such knowledge may be used to improve our understanding of information needs, the organization of search results, the evaluation and analysis of retrieval algorithms, and the generation of ground truth.

8.2 Future research directions

We walk through our domain-specific search applications once more, pointing out cross-connections and directions for future work. We sometimes abstract away somewhat from the exact task considered in each of our research chapters (Chapters 3–7), and add suggestions for future work to those already offered in the conclusion sections of those chapters. Recurring themes are the importance of error analysis, and opportunities for automatic generation of ground truth for optimization and evaluation.

In [231], and in our people search query classification study (Chapter 3), we have observed large classes of queries. It would be interesting to corroborate the proposed classification scheme with a user questionnaire, much like in [38], where people would be asked if their query fits in any of the proposed classes or if they would rather describe their information need in some other way. If the classification has merit in the eye of the users, then it makes sense to consider adapting the search engine response to the type of query, opening up many opportunities for further research. One can think of changes in the SERP, or changes in the ranking algorithm. Learning to rank algorithms performing

query specialization are already being developed, e.g., [34]. In our automatic classification experiment, we saw that regular high-profile queries were difficult. It would be interesting to drill down further and see if misclassified instances share some characteristics, e.g., perhaps they are well-known people who nonetheless did not receive much searcher attention. And, are these misclassifications harder for human annotators, too, i.e., is annotator disagreement higher?

The high ambiguity of person name queries motivated research into clustering search results, with each cluster containing document that refer to a particular individual. We have seen in our result disambiguation study (Chapter 4) that different types of search results (social media profiles vs. other documents) should be treated quite differently. This finding is most likely applicable to other domains and tasks as well. Systematic error analysis is one way to detect such opportunities: plot how documents contribute to the performance scores, contrasting documents with different characteristics. There are still many opportunities for improvements in the clustering task. One could take into account the kind of query, i.e., the predictions of the classifiers from Chapter 3. More in general, one can devise query features based on which regression models could be trained to predict the optimal stopping criterion in an unsupervised HAC algorithm. Another interesting direction is to look at supervised learning for the task of predicting whether or not two documents refer to the same individual. Training data for such algorithms can also be used in a semi-supervised clustering approach such as [26, 126]. An interesting direction for future work is to generate this kind of pairwise training data automatically. This could be done using sparse features (i.e., features that are only available for a small fraction of document pairs) with which document pairs can be clustered with high precision. Indeed, features such as face recognition, which were too sparse to improve search result clustering performance in Chapter 4, may be useful in such a setup.

In Chapter 5 we focused on evaluation of expert profiling algorithms, and we also focus on evaluation in pointing out directions for future work. One desirable extension is to be able to evaluate algorithms that perform temporal expertise profiling, e.g., [81, 191]. Automatic approaches for this and other expertise retrieval tasks are being used, e.g., keywords on scientific articles as expertise areas of the authors [81], and workshop organizers and program committe members as experts on the topic of the workshop [36]. Hashemi et al. [96] use top search results of a commercial expert search engine as a source of ground truth for expert finding. To better understand evaluation with such approaches, re-use of the proposed test-beds should be encouraged. In addition, related tasks deserve more attention, e.g., matching profiles of researchers to job openings, where subsequent job offers may be used as ground truth. Coming back to our expert profiling study, in the error analysis of which experts are difficult to profile, we see that we are much better able to generate profiles for experts if they have a natural language description of their research interests. This is another example of how different kinds of documents affect performance differently (cf. the distinction between social and non-social documents in Chapter 4). The way we performed the error analysis, considering aspects of documents one after the other and correlating them with performance metrics, is useful and should be attempted more often in information retrieval, e.g., in workshops like the Reliable Information Access worskshop [95].

In Chapters 6 and 7 we created ground truth for vertical search applications, using specific annotations available in the underlying document collections (keywords in sci-

entific articles, and hashtags in tweets). In the abscence of human labeled ground truth, such data can be used for evaluation purposes; the expertise retrieval studies discussed above are other examples. It remains an interesting question to what extent observations about system performance obtained using these automated techniques would be acknowledged by end users of search applications. In Chapter 6, we compared system rankings obtained on our PTCs to system rankings obtained on hand-crafted collections. Indirectly, this kind of research can give insights about how automatic evaluation may predict user preferences, because user studies to validate evaluation outcomes obtained using hand-crafted test collections have been performed in the literature [2, 208, 223]. It would be interesting to also perform user studies to directly validate some of the automatic evaluation approaches we discussed in Chapter 2.

In our PTC work, however, most of our attention was devoted to optimizing L2R algorithms. Here, there are several opportunities for further work. While we have shown that L2R models optimized on a PTC obtain performance on par with models trained on human labeled data, there is potential to improve L2R performance by using both hand-crafted test collections and PTCs in the training process. In fact, the Hashtags-TI collection from Chapter 7 anticipates this combination: it needs hand-crafted training data anyway, to select interesting tweets. Seen in this light, PTC generation is an attempt at semi-supervised learning (see e.g., [213] for a recent overview), combining evidence from labeled data with evidence from (possibly large) amounts of unlabeled data. In domain adaptation terminology, we can view PTCs as large reservoirs of training data from a "source domain," and hand-crafted collections as additional sparse training data from the "target domain." An example method, also known as ensemble learning, is to use the predictions of one or more PTC-trained L2R models as features for a L2R model that is subsequently trained on human labeled data. As to the choice of L2R model used in training, more complex models than the linear models we employed deserve attention, e.g., gradient boosted regression trees [85], or random forests [37]. This is because complex models in particular need large amounts of training data, as they are more prone to overfitting on small amounts of it. Future work in this direction should put some emphasis on efficiency also: training more complex models on larger amounts of training data should be demonstrated to be efficient and practical for operational search engines, and running a learned model of course needs to produce a ranking in a matter of microseconds.

In our automatic ground truth generation efforts, we have focused on using information encoded in document collections, e.g., keywords in scientific articles and hashtags in microblog posts. Another popular evaluation approach is to observe interaction of users with search engine result pages to perform evaluation, e.g., to perform A/B testing or to estimate document relevance from clickthrough rates. It would be interesting to combine content based approaches such as those proposed in this dissertation with interaction data approaches. The two approaches could complement each other, since their strengths and weaknesses do not overlap much. Interaction data reflects user preferences, which is a strength, but a lot of it is needed to achieve reliable estimates, because it is noisy. Information in the data collection reflects what content creators find important, and so it complements the user perspective. It could be given more importance in those areas where there is less interaction, e.g., for long tail queries, or for applications with a relatively small user base.

Appendices

A

Description of the TU expert collection

The TU expert collection consists of a corpus of documents, a thesaurus of expertise areas, and two sets of relevance assessments. We devote a section to each. Before we start, we briefly introduce the original UvT collection and highlight the main differences to it.

A.1 Differences with the original UvT expert collection

The original UvT expert collection was harvested from the *Webwijs* ("Webwise") system developed at Tilburg University (TU) in the Netherlands. As explained in the introduction of Chapter 5, *Webwijs* is a publicly accessible database of UvT employees who are involved in research or teaching. The UvT expert collection consists of four types of document: research descriptions, course descriptions, publications, and academic homepages. The majority of the data set was crawled in October 2006 [35].

The TU expert collection was compiled in December 2008. The main reason necessitating the update is the fact that the data contained in the original version of the collection has become outdated; employees have left the organization, others have possibly changed their areas of interest, and new documents have been generated. One additional change we implement is to exclude academic homepages from the data set, as their usefulness has been found to be limited [18, 20]. Instead, we add another document type: *student theses*; these are Bachelor and Master theses of students supervised by researchers that are connected to Tilburg University.

A.2 Documents in the TU expert collection

Table A.1 lists the types of documents available in the TU expert collection along with some descriptive statistics. It is important to note about this new collection that XML files containing research descriptions include the previously selected expertise areas in the `subject` tags; this is ground truth. We did not index the contents of these tags, so this information is not exploited in our experiments. Researchers using the TU expert collection should take care to also disregard the contents of these subject tags if they want to benchmark expert profiling systems.

Table A.1: Descriptive statistics of the TU expert collection. We list the number of documents of each type, the number of different people associated with documents of that type, and the average number of people associated with a single document.

Document type	Documents	People	People per document
Research descriptions (UK)	495	495	1.00
Research descriptions (NL)	524	524	1.00
Course descriptions	543	543	1.00
Publications	25,853	668	1.12
Student theses	5,152	520	1.17

Table A.2: Binary relations between areas x and y in the thesaurus

Abbreviation	Description	Count
BT	Area x is a broader term than area y.	1075
NT	The inverse of BT.	1076
USE	Area x is the preferred term for area y.	247
UF	The inverse of USE.	247
RT	Area x is related to area y.	1510

A.3 Expertise areas in the TU expert collection

There is a total of 2507 expertise areas. Each area is identified uniquely by a numeric identifier and has a Dutch textual label; most areas have an English translation as well. Expertise areas are organized in a thesaurus; see the next section.

A.4 A thesaurus of expertise areas

The thesaurus of expertise areas has broader-term/narrower-term relations between areas and related-term relations. In addition, it has preferred-term relations, where one area is the preferred term for another area. Of the total 2507 knowledge areas, 2266 are actually "approved."[1] When we discard areas that are not approved, there are 4155 relations in the thesaurus; Table A.2 lists the number of relations per type.

Using only the "broader term" relation we can build a directed graph, with an edge pointing from area x to area y if x is a broader term than y. Apart from a few erroneous self referencing areas this graph is acyclic. Ignoring the direction of edges for a moment, we can find the connected components in this graph. There are no fewer than 635 connected components. One is very big with 718 edges, the second biggest has only 20 edges.

[1] New expertise areas can be suggested for inclusion in the thesaurus by TU employees. These suggested areas need to be reviewed by TU university librarians and properly integrated into the thesaurus before they are fully approved.

Table A.3: Experts, total number of distinct relevant areas, and average number of areas per profile in both sets of ground truth.

	Experts	Areas	Average areas per profile
Self-selected profiles	761	1662	6.4
Judged system-generated profiles	239	1266	8.8

A.5 Two sets of relevance assessments

The TU expert collection comes with two main sets of relevance assessments, in the form of two files in standard `trec_eval` format. In both files, experts and areas are represented unique numeric identifiers. The first set of relevance assessments consists of profiles containing *self-selected areas* that experts selected from an alphabetic list of expertise areas. The second consists of profiles containing *judged system-generated areas*. Basic statistics about the number of experts and areas in both sets of ground truth are listed in Table A.3. See also Figure 5.2 for the distribution of knowledge areas.

In addition to the two main sets of relevance assessments, we also release the three intermediate sets of relevance assessments used in the analysis in Section 5.5, so that researchers can repeat our analysis with their systems.

Bibliography

[1] M. Agosti, R. Berendsen, T. Bogers, M. Braschler, P. Buitelaar, K. Choukri, G. Maria Di Nunzio, N. Ferro, P. Forner, A. Hanbury, et al. Promise retreat report prospects and opportunities for information access evaluation. In *ACM SIGIR Forum*, volume 46, pages 60–84, 2012.

[2] J. Allan, B. Carterette, and J. Lewis. When will information retrieval be good enough? In *SIGIR '05*, pages 433–440, 2005.

[3] G. Amati. Frequentist and bayesian approach to information retrieval. In *Advances in Information Retrieval*, pages 13–24. 2006.

[4] G. Amati and C. Carpineto. Fub at trec-10 web track: A probabilistic framework for topic relevance term weighting. In *TREC '10*, 2010.

[5] G. Amati and C. Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems*, 20(4):357–389, 2002.

[6] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12:461–486, 2009.

[7] E. Amigó, J. Gonzalo, J. Artiles, and M. F. Verdejo. Combining evaluation metrics via the unanimous improvement ratio and its application to clustering tasks. *Journal of Articial Intelligence Research*, 42: 689–718, 2011.

[8] E. Amitay, D. Carmel, R. Lempel, and A. Soffer. Scaling ir-system evaluation using term relevance sets. In *SIGIR '04*, pages 10–17, 2004.

[9] J. Arguello, F. Diaz, J. Callan, and B. Carterette. A methodology for evaluating aggregated search results. In *Advances in Information Retrieval*, pages 141–152. 2011.

[10] J. Artiles. *Web People Search*. PhD thesis, UNED University, 2009.

[11] J. Artiles, J. Gonzalo, and S. Sekine. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *4th International Workshop on Semantic Evaluations*, pages 64–69, 2007.

[12] J. Artiles, J. Gonzalo, and S. Sekine. Weps 2 evaluation campaign: overview of the web people search clustering task. In *2nd Web People Search Evaluation Workshop (WePS '09), at WWW '09*, 2009.

[13] J. Artiles, A. Borthwick, J. Gonzalo, S. Sekine, and E. Amigó. WePS-3 Evaluation Campaign: Overview of the Web People Search Clustering and Attribute Extraction Tasks. In *CLEF 2010 Working Notes*, 2010.

[14] N. Asadi, D. Metzler, T. Elsayed, and J. Lin. Pseudo test collections for learning web search ranking functions. In *SIGIR '11*, pages 1073–1082. ACM, 2011.

[15] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: an analysis using six european languages. In *SIGIR '07*, pages 455–462. ACM, 2007.

[16] P. Bailey, N. Craswell, I. Soboroff, and A. de Vries. The CSIRO enterprise search test collection. *ACM SIGIR Forum*, 41, 2007.

[17] P. Bailey, N. Craswell, A. P. de Vries, and I. Soboroff. Overview of the TREC 2007 Enterprise Track. In *TREC '07*, 2008.

[18] K. Balog. *People Search in the Enterprise*. PhD thesis, University of Amsterdam, June 2008.

[19] K. Balog and M. de Rijke. Determining expert profiles (with an application to expert finding). In *IJCAI'07*, pages 2657–2662, 2007.

[20] K. Balog, T. Bogers, L. Azzopardi, M. de Rijke, and A. van den Bosch. Broad expertise retrieval in sparse data environments. In *SIGIR '07*, pages 551–558, 2007.

[21] K. Balog, L. Azzopardi, and M. de Rijke. A language modeling framework for expert finding. *Information Processing and Management*, 45(1):1–19, January 2009.

[22] K. Balog, L. Azzopardi, and M. de Rijke. Resolving person names in web people search. In *Weaving Services and People on the World Wide Web*, pages 301–323. Springer Berlin Heidelberg, 2009.

[23] K. Balog, I. Soboroff, P. Thomas, N. Craswell, A. P. de Vries, and P. Bailey. Overview of the TREC 2008 Enterprise Track. In *TREC '08*, 2009.

[24] K. Balog, Y. Fang, M. de Rijke, P. Serdyukov, L. Si, et al. Expertise retrieval. *Foundations and Trends in Information Retrieval*, 6(2-3):127–256, 2012.

[25] A. Bandyopadhyay. Query expansion for microblog retrieval: 2013. In *TREC '13*, 2013.

[26] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD '04*, pages 59–68, 2004.

[27] S. M. Beitzel, E. C. Jensen, A. Chowdhury, and D. Grossman. Using titles and category names from editor-driven taxonomies for automatic evaluation. In *CIKM '03*, pages 17–23, 2003.

[28] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Using manually-built web

directories for automatic evaluation of known-item retrieval. In *SIGIR '03*, pages 373–374, 2003.
[29] R. Berendsen, M. De Rijke, K. Balog, T. Bogers, and A. Van den Bosch. On the assessment of expertise profiles. *Journal of the American Society of Information Science and Technology*, 64(10):2024–2044.
[30] R. Berendsen, B. Kovachev, E. Meij, M. De Rijke, and W. Weerkamp. Classifying queries submitted to a vertical search engine. In *WebSci '11*, pages 4:1–4:6, 2011.
[31] R. Berendsen, B. Kovachev, E.-P. Nastou, M. De Rijke, and W. Weerkamp. Result disambiguation in web people search. In *ECIR '12*, pages 146–157, 2012.
[32] R. Berendsen, E. Tsagkias, M. De Rijke, and E. Meij. Generating pseudo test collections for learning to rank scientific articles. In *CLEF '12*, pages 42–53, 2012.
[33] R. Berendsen, M. Tsagkias, W. Weerkamp, and M. De Rijke. Pseudo test collections for training and tuning microblog rankers. In *SIGIR '13*, pages 53–62, 2013.
[34] J. Bian, X. Li, F. Li, Z. Zheng, and H. Zha. Ranking specialization for web search: a divide-and-conquer approach by using topical ranksvm. In *WWW '10*, pages 131–140, 2010.
[35] T. Bogers and K. Balog. The UvT expert collection, 2006. URL: `http://ilk.uvt.nl/uvt-expert-collection/`.
[36] G. Bordea, T. Bogers, and P. Buitelaar. Benchmarking domain-specific expert search using workshop program committees. In *Workshop on computational scientometrics: theory & applications*, pages 19–24, 2013.
[37] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
[38] A. Broder. A taxonomy of web search. *ACM SIGIR Forum*, 36(2):3–10, 2002.
[39] M. Bron, E. Meij, M. Peetz, M. Tsagkias, and M. de Rijke. Team COMMIT at TREC 2011. In *TREC '11*, 2011.
[40] M. Bron, J. Van Gorp, F. Nack, M. de Rijke, A. Vishneuski, and S. de Leeuw. A subjunctive exploratory search interface to support media studies researchers. In *SIGIR '12*, pages 425–434, 2012.
[41] C. Buckley and E. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR '04*, pages 25–32, 2004.
[42] S. Büttcher, C. L. A. Clarke, P. C. K. Yeung, and I. Soboroff. Reliable information retrieval evaluation with incomplete and biased judgements. In *SIGIR '07*, pages 63–70, 2007.
[43] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Computing Surveys*, 41(3):17:1–17:38, 2009.
[44] S. Carter, W. Weerkamp, and E. Tsagkias. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation Journal*, 2012.
[45] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *SIGIR '06*, pages 268–275, 2006.
[46] B. Carterette, V. Pavlu, E. Kanoulas, J. Aslam, and J. Allan. Evaluation over thousands of queries. In *SIGIR '08*, pages 651–658, 2008.
[47] M. Cha, H. Haddadi, F. Benevenuto, and P. K. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *ICWSM '10*, volume 10, page 30, 2010.
[48] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM '09*, pages 621–630, 2009.
[49] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Boosted multi-task learning. *Machine learning*, 85(1-2):149–173, 2011.
[50] L. Chen, L. Chun, L. Ziyu, and Z. Quan. Hybrid pseudo-relevance feedback for microblog retrieval. *Journal of Information Science*, 39(6):773–788, 2013.
[51] Y. Chen, S. Lee, and C. Huang. Polyuhk: A robust information extraction system for web personal names. In *2nd Web People Search Evaluation Workshop (WePS '09), at WWW '09*, 2009.
[52] J. Choi and W. B. Croft. Temporal models for microblogs. In *CIKM '12*, pages 2491–2494, 2012.
[53] A. Chuklin, I. Markov, and M. de Rijke. *Click Models for Web Search*. Morgan & Claypool Publishers, August 2015.
[54] C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR '08*, pages 659–666, 2008.
[55] S. Clinchant and E. Gaussier. Bridging language modeling and divergence from randomness models: A log-logistic model for ir. *Advances in Information Retrieval Theory*, pages 54–65, 2009.
[56] P. Cohen. *Empirical Methods for Artificial Intelligence*, volume 55. MIT press Cambridge, MA, 1995.
[57] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
[58] D. Crabtree, X. Gao, and P. Andreae. Improving web clustering by cluster selection. In *WI'05*, pages 172–178, 2005.
[59] R. Crane and D. Sornette. Robust dynamic classes revealed by measuring the response function of a

social system. *Proceedings of the National Academy of Sciences*, 105(41):15649–15653, 2008.
[60] N. Craswell, A. de Vries, and I. Soboroff. Overview of the TREC-2005 Enterprise Track. In *TREC '05*, 2006.
[61] W. B. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*. Addison Wesley, 2010.
[62] S. Cronen-Townsend and W. Croft. Quantifying query ambiguity. In *HLT 2002*, pages 104–109, 2002.
[63] S. Cronen-Townsend and W. B. Croft. Quantifying query ambiguity. In *HLT '02*, pages 104–109, 2002.
[64] R. Cross, A. Parker, and S. Borgatti. A bird's-eye view: Using social network analysis to improve knowledge creation and sharing. *IBM Institute for Business Value*, pages 1669–1600, 2002.
[65] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL '07*, volume 7, pages 708–716, 2007.
[66] V. Dang and B. W. Croft. Query reformulation using anchor text. In *WSDM '10*, pages 41–50, 2010.
[67] S. Daskalaki, I. Kopanas, and N. Avouris. Evaluation of classifiers for an uneven class distribution problem. *Applied artificial intelligence*, 20(5):381–417, 2006.
[68] M. De Rijke, K. Balog, T. Bogers, and A. van den Bosch. On the evaluation of entity profiles. In *CLEF '10*, pages 94–99, 2010.
[69] A. D. Delgado, R. Martínez, V. Víctor Fresno, and S. Montalvo. A data driven approach for person name disambiguation in web search results. In *COLING '14*, pages 301–310, 2014.
[70] G. M. Di Nunzio. Appendix D, Results of the Domain Specific Track. In *CLEF '08 Working Notes*, 2008.
[71] F. Diaz. Performance prediction using spatial autocorrelation. In *SIGIR '07*, pages 583–590, 2007.
[72] F. Diaz and D. Metzler. Improving the estimation of relevance models using large external corpora. In *SIGIR '06*, pages 154–161, 2006.
[73] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, Mar. 1993.
[74] D. Easley and J. Kleinberg. *Networks, crowds, and markets*. Cambridge University Press, 2010.
[75] M. Efron. Hashtag retrieval in a microblogging environment. In *SIGIR '10*, pages 787–788, 2010.
[76] M. Efron. Information search and retrieval in microblogs. *Journal of the American Society of Information Science and Technology*, 62:996–1008, June 2011.
[77] M. Efron and G. Golovchinsky. Estimation methods for ranking recent information. In *SIGIR '11*, pages 495–504, 2011.
[78] A. S. El Din and W. Magdy. Web-based pseudo relevance feedback for microblog retrieval. In *TREC '12*, 2012.
[79] T. El-Ganainy, W. Magdy, and W. G. Z. Wei. Qcri at trec 2013 microblog track. In *TREC '13*, 2013.
[80] T. El-Ganainy, W. Magdy, and A. Rafea. Hyperlink-extended pseudo relevance feedback for improved microblog retrieval. In *Proceedings of the First International Workshop on Social Media Retrieval and Analysis (SoMeRa '14)*, pages 7–12, 2014.
[81] Y. Fang and A. Godavarthy. Modeling the dynamics of personal expertise. In *SIGIR '14*, pages 1107–1110, 2014.
[82] Y. Fang, L. Si, and A. P. Mathur. Discriminative models of integrating document evidence and document-candidate associations for expert search. In *SIGIR '10*, pages 683–690, 2010.
[83] C. Fautsch, L. Dolamic, S. Abdou, and J. Savoy. Domain-specific ir for german, english and russian languages. In *CLEF '07*, pages 196–199. 2008.
[84] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. *Software: Practice and Experience*, 38(2):189–225, 2008.
[85] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
[86] M. Gäde, J. Stiller, R. Berendsen, and V. Petras. Interface language, user language and success rates in the european library. In *CLEF '11 Working Notes*.
[87] F. Geraci, M. Pellegrini, P. Pisati, and F. Sebastiani. A scalable algorithm for high-quality clustering of web snippets. In *SAC'06*, pages 1058–1062, 2006.
[88] E. Graf and L. Azzopardi. A methodology for building a patent test collection for prior art search. In *2nd international workshop on evaluating information access (EVIA)*, pages 60–71, 2008.
[89] T. Gruetze, G. Kasneci, Z. Zuo, and F. Naumann. Bootstrapping wikipedia to answer ambiguous person name queries. In *ICDEW '14*, pages 56–61, 2014.
[90] M. Hagen, B. Stein, and T. Rüb. Query session detection as a cascade. In *CIKM '11*, pages 147–152, 2011.
[91] M. Hagen, J. Gomoll, A. Beyer, and B. Stein. From search session detection to search mission detection.

In *OAIR '13*, pages 85–92, 2013.
[92] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
[93] X. Han and J. Zhao. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *CIKM '09*, pages 215–224, 2009.
[94] D. Harman. Overview of the second text retrieval conference (TREC-2). *Information Processing & Management*, 31(3):271–289, 1995.
[95] D. Harman and C. Buckley. Overview of the reliable information access workshop. *Information Retrieval*, 12:615–641, 2009.
[96] S. H. Hashemi, M. Neshati, and H. Beigy. Expertise retrieval in bibliographic network: a topic dominance learning approach. In *CIKM '13*, pages 1117–1126, 2013.
[97] C. Hauff, D. Hiemstra, L. Azzopardi, and F. De Jong. A case for automatic system evaluation. *Advances in Information Retrieval*, pages 153–165, 2010.
[98] M. Hertzum and A. M. Pejtersen. The information-seeking practices of engineers: Searching for documents as well as for people. *Information Processing & Management*, 36(5):761–778, 2006.
[99] K. Hofmann. *Fast and reliable online learning to rank for information retrieval*. PhD thesis, University of Amsterdam, 2013.
[100] K. Hofmann, K. Balog, T. Bogers, and M. de Rijke. Contextual factors for finding similar experts. *Journal of the American Society for Information Science and Technology*, 61(5):994–1014, 2010.
[101] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *CIKM '11*, 2011.
[102] B. Huurnink, K. Hofmann, M. de Rijke, and M. Bron. Validating query simulators: An experiment using commercial searches and purchases. In *CLEF '10*, pages 40–51, 2010.
[103] B. Huurnink, L. Hollink, W. van den Heuvel, and M. de Rijke. Search Behavior of Media Professionals at an Audiovisual Archive: A Transaction Log Analysis. *Journal of the American Society for Information Science and Technology*, 61(6):1180–1197, June 2010.
[104] M. Ikeda, S. Ono, I. Sato, M. Yoshida, and H. Nakagawa. Person Name Disambiguation on the Web by Two-Stage Clustering. In *2nd Web People Search Evaluation Workshop (WePS '09), at WWW '09*, 2009.
[105] J. M. Jacobstein, R. M. Mersky, and D. J. Dunn. *Fundamentals of legal research*. Foundation Press, 1994.
[106] J. Janruang and W. Kreesuradej. A new web search result clustering based on true common phrase label discovery. In *International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA)*, 2006.
[107] B. J. Jansen, D. L. Booth, and A. Spink. Determining the informational, navigational, and transactional intent of web queries. *Information Processing and Management*, 44(3):1251–1266, 2008.
[108] N. Jardine and C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information storage and retrieval*, 7(5):217–240, 1971.
[109] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.
[110] M. E. Jens Kürsten, Thomas Wilhelm. The xtrieval framework at clef 2008: Domain-specific track. In *CLEF '08 Working Notes*, 2008.
[111] T. Joachims. Training linear svms in linear time. In *KDD '06*, pages 217–226, 2006.
[112] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05*, pages 154–161, 2005.
[113] R. Jones and F. Diaz. Temporal profiles of queries. *ACM Transactions on Information Systems*, 25(3): 14, 2007.
[114] S. JONES. Report on the need for and provision of an" ideal" information retrieval test collection. 1975.
[115] P. Jurczyk and E. Agichtein. Hits on question answer portals: exploration of link analysis for author ranking. In *SIGIR '07*, pages 845–846, 2007.
[116] M. Keikha, S. Gerani, and F. Crestani. Time-based relevance models. In *SIGIR '11*, pages 1087–1088, 2011.
[117] M. Kellar, C. R. Watters, and M. A. Shepherd. A field study characterizing web-based information-seeking tasks. *Journal of the American Society for Information Science and Technology*, 58(7):999–1018, 2007.
[118] R. Kelly. Twitter study reveals interesting results about usage. *PearAnalytics. August 12th*, 2009.
[119] M. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.
[120] J. Kim and W. B. Croft. Building pseudo-desktop collections. In *SIGIR '09 Workshop on the Future of*

IR Evaluation, pages 39–40, 2009.
[121] J. Kim and W. B. Croft. Retrieval experiments using pseudo-desktop collections. In *CIKM '09*, pages 1297–1306, 2009.
[122] M. Kluck. The girt data in the evaluation of clir systems–from 1997 until 2003. In *CLEF '03 Working Notes*, pages 376–390, 2004.
[123] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery*, 18(1):140–181, 2009.
[124] M. Koolen, J. Kamps, and G. Kazai. Social book search: Comparing topical relevance judgements and book suggestions for evaluation. In *CIKM '12*, pages 185–194, 2012.
[125] A. Kopliku, K. Pinel-Sauvagnat, and M. Boughanem. Aggregated search: a new information retrieval paradigm. *ACM Computing Surveys (CSUR)*, 46(3):41, 2014.
[126] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22, 2009.
[127] A. Kulkarni, J. Teevan, K. Svore, and S. Dumais. Understanding temporal query dynamics. In *WSDM '11*, 2011.
[128] N. Kumar and B. Carterette. Time based feedback and query expansion for twitter search. In *ECIR '13*, pages 734–737, 2013.
[129] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *WWW '10*, pages 591–600, 2010.
[130] V. Lavrenko and W. Croft. Relevance based language models. In *SIGIR '01*, pages 120–127, 2001.
[131] J. Lazar, J. Feng, and H. Hochheiser. *Research Methods in Human-Computer Interaction*. John Wiley & Sons Inc, 2010.
[132] G. G. Lee, J. Seo, S. Lee, H. Jung, B. hyun Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, H. Kim, and K. Kim. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *TREC '01*, pages 442–451, 2001.
[133] J. H. Lee, A. Renear, and L. C. Smith. Known-item search: Variations on a concept. In *ASIS&T '06*, volume 43, pages 1–17, 2006.
[134] Y. Li, Z. Zheng, and H. K. Dai. Kdd cup-2005 report: Facing a great challenge. *ACM SIGKDD Explorations Newsletter*, 7(2):91–99, 2005.
[135] F. Liang, R. Qiang, and J. Yang. Exploiting real-time information retrieval in the microblogosphere. In *JCDL '12*, pages 267–276, 2012.
[136] S. Liang, Z. Ren, W. Weerkamp, E. Meij, and M. de Rijke. Time-aware rank aggregation for microblog search. In *CIKM '14*, pages 989–998, 2014.
[137] J. Lin and M. Efron. Overview of the trec-2013 microblog track. In *TREC '13*, 2013.
[138] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Now Publishers Inc., 2009.
[139] C. Lundquist, D. Grossman, and O. Frieder. Improving relevance feedback in the vector space model. In *CIKM '97*, pages 16–23, 1997.
[140] M. Lupu and A. Hanbury. Patent retrieval. *Foundations and Trends in Information Retrieval*, 7(1):1–97, 2013.
[141] C. Lv, F. Fan, R. Qiang, Y. Fei, and J. Yang. Pkuicst at trec 2014 microblog track. In *TREC '14*, 2014.
[142] C. Macdonald and I. Ounis. Voting techniques for expert search. *Knowledge and information systems*, 16(3):259–280, 2008.
[143] C. Macdonald, R. Santos, and I. Ounis. The whens and hows of learning to rank for web search. *Information Retrieval*, pages 1–45, 2012.
[144] M. Madden and A. Smith. Reputation management and social media: How people monitor their identity and search for others online. Technical report, PewResearchCenter, 2010.
[145] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.
[146] K. Massoudi, E. Tsagkias, M. de Rijke, and W. Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. In *ECIR '11*, pages 362–367, 2011.
[147] R. McCreadie and C. Macdonald. Relevance in microblogs: Enhancing tweet retrieval using hyperlinked documents. In *OAIR '13*, pages 189–196, 2013.
[148] G. Mecca, S. Raunich, and A. Pappalardo. A new algorithm for clustering search results. *Data & Knowledge Engineering*, 62(3):504–522, 2007.
[149] E. Meij and M. de Rijke. The University of Amsterdam at the CLEF 2008 Domain Specific Track - parsimonious relevance and concept models. In *CLEF '08 Working Notes*, 2008.
[150] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *WSDM '12*, 2012.
[151] D. Metzler. *A Feature-Centric View of Information Retrieval*. Springer, 2011.

[152] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Information Processing & Management*, 40:735–750, September 2004.
[153] G. Mishne and M. de Rijke. A study of blog search. In *ECIR '06*, pages 289–301, 2006.
[154] T. Miyanishi, K. Seki, and K. Uehara. Trec 2012 microblog track experiments at kobe university. In *TREC '12*, 2012.
[155] T. Miyanishi, K. Seki, and K. Uehara. Combining recency and topic-dependent temporal variation for microblog search. In *ECIR '13*, volume 7814, pages 331–343, 2013.
[156] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems*, 27(1):2, 2008.
[157] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. *Journal of Machine Learning Research, Workshop and Conference Proceedings*, 14:77–89, 2011.
[158] C. Monz and W. Weerkamp. A comparison of retrieval-based hierarchical clustering approaches to person name disambiguation. In *SIGIR'09*, pages 650–651, 2009.
[159] V. Murdock and M. Lalmas. Workshop on aggregated search. In *SIGIR Forum*, volume 42, pages 80–83, 2008.
[160] R. B. Nattiya Kanhabua and K. Nrv*g. Temporal information retrieval. *Foundations and Trends in Information Retrieval*, 9(2):91–208, 2015.
[161] N. Naveed, T. Gottron, J. Kunegis, and A. C. Alhadi. Searching microblogs: coping with sparsity and document quality. In *CIKM '11*, pages 183–188, 2011.
[162] R. Nuray and F. Can. Automatic ranking of information retrieval systems using data fusion. *Information Processing & Management*, 42(3):595–614, 2006.
[163] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *SIGIR '03*, pages 143–150, 2003.
[164] M. Osborn, T. Strzalkowski, and M. Marinescu. Evaluating document retrieval in patent database: a preliminary report. In *CIKM '97*, pages 216–221, 1997.
[165] I. Ounis. Terrier: A high performance and scalable information retrieval platform. In *OSIR workshop at SIGIR '06*, 2006.
[166] I. Ounis, C. Macdonald, J. Lin, and I. Soboroff. Overview of the TREC 2011 Microblog track. In *TREC '11*, 2011.
[167] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.
[168] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *ACL'06*, pages 113–120, 2006.
[169] M.-H. Peetz and M. De Rijke. Cognitive temporal document priors. In *ECIR '13*, pages 318–330, 2013.
[170] M.-H. Peetz, J. Van Gorp, M. De Rijke, M. Bron, R. Berendsen, and W. van Dolen. Social media analysis at work: Outcomes from a multi-method observational study. *Submitted*.
[171] J. Peng, C. Macdonald, B. He, V. Plachouras, and I. Ounis. Incorporating term dependency in the dfr framework. In *SIGIR '07*, pages 843–844, 2007.
[172] V. Petras. How one word can make all the difference - using subject metadata for automatic query expansion and reformulation. In *CLEF '05 Working Notes*, 2005.
[173] V. Petras. The domain-specific track at CLEF 2008. In *CLEF '08 Working Notes*, 2008.
[174] V. Petras, S. Baerisch, and M. Stempfhuber. The domain-specific track at clef 2007. In *CLEF '07 Working Notes*, pages 160–173, 2008.
[175] A. Pilz and G. Paaß. From names to entities using thematic context distance. In *CIKM '11*, pages 857–866, 2011.
[176] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98*, pages 275–281, 1998.
[177] R. Qiang, F. Liang, and J. Yang. Exploiting ranking factorization machines for microblog retrieval. In *CIKM '13*, pages 1783–1788, 2013.
[178] J. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.
[179] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *SIGIR '10*, pages 667–674, 2010.
[180] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*, pages 43–52, 2008.
[181] S. Rajput, M. Ekstrand-Abueg, V. Pavlu, and J. Aslam. Constructing test collections by inferring document relevance via extracted relevant information. In *CIKM '12*, pages 145–154, 2012.
[182] P. Rayson and R. Garside. Comparing corpora using frequency profiling. In *Workshop on Comparing*

Corpora '00, pages 1–6, 2000.
[183] S. Robertson and K. Jones. Simple, proven approaches to text retrieval. Technical report, Computer Laboratory, University of Cambridge, 1997.
[184] S. Robertson and H. Zaragoza. On rank-based effectiveness measures and optimization. *Information Retrieval*, 10(3):321–339, 2007.
[185] S. Robertson and H. Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.
[186] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC '94*, pages 109–109, 1995.
[187] S. E. Robertson. The probability ranking principle in ir. *Journal of documentation*, 33(4):294–304, 1977.
[188] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing*. 1971.
[189] D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW '04*, pages 13–19, 2004.
[190] J. Rowley. Product search in e-shopping: a review and research propositions. *Journal of consumer marketing*, 17(1):20–35, 2000.
[191] J. Rybak, K. Balog, and K. Nørvåg. Temporal expertise profiling. In *ECIR '14*, pages 540–546, 2014.
[192] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW '10*, pages 851–860, 2010.
[193] M. Sanderson. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4:247–375, 2010.
[194] M. Sanderson and I. Soboroff. Problems with Kendall's tau. In *SIGIR '07*, pages 839–840, 2007.
[195] M. Sanderson and J. Zobel. Information retrieval system evaluation: effort, sensitivity, and reliability. In *SIGIR '05*, pages 162–169, 2005.
[196] R. L. Santos, C. Macdonald, and I. Ounis. Search result diversification. *Foundations and Trends in Information Retrieval*, 9(1):1–90, 2015.
[197] D. Sculley. Large scale learning to rank. In *NIPS '09 Workshop on Advances in Ranking*, 2009.
[198] D. Sculley. Combined regression and ranking. In *KDD '10*, pages 979–988, 2010.
[199] J. Seo and W. Croft. Thread-based expert finding. In *SIGIR '09: the Search in Social Media Workshop*, volume 9, 2009.
[200] P. Serdyukov, H. Rode, and D. Hiemstra. Modeling multi-step relevance propagation for expert finding. In *CIKM '08*, pages 1133–1142, 2008.
[201] A. Severyn, A. Moschitti, M. Tsagkias, R. Berendsen, and M. de Rijke. A syntax-aware re-ranker for microblog retrieval. In *SIGIR '14*, pages 1067–1070, 2014.
[202] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.
[203] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *Knowledge and Data Engineering*, 27(2):443–460, 2015.
[204] D. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Pr I Llc, 2011.
[205] N. R. Smalheiser and V. I. Torvik. Author name disambiguation. *Annual review of information science and technology*, 43(1):1–43, 2009.
[206] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.
[207] E. Smirnova and K. Balog. A user-oriented model for expert finding. In *ECIR '11*, pages 580–592, 2011.
[208] C. Smith and P. Kantor. User adaptation: good results from poor systems. In *SIGIR '08*, pages 147–154, 2008.
[209] M. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM '07*, pages 623–632, 2007.
[210] I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *SIGIR '01*, pages 66–73, 2001.
[211] I. Soboroff, A. de Vries, and N. Craswell. Overview of the TREC-2006 Enterprise Track. In *TREC '06*, 2007.
[212] I. Soboroff, I. Ounis, J. Lin, and I. Soboroff. Overview of the trec-2012 microblog track. In *TREC '12*, 2012.
[213] A. Søgaard. Semi-supervised learning and domain adaptation in natural language processing. *Synthesis Lectures on Human Language Technologies*, 6(2):1–103, 2013.
[214] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks.

Information Processing & Management, 45(4):427–437, 2009.
[215] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.
[216] Spärck Jones, K. Automatic indexing. *Journal of Documentation*, 30(4):393–432, 1974.
[217] A. Spink, B. Jansen, and J. Pedersen. Searching for people on web search engines. *Journal of Documentation*, 60(3):266–278, 2004.
[218] A. Spoerri. Using the structure of overlap between search results to rank retrieval systems without relevance judgments. *Information Processing & Management*, 43(4):1059–1070, 2007.
[219] M. Szummer and E. Yilmaz. Semi-supervised learning to rank with preference regularization. In *CIKM '11*, pages 269–278, 2011.
[220] J. Tague. Problems in the simulation of bibliographic retrieval systems. In *SIGIR '80*, pages 236–255, 1980.
[221] J. Teevan, D. Ramage, and M. R. Morris. # twittersearch: a comparison of microblog search and web search. In *WSDM '11*, pages 35–44, 2011.
[222] D. Tunkelang. Faceted search. *Synthesis lectures on information concepts, retrieval, and services*, 1(1): 1–80, 2009.
[223] A. Turpin and F. Scholer. User performance versus precision measures for simple search tasks. In *SIGIR '06*, pages 11–18, 2006.
[224] E. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing & Management*, 36(5):697–716, 2000.
[225] E. Voorhees and C. Buckley. The effect of topic set size on retrieval experiment error. In *SIGIR '02*, pages 316–323, 2002.
[226] E. Voorhees and D. Harman. Overview of the fifth Text REtrieval Conference (TREC-5). In *TREC '96*, 1996.
[227] E. Voorhees and D. Harman. Overview of the ninth Text REtrieval Conference (TREC-9). In *TREC '00*, 2000.
[228] Y. Wang and J. Lin. The impact of future term statistics in real-time tweet search. In *ECIR '14*, pages 567–572, 2014.
[229] W. Weerkamp and M. de Rijke. Credibility improves topical blog post retrieval. In *ACL '08: HLT*, pages 923–931, 2008.
[230] W. Weerkamp, K. Balog, and M. de Rijke. A generative blog post retrieval model that uses query expansion based on external collections. In *ACL-IJCNLP '09*, pages 1057–1065, 2009.
[231] W. Weerkamp, K. Balog, M. de Rijke, R. Berendsen, B. Kovachev, and E. Meij. People searching for people: Analysis of a people search engine log. In *SIGIR '11*, pages 45–54, 2011.
[232] B. Welch. The generalization of Students' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.
[233] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM '10*, pages 261–270, 2010.
[234] H. D. White. Scientific communication and literature retrieval. *The handbook of research synthesis and meta-analysis*, 2:51–71, 2009.
[235] T. D. Wilson. On user studies and information needs. *Journal of documentation*, 37(1):3–15, 1981.
[236] S. Wu and F. Crestani. Methods for ranking information retrieval systems without relevance judgments. In *SAC '03*, pages 811–816, 2003.
[237] T. Xu, P. McNamee, and D. W. Oard. Hltcoe at trec 2014: Microblog and clinical decision support. In *TREC '14*, 2014.
[238] X. Xue and W. B. Croft. Automatic query generation for patent search. In *CIKM '09*, pages 2037–2040, 2009.
[239] A. Yeh. More accurate tests for the statistical significance of result differences. In *COLING '00*, volume 2, pages 947–953, 2000.
[240] M. Yoshida, M. Ikeda, S. Ono, I. Sato, and H. Nakagawa. Person name disambiguation by bootstrapping. In *SIGIR '10*, pages 10–17, 2010.
[241] H. Zaragoza, N. Craswell, M. Taylor, S. Saria, and S. Robertson. Microsoft cambridge at trec-13: Web and hard tracks. In *TREC '13*.
[242] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01*, pages 334–342, 2001.
[243] D. Zhang and Y. Dong. Semantic, hierarchical, online clustering of web search results. In *APWeb '04*, pages 69–78, 2004.
[244] B. Zhu, J. Gao, X. Han, C. Shi, S. Liu, Y. Liu, and X. Cheng. ICTNET at Microblog Track TREC 2012.

In *TREC '12*, 2012.

[245] D. Zhu and H. Dreher. Improving web search by categorization, clustering, and personalization. In *ADMA '08*, pages 659–666, 2008.

[246] S. Zhu, Z. Gao, Y. Yuan, H. Wang, and G. Chen. Pris at trec 2013 microblog track. In *TREC '13*, 2013.

[247] J. Zobel. How reliable are the results of large-scale information retrieval experiments? In *SIGIR '98*, pages 307–314, 1998.

Summary

There is a growing diversity of information access applications. While general web search has been dominant in the past few decades, a wide variety of so-called *vertical search* tasks and applications have come to the fore. Vertical search is an often used term for search that targets specific content. Examples include Youtube video search, Facebook graph search, Spotify music recommendation, product search, expertise retrieval, and scientific literature search. In this dissertation, we study vertical search engines for finding people, finding papers, and finding posts.

In a vertical search application, we typically have some background knowledge about the context in which search is taking place. We may know something about the user population, about the tasks they wish to perform, about their information needs, and about the information objects in the collection we make available to them. This knowledge can inform adaptation of retrieval algorithms and evaluation methodology, to provide a better ranking of information objects, or to organize search results more effectively.

For a people search engine, we analyze and automatically detect the types of information needs searchers have. We also provide methods to deal with the high ambiguity of person names, by finding correct referents for name mentions in search results. We find that, to do this sucessfully for people search results, social media profiles need to be treated seperately from other, regular web pages. A special case of people search is expert finding. The flip-side of that task is expert profiling: ranking knowledge areas for an expert of interest. We evaluate expert profiling algorithms by asking experts to judge profiles we generated for them.

Next, we study a search engine for finding scientific papers in a collection where each paper has been labeled with keywords from a thesaurus with research areas, research methodologies, and classification codes. We show that these annotations can be used to automatically construct ground truth for the optimization and evaluation of algorithms for scientific literature search. Such methods have the potential to reduce the need for manual evaluation of the quality of search engines. We show that with a similar method, ground truth for a completely different search task can be generated: finding microblog posts. Microblog posts, e.g., tweets, contain hashtags, which can be interpreted as annotations about the topic of a tweet. The generated ground truth can be used to train machine learning algorithms for microblog search.

This dissertation showcases the need, as well as many opportunities, to leverage background knowledge in vertical search applications. It provides pointers on how this may be done to aid in understanding user information needs, organizing search results, evaluating retrieval algorithms, and generating ground truth.

Samenvatting

Er is een groeiende diversiteit in applicaties voor het ontsluiten van informatie. Internet zoekmachines hebben over de jaren een dominante positie veroverd, maar daarnaast is er een grote verscheidenheid aan zogenaamde *verticale* zoektaken, zoekmachines en applicaties op de voorgrond getreden. Met verticaal zoeken wordt zoeken naar een specifiek soort informatie bedoeld. Voorbeelden zijn Youtube video search, Facebook graph search, Spotify muziekaanbevelingen, internet winkels (zoeken naar producten), expertise zoekmachines en zoekmachines voor het zoeken naar wetenschappelijke literatuur. In dit proefschrift worden verticale zoekmachines voor zoeken naar mensen, wetenschappelijke artikelen en microblog posts onderzocht.

In een verticale zoekapplicatie is er doorgaans achtergrondkennis beschikbaar over de context waarin gezocht wordt. We weten misschien iets over de gebruikerspopulatie, over de taken die men wil volbrengen, over de soort informatie die men wil vinden en over de informatie-objecten in de collectie die door de zoekapplicatie wordt ontsloten. Deze kennis kan gebruikt worden om zoekalgoritmes en evaluatiemethodologie aan te passen, zodat een betere rangschikking of organisatie van informatie-objecten onstaat.

Voor een zoekmachine voor het zoeken naar mensen stellen we automatisch vast naar wat voor soort informatie gebruikers op zoek zijn. Ook onderzoeken we methoden om om te gaan met de grote ambiguiteit van persoonsnamen, door middel van het vinden van juiste referenten voor persoonsnamen in zoekresultaten. Om dit goed te kunnen doen blijkt dat het nodig is om profielen van sociale media platformen anders te behandelen dan andere, gewone internet pagina's. Een specialisatie van het zoeken naar mensen is het zoeken naar experts. De keerzijde van die taak is het profileren van experts: het rangschikken van expertisegebieden voor een gegeven expert. We vragen experts om profielen die we voor ze genereerden te beoordelen; met behulp van die beoordelingen kunnen we profileringsalgoritmen evalueren.

Vervolgens onderzoeken we een collectie van wetenschappelijke artikelen die zijn geannoteerd met trefwoorden uit een thesaurus dat onderzoeksgebieden, methodologieën en classificatiecodes bevat. We laten zien dat deze annotaties kunnen worden gebruikt om automatisch zoekvragen en juiste antwoorden te genereren; deze kunnen worden gebruikt voor het optimaliseren en evalueren van zoekalgoritmes voor wetenschappelijke artikelen. Dit soort methoden heeft de potentie om de noodzaak voor handmatige evaluatie van zoekmachines te verminderen. Met een soortgelijke methode kunnen we ook trainings- en evaluatiemateriaal genereren voor een volledig andere taak: het zoeken van microblog posts. Microblog posts, bijvoorbeeld tweets, bevatten hashtags. Deze hashtags kunnen worden geïnterpreteerd als annotaties met betrekking tot het onderwerp van een tweet. We laten zien dat de gegenereerde zoekvragen en antwoorden kunnen worden gebruikt om lerende zoekalgoritmes voor microblog posts te trainen.

Dit proefschrift laat de noodzaak zien van het gebruik van achtergrondkennis omtrent verticale zoekmachines, alsook de vele mogelijkheden daartoe. Het laat zien hoe achtergrondkennis kan helpen bij het begrijpen van het soort informatie waar gebruikers naar op zoek zijn, bij het organiseren van zoekresultaten, bij de evaluatie van zoekalgoritmes, en bij het genereren van trainings- en evaluatiemateriaal.

SIKS Dissertation Series

1998

1 1998-1 Johan van den Akker (CWI) *DEGAS - An Active, Temporal Database of Autonomous Objects*
2 1998-2 Floris Wiesman (UM) *Information Retrieval by Graphically Browsing Meta-Information*
3 1998-3 Ans Steuten (TUD) *A Contribution to the Linguistic Analysis of Business Conversations*
4 1998-4 Dennis Breuker (UM) *Memory versus Search in Games*
5 1998-5 E.W.Oskamp (RUL) *Computerondersteuning bij Straftoemeting*

1999

1 1999-1 Mark Sloof (VU) *Physiology of Quality Change Modelling; Automated modelling of*
2 1999-2 Rob Potharst (EUR) *Classification using decision trees and neural nets*
3 1999-3 Don Beal (UM) *The Nature of Minimax Search*
4 1999-4 Jacques Penders (UM) *The practical Art of Moving Physical Objects*
5 1999-5 Aldo de Moor (KUB) *Empowering Communities: A Method for the Legitimate User-Driven*
6 1999-6 Niek J.E. Wijngaards (VU) *Re-design of compositional systems*
7 1999-7 David Spelt (UT) *Verification support for object database design*
8 1999-8 Jacques H.J. Lenting (UM) *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism*

2000

1 2000-1 Frank Niessink (VU) *Perspectives on Improving Software Maintenance*
2 2000-2 Koen Holtman (TUE) *Prototyping of CMS Storage Management*
3 2000-3 Carolien M.T. Metselaar (UVA) *Sociaal-organisatorische gevolgen van kennistechnologie;*
4 2000-4 Geert de Haan (VU) *ETAG, A Formal Model of Competence Knowledge for User Interface*
5 2000-5 Ruud van der Pol (UM) *Knowledge-based Query Formulation in Information Retrieval*
6 2000-6 Rogier van Eijk (UU) *Programming Languages for Agent Communication*
7 2000-7 Niels Peek (UU) *Decision-theoretic Planning of Clinical Patient Management*
8 2000-8 Veerle Coup (EUR) *Sensitivity Analyis of Decision-Theoretic Networks*
9 2000-9 Florian Waas (CWI) *Principles of Probabilistic Query Optimization*
10 2000-10 Niels Nes (CWI) *Image Database Management System Design Considerations, Algorithms and Architecture*
11 2000-11 Jonas Karlsson (CWI) *Scalable Distributed Data Structures for Database Management*

2001

1 2001-1 Silja Renooij (UU) *Qualitative Approaches to Quantifying Probabilistic Networks*
2 2001-2 Koen Hindriks (UU) *Agent Programming Languages: Programming with Mental Models*
3 2001-3 Maarten van Someren (UvA) *Learning as problem solving*
4 2001-4 Evgueni Smirnov (UM) *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*
5 2001-5 Jacco van Ossenbruggen (VU) *Processing Structured Hypermedia: A Matter of Style*
6 2001-6 Martijn van Welie (VU) *Task-based User Interface Design*
7 2001-7 Bastiaan Schonhage (VU) *Diva: Architectural Perspectives on Information Visualization*
8 2001-8 Pascal van Eck (VU) *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*
9 2001-9 Pieter Jan 't Hoen (RUL) *Towards Distributed Development of Large Object-Oriented Models,*
10 2001-10 Maarten Sierhuis (UvA) *Modeling and Simulating Work Practice*
11 2001-11 Tom M. van Engers (VUA) *Knowledge Management:*

2002

1 2002-01 Nico Lassing (VU) *Architecture-Level Modifiability Analysis*
2 2002-02 Roelof van Zwol (UT) *Modelling and searching web-based document collections*
3 2002-03 Henk Ernst Blok (UT) *Database Optimization Aspects for Information Retrieval*
4 2002-04 Juan Roberto Castelo Valdueza (UU) *The Discrete Acyclic Digraph Markov Model in Data Mining*
5 2002-05 Radu Serban (VU) *The Private Cyberspace Modeling Electronic*
6 2002-06 Laurens Mommers (UL) *Applied legal epistemology; Building a knowledge-based ontology of*
7 2002-07 Peter Boncz (CWI) *Monet: A Next-Generation DBMS Kernel For Query-Intensive*
8 2002-08 Jaap Gordijn (VU) *Value Based Requirements Engineering: Exploring Innovative*

9 2002-09 Willem-Jan van den Heuvel (KUB) *Integrating Modern Business Applications with Objectified Legacy*
10 2002-10 Brian Sheppard (UM) *Towards Perfect Play of Scrabble*
11 2002-11 Wouter C.A. Wijngaards (VU) *Agent Based Modelling of Dynamics: Biological and Organisational Applications*
12 2002-12 Albrecht Schmidt (Uva) *Processing XML in Database Systems*
13 2002-13 Hongjing Wu (TUE) *A Reference Architecture for Adaptive Hypermedia Applications*
14 2002-14 Wieke de Vries (UU) *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*
15 2002-15 Rik Eshuis (UT) *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*
16 2002-16 Pieter van Langen (VU) *The Anatomy of Design: Foundations, Models and Applications*
17 2002-17 Stefan Manegold (UVA) *Understanding, Modeling, and Improving Main-Memory Database Performance*

2003

1 Heiner Stuckenschmidt (VU) *Ontology-Based Information Sharing in Weakly Structured Environments*
2 Jan Broersen (VU) *Modal Action Logics for Reasoning About Reactive Systems*
3 Martijn Schuemie (TUD) *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*
4 Milan Petkovic (UT) *Content-Based Video Retrieval Supported by Database Technology*
5 Jos Lehmann (UVA) *Causation in Artificial Intelligence and Law - A modelling approach*
6 Boris van Schooten (UT) *Development and specification of virtual environments*
7 Machiel Jansen (UvA) *Formal Explorations of Knowledge Intensive Tasks*
8 Yongping Ran (UM) *Repair Based Scheduling*
9 Rens Kortmann (UM) *The resolution of visually guided behaviour*
10 Andreas Lincke (UvT) *Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture*
11 Simon Keizer (UT) *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*
12 Roeland Ordelman (UT) *Dutch speech recognition in multimedia information retrieval*
13 Jeroen Donkers (UM) *Nosce Hostem - Searching with Opponent Models*
14 Stijn Hoppenbrouwers (KUN) *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*
15 Mathijs de Weerdt (TUD) *Plan Merging in Multi-Agent Systems*
16 Menzo Windhouwer (CWI) *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses*
17 David Jansen (UT) *Extensions of Statecharts with Probability, Time, and Stochastic Timing*
18 Levente Kocsis (UM) *Learning Search Decisions*

2004

1 Virginia Dignum (UU) *A Model for Organizational Interaction: Based on Agents, Founded in Logic*
2 Lai Xu (UvT) *Monitoring Multi-party Contracts for E-business*
3 Perry Groot (VU) *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*
4 Chris van Aart (UVA) *Organizational Principles for Multi-Agent Architectures*
5 Viara Popova (EUR) *Knowledge discovery and monotonicity*
6 Bart-Jan Hommes (TUD) *The Evaluation of Business Process Modeling Techniques*
7 Elise Boltjes (UM) *Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes*
8 Joop Verbeek (UM) *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politile gegevensuitwisseling en digitale expertise*
9 Martin Caminada (VU) *For the Sake of the Argument; explorations into argument-based reasoning*
10 Suzanne Kabel (UVA) *Knowledge-rich indexing of learning-objects*
11 Michel Klein (VU) *Change Management for Distributed Ontologies*
12 The Duy Bui (UT) *Creating emotions and facial expressions for embodied agents*
13 Wojciech Jamroga (UT) *Using Multiple Models of Reality: On Agents who Know how to Play*
14 Paul Harrenstein (UU) *Logic in Conflict. Logical Explorations in Strategic Equilibrium*
15 Arno Knobbe (UU) *Multi-Relational Data Mining*
16 Federico Divina (VU) *Hybrid Genetic Relational Search for Inductive Learning*
17 Mark Winands (UM) *Informed Search in Complex Games*
18 Vania Bessa Machado (UvA) *Supporting the Construction of Qualitative Knowledge Models*
19 Thijs Westerveld (UT) *Using generative probabilistic models for multimedia retrieval*
20 Madelon Evers (Nyenrode) *Learning from Design: facilitating multidisciplinary design teams*

2005

1 Floor Verdenius (UVA) *Methodological Aspects of Designing Induction-Based Applications*

2 Erik van der Werf (UM) *AI techniques for the game of Go*
3 Franc Grootjen (RUN) *A Pragmatic Approach to the Conceptualisation of Language*
4 Nirvana Meratnia (UT) *Towards Database Support for Moving Object data*
5 Gabriel Infante-Lopez (UVA) *Two-Level Probabilistic Grammars for Natural Language Parsing*
6 Pieter Spronck (UM) *Adaptive Game AI*
7 Flavius Frasincar (TUE) *Hypermedia Presentation Generation for Semantic Web Information Systems*
8 Richard Vdovjak (TUE) *A Model-driven Approach for Building Distributed Ontology-based Web Applications*
9 Jeen Broekstra (VU) *Storage, Querying and Inferencing for Semantic Web Languages*
10 Anders Bouwer (UVA) *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*
11 Elth Ogston (VU) *Agent Based Matchmaking and Clustering - A Decentralized Approach to Search*
12 Csaba Boer (EUR) *Distributed Simulation in Industry*
13 Fred Hamburg (UL) *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*
14 Borys Omelayenko (VU) *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics*
15 Tibor Bosse (VU) *Analysis of the Dynamics of Cognitive Processes*
16 Joris Graaumans (UU) *Usability of XML Query Languages*
17 Boris Shishkov (TUD) *Software Specification Based on Re-usable Business Components*
18 Danielle Sent (UU) *Test-selection strategies for probabilistic networks*
19 Michel van Dartel (UM) *Situated Representation*
20 Cristina Coteanu (UL) *Cyber Consumer Law, State of the Art and Perspectives*
21 Wijnand Derks (UT) *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*

2006

1 Samuil Angelov (TUE) *Foundations of B2B Electronic Contracting*
2 Cristina Chisalita (VU) *Contextual issues in the design and use of information technology in organizations*
3 Noor Christoph (UVA) *The role of metacognitive skills in learning to solve problems*
4 Marta Sabou (VU) *Building Web Service Ontologies*
5 Cees Pierik (UU) *Validation Techniques for Object-Oriented Proof Outlines*
6 Ziv Baida (VU) *Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling*
7 Marko Smiljanic (UT) *XML schema matching – balancing efficiency and effectiveness by means of clustering*
8 Eelco Herder (UT) *Forward, Back and Home Again - Analyzing User Behavior on the Web*
9 Mohamed Wahdan (UM) *Automatic Formulation of the Auditor's Opinion*
10 Ronny Siebes (VU) *Semantic Routing in Peer-to-Peer Systems*
11 Joeri van Ruth (UT) *Flattening Queries over Nested Data Types*
12 Bert Bongers (VU) *Interactivation - Towards an e-cology of people, our technological environment, and the arts*
13 Henk-Jan Lebbink (UU) *Dialogue and Decision Games for Information Exchanging Agents*
14 Johan Hoorn (VU) *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*
15 Rainer Malik (UU) *CONAN: Text Mining in the Biomedical Domain*
16 Carsten Riggelsen (UU) *Approximation Methods for Efficient Learning of Bayesian Networks*
17 Stacey Nagata (UU) *User Assistance for Multitasking with Interruptions on a Mobile Device*
18 Valentin Zhizhkun (UVA) *Graph transformation for Natural Language Processing*
19 Birna van Riemsdijk (UU) *Cognitive Agent Programming: A Semantic Approach*
20 Marina Velikova (UvT) *Monotone models for prediction in data mining*
21 Bas van Gils (RUN) *Aptness on the Web*
22 Paul de Vrieze (RUN) *Fundaments of Adaptive Personalisation*
23 Ion Juvina (UU) *Development of Cognitive Model for Navigating on the Web*
24 Laura Hollink (VU) *Semantic Annotation for Retrieval of Visual Resources*
25 Madalina Drugan (UU) *Conditional log-likelihood MDL and Evolutionary MCMC*
26 Vojkan Mihajlovic (UT) *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*
27 Stefano Bocconi (CWI) *Vox Populi: generating video documentaries from semantically annotated media repositories*
28 Borkur Sigurbjornsson (UVA) *Focused Information Access using XML Element Retrieval*

2007

1 Kees Leune (UvT) *Access Control and Service-Oriented Architectures*
2 Wouter Teepe (RUG) *Reconciling Information Exchange and Confidentiality: A Formal Approach*
3 Peter Mika (VU) *Social Networks and the Semantic Web*

4 Jurriaan van Diggelen (UU) *Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach*
5 Bart Schermer (UL) *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*
6 Gilad Mishne (UVA) *Applied Text Analytics for Blogs*
7 Natasa Jovanovic' (UT) *To Whom It May Concern - Addressee Identification in Face-to-Face Meetings*
8 Mark Hoogendoorn (VU) *Modeling of Change in Multi-Agent Organizations*
9 David Mobach (VU) *Agent-Based Mediated Service Negotiation*
10 Huib Aldewereld (UU) *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*
11 Natalia Stash (TUE) *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*
12 Marcel van Gerven (RUN) *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*
13 Rutger Rienks (UT) *Meetings in Smart Environments; Implications of Progressing Technology*
14 Niek Bergboer (UM) *Context-Based Image Analysis*
15 Joyca Lacroix (UM) *NIM: a Situated Computational Memory Model*
16 Davide Grossi (UU) *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*
17 Theodore Charitos (UU) *Reasoning with Dynamic Networks in Practice*
18 Bart Orriens (UvT) *On the development an management of adaptive business collaborations*
19 David Levy (UM) *Intimate relationships with artificial partners*
20 Slinger Jansen (UU) *Customer Configuration Updating in a Software Supply Network*
21 Karianne Vermaas (UU) *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*
22 Zlatko Zlatev (UT) *Goal-oriented design of value and process models from patterns*
23 Peter Barna (TUE) *Specification of Application Logic in Web Information Systems*
24 Georgina Ramrez Camps (CWI) *Structural Features in XML Retrieval*
25 Joost Schalken (VU) *Empirical Investigations in Software Process Improvement*

2008

1 Katalin Boer-Sorbn (EUR) *Agent-Based Simulation of Financial Markets: A modular, continuous-time approach*
2 Alexei Sharpanskykh (VU) *On Computer-Aided Methods for Modeling and Analysis of Organizations*
3 Vera Hollink (UVA) *Optimizing hierarchical menus: a usage-based approach*
4 Ander de Keijzer (UT) *Management of Uncertain Data - towards unattended integration*
5 Bela Mutschler (UT) *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*
6 Arjen Hommersom (RUN) *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*
7 Peter van Rosmalen (OU) *Supporting the tutor in the design and support of adaptive e-learning*
8 Janneke Bolt (UU) *Bayesian Networks: Aspects of Approximate Inference*
9 Christof van Nimwegen (UU) *The paradox of the guided user: assistance can be counter-effective*
10 Wauter Bosma (UT) *Discourse oriented summarization*
11 Vera Kartseva (VU) *Designing Controls for Network Organizations: A Value-Based Approach*
12 Jozsef Farkas (RUN) *A Semiotically Oriented Cognitive Model of Knowledge Representation*
13 Caterina Carraciolo (UVA) *Topic Driven Access to Scientific Handbooks*
14 Arthur van Bunningen (UT) *Context-Aware Querying; Better Answers with Less Effort*
15 Martijn van Otterlo (UT) *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains*
16 Henriette van Vugt (VU) *Embodied agents from a user's perspective*
17 Martin Op 't Land (TUD) *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*
18 Guido de Croon (UM) *Adaptive Active Vision*
19 Henning Rode (UT) *From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search*
20 Rex Arendsen (UVA) *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven*
21 Krisztian Balog (UVA) *People Search in the Enterprise*
22 Henk Koning (UU) *Communication of IT-Architecture*
23 Stefan Visscher (UU) *Bayesian network models for the management of ventilator-associated pneumonia*

24 Zharko Aleksovski (VU) *Using background knowledge in ontology matching*
25 Geert Jonker (UU) *Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency*
26 Marijn Huijbregts (UT) *Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled*
27 Hubert Vogten (OU) *Design and Implementation Strategies for IMS Learning Design*
28 Ildiko Flesch (RUN) *On the Use of Independence Relations in Bayesian Networks*
29 Dennis Reidsma (UT) *Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans*
30 Wouter van Atteveldt (VU) *Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content*
31 Loes Braun (UM) *Pro-Active Medical Information Retrieval*
32 Trung H. Bui (UT) *Toward Affective Dialogue Management using Partially Observable Markov Decision Processes*
33 Frank Terpstra (UVA) *Scientific Workflow Design; theoretical and practical issues*
34 Jeroen de Knijf (UU) *Studies in Frequent Tree Mining*
35 Ben Torben Nielsen (UvT) *Dendritic morphologies: function shapes structure*

2009

1 Rasa Jurgelenaite (RUN) *Symmetric Causal Independence Models*
2 Willem Robert van Hage (VU) *Evaluating Ontology-Alignment Techniques*
3 Hans Stol (UvT) *A Framework for Evidence-based Policy Making Using IT*
4 Josephine Nabukenya (RUN) *Improving the Quality of Organisational Policy Making using Collaboration Engineering*
5 Sietse Overbeek (RUN) *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality*
6 Muhammad Subianto (UU) *Understanding Classification*
7 Ronald Poppe (UT) *Discriminative Vision-Based Recovery and Recognition of Human Motion*
8 Volker Nannen (VU) *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*
9 Benjamin Kanagwa (RUN) *Design, Discovery and Construction of Service-oriented Systems*
10 Jan Wielemaker (UVA) *Logic programming for knowledge-intensive interactive applications*
11 Alexander Boer (UVA) *Legal Theory, Sources of Law & the Semantic Web*
12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin) *Operating Guidelines for Services*
13 Steven de Jong (UM) *Fairness in Multi-Agent Systems*
14 Maksym Korotkiy (VU) *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*
15 Rinke Hoekstra (UVA) *Ontology Representation - Design Patterns and Ontologies that Make Sense*
16 Fritz Reul (UvT) *New Architectures in Computer Chess*
17 Laurens van der Maaten (UvT) *Feature Extraction from Visual Data*
18 Fabian Groffen (CWI) *Armada, An Evolving Database System*
19 Valentin Robu (CWI) *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*
20 Bob van der Vecht (UU) *Adjustable Autonomy: Controling Influences on Decision Making*
21 Stijn Vanderlooy (UM) *Ranking and Reliable Classification*
22 Pavel Serdyukov (UT) *Search For Expertise: Going beyond direct evidence*
23 Peter Hofgesang (VU) *Modelling Web Usage in a Changing Environment*
24 Annerieke Heuvelink (VUA) *Cognitive Models for Training Simulations*
25 Alex van Ballegooij (CWI) *"RAM: Array Database Management through Relational Mapping"*
26 Fernando Koch (UU) *An Agent-Based Model for the Development of Intelligent Mobile Services*
27 Christian Glahn (OU) *Contextual Support of social Engagement and Reflection on the Web*
28 Sander Evers (UT) *Sensor Data Management with Probabilistic Models*
29 Stanislav Pokraev (UT) *Model-Driven Semantic Integration of Service-Oriented Applications*
30 Marcin Zukowski (CWI) *Balancing vectorized query execution with bandwidth-optimized storage*
31 Sofiya Katrenko (UVA) *A Closer Look at Learning Relations from Text*
32 Rik Farenhorst (VU) *Architectural Knowledge Management: Supporting Architects and Auditors*
33 Khiet Truong (UT) *How Does Real Affect Affect Affect Recognition In Speech?*
34 Inge van de Weerd (UU) *Advancing in Software Product Management: An Incremental Method Engineering Approach*
35 Wouter Koelewijn (UL) *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling*
36 Marco Kalz (OUN) *Placement Support for Learners in Learning Networks*
37 Hendrik Drachsler (OUN) *Navigation Support for Learners in Informal Learning Networks*

38 Riina Vuorikari (OU) *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*
39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin) *Service Substitution – A Behavioral Approach Based on Petri Nets*
40 Stephan Raaijmakers (UvT) *Multinomial Language Learning: Investigations into the Geometry of Language*
41 Igor Berezhnyy (UvT) *Digital Analysis of Paintings*
42 Toine Bogers (UvT) *Recommender Systems for Social Bookmarking*
43 Virginia Nunes Leal Franqueira (UT) *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients*
44 Roberto Santana Tapia (UT) *Assessing Business-IT Alignment in Networked Organizations*
45 Jilles Vreeken (UU) *Making Pattern Mining Useful*
46 Loredana Afanasiev (UvA) *Querying XML: Benchmarks and Recursion*

2010

1 Matthijs van Leeuwen (UU) *Patterns that Matter*
2 Ingo Wassink (UT) *Work flows in Life Science*
3 Joost Geurts (CWI) *A Document Engineering Model and Processing Framework for Multimedia documents*
4 Olga Kulyk (UT) *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments*
5 Claudia Hauff (UT) *Predicting the Effectiveness of Queries and Retrieval Systems*
6 Sander Bakkes (UvT) *Rapid Adaptation of Video Game AI*
7 Wim Fikkert (UT) *Gesture interaction at a Distance*
8 Krzysztof Siewicz (UL) *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments*
9 Hugo Kielman (UL) *A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging*
10 Rebecca Ong (UL) *Mobile Communication and Protection of Children*
11 Adriaan Ter Mors (TUD) *The world according to MARP: Multi-Agent Route Planning*
12 Susan van den Braak (UU) *Sensemaking software for crime analysis*
13 Gianluigi Folino (RUN) *High Performance Data Mining using Bio-inspired techniques*
14 Sander van Splunter (VU) *Automated Web Service Reconfiguration*
15 Lianne Bodenstaff (UT) *Managing Dependency Relations in Inter-Organizational Models*
16 Sicco Verwer (TUD) *Efficient Identification of Timed Automata, theory and practice*
17 Spyros Kotoulas (VU) *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications*
18 Charlotte Gerritsen (VU) *Caught in the Act: Investigating Crime by Agent-Based Simulation*
19 Henriette Cramer (UvA) *People's Responses to Autonomous and Adaptive Systems*
20 Ivo Swartjes (UT) *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative*
21 Harold van Heerde (UT) *Privacy-aware data management by means of data degradation*
22 Michiel Hildebrand (CWI) *End-user Support for Access to Heterogeneous Linked Data*
23 Bas Steunebrink (UU) *The Logical Structure of Emotions*
24 Zulfiqar Ali Memon (VU) *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective*
25 Ying Zhang (CWI) *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines*
26 Marten Voulon (UL) *Automatisch contracteren*
27 Arne Koopman (UU) *Characteristic Relational Patterns*
28 Stratos Idreos (CWI) *Database Cracking: Towards Auto-tuning Database Kernels*
29 Marieke van Erp (UvT) *Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval*
30 Victor de Boer (UVA) *Ontology Enrichment from Heterogeneous Sources on the Web*
31 Marcel Hiel (UvT) *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems*
32 Robin Aly (UT) *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval*
33 Teduh Dirgahayu (UT) *Interaction Design in Service Compositions*
34 Dolf Trieschnigg (UT) *Proof of Concept: Concept-based Biomedical Information Retrieval*
35 Jose Janssen (OU) *Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification*
36 Niels Lohmann (TUE) *Correctness of services and their composition*
37 Dirk Fahland (TUE) *From Scenarios to components*
38 Ghazanfar Farooq Siddiqui (VU) *Integrative modeling of emotions in virtual agents*
39 Mark van Assem (VU) *Converting and Integrating Vocabularies for the Semantic Web*
40 Guillaume Chaslot (UM) *Monte-Carlo Tree Search*

41 Sybren de Kinderen (VU) *Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach*
42 Peter van Kranenburg (UU) *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*
43 Pieter Bellekens (TUE) *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain*
44 Vasilios Andrikopoulos (UvT) *A theory and model for the evolution of software services*
45 Vincent Pijpers (VU) *e3alignment: Exploring Inter-Organizational Business-ICT Alignment*
46 Chen Li (UT) *Mining Process Model Variants: Challenges, Techniques, Examples*
47 Jahn-Takeshi Saito (UM) *Solving difficult game positions*
48 Bouke Huurnink (UVA) *Search in Audiovisual Broadcast Archives*
49 Alia Khairia Amin (CWI) *Understanding and supporting information seeking tasks in multiple sources*
50 Peter-Paul van Maanen (VU) *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention*
51 Edgar Meij (UVA) *Combining Concepts and Language Models for Information Access*

2011

1 Botond Cseke (RUN) *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*
2 Nick Tinnemeier (UU) *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language*
3 Jan Martijn van der Werf (TUE) *Compositional Design and Verification of Component-Based Information Systems*
4 Hado van Hasselt (UU) *Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference*
5 Base van der Raadt (VU) *Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline*
6 Yiwen Wang (TUE) *Semantically-Enhanced Recommendations in Cultural Heritage*
7 Yujia Cao (UT) *Multimodal Information Presentation for High Load Human Computer Interaction*
8 Nieske Vergunst (UU) *BDI-based Generation of Robust Task-Oriented Dialogues*
9 Tim de Jong (OU) *Contextualised Mobile Media for Learning*
10 Bart Bogaert (UvT) *Cloud Content Contention*
11 Dhaval Vyas (UT) *Designing for Awareness: An Experience-focused HCI Perspective*
12 Carmen Bratosin (TUE) *Grid Architecture for Distributed Process Mining*
13 Xiaoyu Mao (UvT) *Airport under Control. Multiagent Scheduling for Airport Ground Handling*
14 Milan Lovric (EUR) *Behavioral Finance and Agent-Based Artificial Markets*
15 Marijn Koolen (UvA) *The Meaning of Structure: the Value of Link Evidence for Information Retrieval*
16 Maarten Schadd (UM) *Selective Search in Games of Different Complexity*
17 Jiyin He (UVA) *Exploring Topic Structure: Coherence, Diversity and Relatedness*
18 Mark Ponsen (UM) *Strategic Decision-Making in complex games*
19 Ellen Rusman (OU) *The Mind ' s Eye on Personal Profiles*
20 Qing Gu (VU) *Guiding service-oriented software engineering - A view-based approach*
21 Linda Terlouw (TUD) *Modularization and Specification of Service-Oriented Systems*
22 Junte Zhang (UVA) *System Evaluation of Archival Description and Access*
23 Wouter Weerkamp (UVA) *Finding People and their Utterances in Social Media*
24 Herwin van Welbergen (UT) *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior*
25 Syed Waqar ul Qounain Jaffry (VU) *Analysis and Validation of Models for Trust Dynamics*
26 Matthijs Aart Pontier (VU) *Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots*
27 Aniel Bhulai (VU) *Dynamic website optimization through autonomous management of design patterns*
28 Rianne Kaptein (UVA) *Effective Focused Retrieval by Exploiting Query Context and Document Structure*
29 Faisal Kamiran (TUE) *Discrimination-aware Classification*
30 Egon van den Broek (UT) *Affective Signal Processing (ASP): Unraveling the mystery of emotions*
31 Ludo Waltman (EUR) *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*
32 Nees-Jan van Eck (EUR) *Methodological Advances in Bibliometric Mapping of Science*
33 Tom van der Weide (UU) *Arguing to Motivate Decisions*
34 Paolo Turrini (UU) *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations*
35 Maaike Harbers (UU) *Explaining Agent Behavior in Virtual Training*
36 Erik van der Spek (UU) *Experiments in serious game design: a cognitive approach*

37 Adriana Burlutiu (RUN) *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference*
38 Nyree Lemmens (UM) *Bee-inspired Distributed Optimization*
39 Joost Westra (UU) *Organizing Adaptation using Agents in Serious Games*
40 Viktor Clerc (VU) *Architectural Knowledge Management in Global Software Development*
41 Luan Ibraimi (UT) *Cryptographically Enforced Distributed Data Access Control*
42 Michal Sindlar (UU) *Explaining Behavior through Mental State Attribution*
43 Henk van der Schuur (UU) *Process Improvement through Software Operation Knowledge*
44 Boris Reuderink (UT) *Robust Brain-Computer Interfaces*
45 Herman Stehouwer (UvT) *Statistical Language Models for Alternative Sequence Selection*
46 Beibei Hu (TUD) *Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work*
47 Azizi Bin Ab Aziz (VU) *Exploring Computational Models for Intelligent Support of Persons with Depression*
48 Mark Ter Maat (UT) *Response Selection and Turn-taking for a Sensitive Artificial Listening Agent*
49 Andreea Niculescu (UT) *Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality*

2012

1 Terry Kakeeto (UvT) *Relationship Marketing for SMEs in Uganda*
2 Muhammad Umair (VU) *Adaptivity, emotion, and Rationality in Human and Ambient Agent Models*
3 Adam Vanya (VU) *Supporting Architecture Evolution by Mining Software Repositories*
4 Jurriaan Souer (UU) *Development of Content Management System-based Web Applications*
5 Marijn Plomp (UU) *Maturing Interorganisational Information Systems*
6 Wolfgang Reinhardt (OU) *Awareness Support for Knowledge Workers in Research Networks*
7 Rianne van Lambalgen (VU) *When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions*
8 Gerben de Vries (UVA) *Kernel Methods for Vessel Trajectories*
9 Ricardo Neisse (UT) *Trust and Privacy Management Support for Context-Aware Service Platforms*
10 David Smits (TUE) *Towards a Generic Distributed Adaptive Hypermedia Environment*
11 J.C.B. Rantham Prabhakara (TUE) *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics*
12 Kees van der Sluijs (TUE) *Model Driven Design and Data Integration in Semantic Web Information Systems*
13 Suleman Shahid (UvT) *Fun and Face: Exploring non-verbal expressions of emotion during playful interactions*
14 Evgeny Knutov (TUE) *Generic Adaptation Framework for Unifying Adaptive Web-based Systems*
15 Natalie van der Wal (VU) *Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes*
16 Fiemke Both (VU) *Helping people by understanding them - Ambient Agents supporting task execution and depression treatment*
17 Amal Elgammal (UvT) *Towards a Comprehensive Framework for Business Process Compliance*
18 Eltjo Poort (VU) *Improving Solution Architecting Practices*
19 Helen Schonenberg (TUE) *What's Next? Operational Support for Business Process Execution*
20 Ali Bahramisharif (RUN) *Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing*
21 Roberto Cornacchia (TUD) *Querying Sparse Matrices for Information Retrieval*
22 Thijs Vis (UvT) *Intelligence, politie en veiligheidsdienst: verenigbare grootheden?*
23 Christian Muehl (UT) *Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction*
24 Laurens van der Werff (UT) *Evaluation of Noisy Transcripts for Spoken Document Retrieval*
25 Silja Eckartz (UT) *Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application*
26 Emile de Maat (UVA) *Making Sense of Legal Text*
27 Hayrettin Gurkok (UT) *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games*
28 Nancy Pascall (UvT) *Engendering Technology Empowering Women*
29 Almer Tigelaar (UT) *Peer-to-Peer Information Retrieval*
30 Alina Pommeranz (TUD) *Designing Human-Centered Systems for Reflective Decision Making*
31 Emily Bagarukayo (RUN) *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure*
32 Wietske Visser (TUD) *Qualitative multi-criteria preference representation and reasoning*
33 Rory Sie (OUN) *Coalitions in Cooperation Networks (COCOON)*

34 Pavol Jancura (RUN) *Evolutionary analysis in PPI networks and applications*
35 Evert Haasdijk (VU) *Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics*
36 Denis Ssebugwawo (RUN) *Analysis and Evaluation of Collaborative Modeling Processes*
37 Agnes Nakakawa (RUN) *A Collaboration Process for Enterprise Architecture Creation*
38 Selmar Smit (VU) *Parameter Tuning and Scientific Testing in Evolutionary Algorithms*
39 Hassan Fatemi (UT) *Risk-aware design of value and coordination networks*
40 Agus Gunawan (UvT) *Information Access for SMEs in Indonesia*
41 Sebastian Kelle (OU) *Game Design Patterns for Learning*
42 Dominique Verpoorten (OU) *Reflection Amplifiers in self-regulated Learning*
43 Anna Tordai (VU) *On Combining Alignment Techniques*
44 Benedikt Kratz (UvT) *A Model and Language for Business-aware Transactions*
45 Simon Carter (UVA) *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation*
46 Manos Tsagkias (UVA) *Mining Social Media: Tracking Content and Predicting Behavior*
47 Jorn Bakker (TUE) *Handling Abrupt Changes in Evolving Time-series Data*
48 Michael Kaisers (UM) *Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions*
49 Steven van Kervel (TUD) *Ontologogy driven Enterprise Information Systems Engineering*
50 Jeroen de Jong (TUD) *Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching*

2013

1 Viorel Milea (EUR) *News Analytics for Financial Decision Support*
2 Erietta Liarou (CWI) *MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing*
3 Szymon Klarman (VU) *Reasoning with Contexts in Description Logics*
4 Chetan Yadati (TUD) *Coordinating autonomous planning and scheduling*
5 Dulce Pumareja (UT) *Groupware Requirements Evolutions Patterns*
6 Romulo Goncalves (CWI) *The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience*
7 Giel van Lankveld (UvT) *Quantifying Individual Player Differences*
8 Robbert-Jan Merk (VU) *Making enemies: cognitive modeling for opponent agents in fighter pilot simulators*
9 Fabio Gori (RUN) *Metagenomic Data Analysis: Computational Methods and Applications*
10 Jeewanie Jayasinghe Arachchige (UvT) *A Unified Modeling Framework for Service Design*
11 Evangelos Pournaras (TUD) *Multi-level Reconfigurable Self-organization in Overlay Services*
12 Marian Razavian (VU) *Knowledge-driven Migration to Services*
13 Mohammad Safiri (UT) *Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly*
14 Jafar Tanha (UVA) *Ensemble Approaches to Semi-Supervised Learning Learning*
15 Daniel Hennes (UM) *Multiagent Learning - Dynamic Games and Applications*
16 Eric Kok (UU) *Exploring the practical benefits of argumentation in multi-agent deliberation*
17 Koen Kok (VU) *The PowerMatcher: Smart Coordination for the Smart Electricity Grid*
18 Jeroen Janssens (UvT) *Outlier Selection and One-Class Classification*
19 Renze Steenhuizen (TUD) *Coordinated Multi-Agent Planning and Scheduling*
20 Katja Hofmann (UvA) *Fast and Reliable Online Learning to Rank for Information Retrieval*
21 Sander Wubben (UvT) *Text-to-text generation by monolingual machine translation*
22 Tom Claassen (RUN) *Causal Discovery and Logic*
23 Patricio de Alencar Silva (UvT) *Value Activity Monitoring*
24 Haitham Bou Ammar (UM) *Automated Transfer in Reinforcement Learning*
25 Agnieszka Anna Latoszek-Berendsen (UM) *Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System*
26 Alireza Zarghami (UT) *Architectural Support for Dynamic Homecare Service Provisioning*
27 Mohammad Huq (UT) *Inference-based Framework Managing Data Provenance*
28 Frans van der Sluis (UT) *When Complexity becomes Interesting: An Inquiry into the Information eXperience*
29 Iwan de Kok (UT) *Listening Heads*
30 Joyce Nakatumba (TUE) *Resource-Aware Business Process Management: Analysis and Support*
31 Dinh Khoa Nguyen (UvT) *Blueprint Model and Language for Engineering Cloud Applications*
32 Kamakshi Rajagopal (OUN) *Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development*
33 Qi Gao (TUD) *User Modeling and Personalization in the Microblogging Sphere*

34 Kien Tjin-Kam-Jet (UT) *Distributed Deep Web Search*
35 Abdallah El Ali (UvA) *Minimal Mobile Human Computer Interaction*
36 Promotor: Prof. dr. L. Hardman (CWI/UVA) *2013-36*
37 Than Lam Hoang (TUe) *Pattern Mining in Data Streams*
38 Dirk B̈rner (OUN) *Ambient Learning Displays*
39 Eelco den Heijer (VU) *Autonomous Evolutionary Art*
40 Joop de Jong (TUD) *A Method for Enterprise Ontology based Design of Enterprise Information Systems*
41 Pim Nijssen (UM) *Monte-Carlo Tree Search for Multi-Player Games*
42 Jochem Liem (UVA) *Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning*
43 Lon Planken (TUD) *Algorithms for Simple Temporal Reasoning*
44 Marc Bron (UVA) *Exploration and Contextualization through Interaction and Concepts*

2014

1 Nicola Barile (UU) *Studies in Learning Monotone Models from Data*
2 Fiona Tuliyano (RUN) *Combining System Dynamics with a Domain Modeling Method*
3 Sergio Raul Duarte Torres (UT) *Information Retrieval for Children: Search Behavior and Solutions*
4 Hanna Jochmann-Mannak (UT) *Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation*
5 Jurriaan van Reijsen (UU) *Knowledge Perspectives on Advancing Dynamic Capability*
6 Damian Tamburri (VU) *Supporting Networked Software Development*
7 Arya Adriansyah (TUE) *Aligning Observed and Modeled Behavior*
8 Samur Araujo (TUD) *Data Integration over Distributed and Heterogeneous Data Endpoints*
9 Philip Jackson (UvT) *Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language*
10 Ivan Salvador Razo Zapata (VU) *Service Value Networks*
11 Janneke van der Zwaan (TUD) *An Empathic Virtual Buddy for Social Support*
12 Willem van Willigen (VU) *Look Ma, No Hands: Aspects of Autonomous Vehicle Control*
13 Arlette van Wissen (VU) *Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains*
14 Yangyang Shi (TUD) *Language Models With Meta-information*
15 Natalya Mogles (VU) *Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare*
16 Krystyna Milian (VU) *Supporting trial recruitment and design by automatically interpreting eligibility criteria*
17 Kathrin Dentler (VU) *Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability*
18 Mattijs Ghijsen (VU) *Methods and Models for the Design and Study of Dynamic Agent Organizations*
19 Vincius Ramos (TUE) *Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support*
20 Mena Habib (UT) *Named Entity Extraction and Disambiguation for Informal Text: The Missing Link*
21 Kassidy Clark (TUD) *Negotiation and Monitoring in Open Environments*
22 Marieke Peeters (UU) *Personalized Educational Games - Developing agent-supported scenario-based training*
23 Eleftherios Sidirourgos (UvA/CWI) *Space Efficient Indexes for the Big Data Era*
24 Davide Ceolin (VU) *Trusting Semi-structured Web Data*
25 Martijn Lappenschaar (RUN) *New network models for the analysis of disease interaction*
26 Tim Baarslag (TUD) *What to Bid and When to Stop*
27 Rui Jorge Almeida (EUR) *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty*
28 Anna Chmielowiec (VU) *Decentralized k-Clique Matching*
29 Jaap Kabbedijk (UU) *Variability in Multi-Tenant Enterprise Software*
30 Peter de Cock (UvT) *Anticipating Criminal Behaviour*
31 Leo van Moergestel (UU) *Agent Technology in Agile Multiparallel Manufacturing and Product Support*
32 Naser Ayat (UvA) *On Entity Resolution in Probabilistic Data*
33 Tesfa Tegegne (RUN) *Service Discovery in eHealth*
34 Christina Manteli (VU) *The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems*
35 Joost van Ooijen (UU) *Cognitive Agents in Virtual Worlds: A Middleware Design Approach*
36 Joos Buijs (TUE) *Flexible Evolutionary Algorithms for Mining Structured Process Models*
37 Maral Dadvar (UT) *Experts and Machines United Against Cyberbullying*

38 Danny Plass-Oude Bos (UT) *Making brain-computer interfaces better: improving usability through post-processing*
39 Jasmina Maric (UvT) *Web Communities, Immigration, and Social Capital*
40 Walter Omona (RUN) *A Framework for Knowledge Management Using ICT in Higher Education*
41 Frederic Hogenboom (EUR) *Automated Detection of Financial Events in News Text*
42 Carsten Eijckhof (CWI/TUD) *Contextual Multidimensional Relevance Models*
43 Kevin Vlaanderen (UU) *Supporting Process Improvement using Method Increments*
44 Paulien Meesters (UvT) *Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden*
45 Birgit Schmitz (OUN) *Mobile Games for Learning: A Pattern-Based Approach*
46 Ke Tao (TUD) *Social Web Data Analytics: Relevance, Redundancy, Diversity*
47 Shangsong Liang (UVA) *Fusion and Diversification in Information Retrieval*

2015

1 Niels Netten (UvA) *Machine Learning for Relevance of Information in Crisis Response*
2 Faiza Bukhsh (UvT) *Smart auditing: Innovative Compliance Checking in Customs Controls*
3 Twan van Laarhoven (RUN) *Machine learning for network data*
4 Howard Spoelstra (OUN) *Collaborations in Open Learning Environments*
5 Christoph Bösch (UT) *Cryptographically Enforced Search Pattern Hiding*
6 Farideh Heidari (TUD) *Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes*
7 Maria-Hendrike Peetz (UvA) *Time-Aware Online Reputation Analysis*
8 Jie Jiang (TUD) *Organizational Compliance: An agent-based model for designing and evaluating organizational interactions*
9 Randy Klaassen (UT) *HCI Perspectives on Behavior Change Support Systems*
10 Henry Hermans (OUN) *OpenU: design of an integrated system to support lifelong learning*
11 Yongming Luo (TUE) *Designing algorithms for big graph datasets: A study of computing bisimulation and joins*
12 Julie M. Birkholz (VU) *Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks*
13 Giuseppe Procaccianti (VU) *Energy-Efficient Software*
14 Bart van Straalen (UT) *A cognitive approach to modeling bad news conversations*
15 Klaas Andries de Graaf (VU) *Ontology-based Software Architecture Documentation*
16 Changyun Wei (TUD) *Cognitive Coordination for Cooperative Multi-Robot Teamwork*
17 Andr van Cleeff (UT) *Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs*
18 Holger Pirk (CWI/UVA) *Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories*
19 Bernardo Tabuenco (OUN) *Ubiquitous Technology for Lifelong Learners*
20 Loïs Vanhée (UU) *Using Culture and Values to Support Flexible Coordination*
21 Sibren Fetter (OUN) *Using Peer-Support to Expand and Stabilize Online Learning*
22 Zhemin Zhu (UT) *Co-occurrence Rate Networks; An alternative theory for undirected graphical models*
23 Luit Gazendam (VU) *Cataloguer Support in Cultural Heritage*
24 Richard Berendsen (UVA) *Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation*