Bauhaus-Universität Weimar
Faculty of Media
Degree Program Computer Science and Media

Keyquery-Based Recommendation of Related Work

Master's Thesis

Anna Beyer

1. Referee: Prof. Dr. Matthias Hagen
2. Referee: Dr. Andreas Jakoby

Date of Submission: September 9th, 2014

Declaration of Authorship

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, September 9th, 2014

.....
Anna Beyer

Abstract

This thesis investigates the applicability of the concept of keyqueries proposed by Gollub et al. [GHMS13] to the recommendation of related work.
We focus on the use case in which the user has already found a small set of documents in an initial research. Thus, we address the following problem: Given a set of input documents, the task is to find a set of related research papers. This problem can be divided into two steps, i. e., the retrieval of candidate documents and the ranking of the obtained candidate documents. In the work at hand, we focus on the first crucial step and the suitability of keyqueries for it. A keyquery for a document is a query that returns the document in the top-ranked results when it is submitted to a reference search engine.
We introduce a novel method that uses the vocabulary extracted from a set of input documents to formulate keyqueries, which are then used to retrieve candidate documents. In order to rank the obtained candidate documents, a simple procedure based on the text similarity of the candidate paper to the set of input papers is applied.
For the purpose of evaluation, three state-of-the-art algorithms, which represent the ways of how humans would search for candidate documents, were implemented: (1) a citation graph-based approach by Golshan et al. [GLT12], (2) a simple query-based method by Nascimento et al. [NLSG11], and (3) an approach based on the scholarly search engine Google Scholar.
We measure the effectiveness of our novel approach and the baseline algorithms in a Cranfield-style experiment. Therefore, we utilize an index comprising about 187,000 research papers from the domain of Computer Science, and we conduct a user study involving 10 experts in order to obtain topics and relevance judgments. The results of the experiments indicate that our novel keyquery-based approach outperforms the baseline algorithms. Consequently, the concept of keyqueries appears to be valuable for the recommendation of related work.

Contents

1 Introduction

In this thesis, we study the problem of automatic recommendation of related work. Given a research task, the term "related work" refers to research papers that investigate topics similar to the research task. In the work at hand, we specifically explore the applicability of the concept of keyqueries proposed by Gollub et al. [GHMS13] for finding related work for a given set of documents.

Scientists collect and analyze related work in order to get a better understanding of their research problem and already existing approaches. A critical survey studying the strengths and weaknesses of related work forms the basis for leveraging previous findings. Nowadays, almost all research papers are accessible on the web, for example on the authors' websites or via electronic libraries, so-called digital libraries. Such collections are continuously becoming more comprehensive; for instance, the ACM Digital Library is increasing on average by 22,000 publications per year [Del11]. Despite the sophisticated search techniques on the web, it is challenging to find related work.

Conventional scholarly search engines like *Google Scholar*, *Microsoft Academic Search*, or *CiteSeerX* provide a keyword-based access to research paper collections. Since researchers usually have limited knowledge when they start a new research topic, it is difficult to formulate queries for finding all related papers. Nevertheless, scholarly search engines are often used for an initial research in order to find a few first articles. After a small set of relevant documents has been found, researchers can use information provided by these papers for finding more relevant literature.

Generally, a paper provides two types of information: content and metadata. The content comprises title, section headings and the continuous text. By using these information, new keywords for querying can be derived. All other information, i. e., bibliographic records and citation information, are metadata. This type of information can also be leveraged for finding further documents, for example by looking for more papers published by the authors or by analyzing papers from the list of references. Relevant documents found in this manner can in turn be used for finding more related papers. This iterative procedure continues until the researcher's knowledge gap is closed. It is due to the iterative structure of the search process and the expensive reading phases that finding related work is time-consuming and arduous. Support is provided by methods which automatically suggest relevant research papers in only one step.

A variety of approaches is known for the problem of finding related work. Basically, we can distinguish between two approaches. First, graph-based methods exploit the citation network spanned by a set of publications. Second, content-based methods use subjects and topics covered in a given text. One interesting content-based approach is to generate queries from a text and obtain related documents by examining the result list returned by the query. None of the existing methods for finding related work involves query-based approaches in a convincing way. Recently, Gollub et al. proposed the concept of *keyqueries* [GHMS13]. A keyquery for a document is a query that returns the document in the top result ranks when issued against a reference search engine. Accordingly, keyqueries are strongly dependent on a search engine's index and retrieval model. They describe a document not only by its own content but also by the content of all other documents contained in the index. We assume that the other results returned by a document's keyquery cover topics similar to the document itself. Thus, the concept of keyqueries seems to be useful for finding related research papers. Moreover, we believe that keyquery-based methods can find relevant documents, which graph-based methods might miss since the citation graph tends to be noisy and sparse [CSMG13]. We further assume that the combination of a keyquery-based approach and an approach that leverages citation information leads to a method which outperforms its components. Based on these assumptions, we address the following two research questions:

(1) Does an approach that is based on the concept of keyqueries outperform current state-of-the-art methods for finding related research papers?

(2) Does a method that combines keyquery-based and citation graph-based approaches outperform its components?

Our contributions are threefold: (1) We develop a keyquery-based approach for finding related work, and we propose a strategy to combine different methods. (2) For the purpose of evaluation, we implement three state-of-the-art algorithms as baselines: a graph-based approach by Golshan et al. [GLT12], a simple query-based method by Nascimento et al. [NLSG11], and an approach based on the scholarly search engine Google Scholar. (3) We conduct a user study involving ten experts in order to obtain topics and relevance judgments for a Cranfield-style experiment, and we evaluate our methods in comparison to the aforementioned baseline algorithms.

The remainder of this thesis is organized as follows: In Chapter 2, we discuss previous investigations of automatic search for related work. Additionally, we examine query formulation strategies as they are related to the concept of keyqueries. Chapter 3 describes three state-of-the-art baseline algorithms, our novel keyquery-based approaches and our strategy for combining several methods for finding related research papers. In Chapter 4, we first describe the design of our experiment. In the second part, we evaluate our approaches in comparison with the three baseline algorithms stated above. Finally, Chapter 5 summarizes the results of this thesis and suggests research questions for future work.

2 Related Work

In this chapter, we present and analyze methods that are related to our approach for searching related research papers. First, Section 2.1 examines existing methods for finding related work. Generally speaking, they can be divided into content-based and citation graph-based approaches. A subcategory of content-based methods are query-based methods. Since the existing approaches only apply simple querying techniques, we investigate more sophisticated query formulation techniques for finding related work in this thesis. Therefore, in Section 2.2 we also review query formulation strategies.

2.1 Recommendation of Research Papers

In the literature, many different denominations for the problem of finding related work exist; for example, the terms literature search, citation recommendation, research paper recommendation and academic recommendation are all synonyms for related work search. The task of related work search is to find topically similar papers according to a given research task. Basically, there are two types of research tasks. Many approaches assume that the research task is given as a set of initial research papers, whereas others try to find references for a given text. In this thesis, we will focus on the first type of research task. Thus, we define the problem of Related Work Search as follows:

Related Work Search

Given: A list of r research papers $\mathcal{I} = \langle d_{i_1}, d_{i_2}, \ldots, d_{i_r} \rangle$.
Task: Find a list of t topically similar research papers $\mathcal{O} = \langle d_{o_1}, d_{o_2}, \ldots, d_{o_t} \rangle$.

As in many previous methods, we tackle the problem of Related Work Search stepwise. Figure 2.1 illustrates the basic processing pipeline. First, researchers specify their initial knowledge in form of a list of documents, which we call *input documents* $\mathcal{I} = \langle d_{i_1}, d_{i_2}, \ldots, d_{i_r} \rangle$. Based on the input papers, *candidate documents* $\mathcal{C} = \{d_{c_1}, d_{c_2}, \ldots, d_{c_s}\}$ are collected in the retrieval step. Then, the candidate documents are ranked according to diverse features. Finally, a list of ranked documents is returned, which we call *output papers* $\mathcal{O} = \langle d_{o_1}, d_{o_2}, \ldots, d_{o_t} \rangle$.

Figure 2.1: Processing pipeline for solving the problem of Related Work Search.

In the literature, more than 80 approaches are known for the problem of finding academic articles [BLG+13]. Since we cannot review all of them, we concentrate on the approaches published in the last five years. For the classification of related work, we focus on the candidate retrieval as the first crucial step in the processing pipeline (see Figure 2.1). Starting with a set of papers, there are two different types of information which can be used for finding candidate papers, namely *content* and *metadata*. We define content to be title, abstract, section headings and the continuous text of a paper. Metadata comprises bibliographic data like authors, publication venue and publication year as well as citation information, i. e., information about references and citations of a paper.

As most previous methods use either metadata, i. e., citation information, or content for finding candidate papers, we classify them into citation graph-based methods (Section 2.1.1) and content-based (Section 2.1.2). Additionally, we describe a few mixed approaches, which use both content-based and graph-based features for candidate retrieval (Section 2.1.3).

2.1.1 Citation Graph-Based Methods

In order to find candidate papers, many citation recommendation methods exploit metadata of given documents. As stated above, we define metadata to be not only bibliographic data but also citation information of a paper. A collection of research papers spans a network of citations, which is called citation graph and often used for candidate retrieval. In the following paragraphs, we focus on the citation graph-based methods. Therefore, we introduce some basic terms: The *citation graph* $\mathcal{G} = (V, E)$ is a directed graph in which vertices refer to papers and directed links represent citations from one paper to another [AJM04]. Let d_1, d_2 be research papers. If d_1 cites d_2, we call d_1 a *citing paper* of d_2 and d_2 a *cited paper* of d_1; cited papers are specified in the reference list of a research paper. A *citation context*, which refers to text around a citation in a citing paper [RRT08], usually restates the idea of the cited paper and is used in many methods.

The following paragraphs explain three ideas of leveraging the citation graph for finding candidate papers.

Simple Approaches

A simple metadata-based approach for candidate retrieval is following the citation graph as applied by Golshan et al. in their system called *Sofia Search* [GLT12]. Starting with an initial set of papers, they build the candidate set by retrieving cited and citing papers for a given recursion depth. Then, they rank the candidate papers by computing a relevance score depending on the initial documents using features like content similarity, frequency and placement of citations or coauthorship similarity. However, Golshan et al. do not evaluate the performance of their approach.

Collaborative Filtering

Several approaches apply collaborative filtering (CF) in order to find related scholarly articles. Generally, CF is often used for recommending items like movies or products. The basic assumption of CF is that users who rate similar sets of items share similar preferences [LYZ13]. The major limitation of this recommendation approach is that it can only recommend items which have been rated before. As a result, very new items without any ratings cannot be found. This problem is a well-known issue for recommender systems; it is called the "cold-start-problem" [WB11].
Recently, Sugiyama and Kan applied CF for recommending scientific articles [SK13]. According to traditional CF, citing papers refer to users and cited papers refer to items; therefore, Sugiyama and Kan use the adjacency matrix associated with the citation graph as a rating matrix. In their experiments, Sugiyama and Kan recommend papers according to a researcher's publication list. They evaluate the results using human relevance judgments and found that their approach outperforms two state-of-the-art methods: the algorithm by Nascimento et al. [NLSG11] and the approach by Wang and Blei [WB11].
Another CF-based approach was recently proposed by Caragea et al. [CSMG13]. They apply singular value decomposition (SVD), which is a form of collaborative filtering based on matrix factorization, to the adjacency matrix of the citation graph. Caragea et al. evaluate their approach using a dataset compiled from CiteSeer and show that their SVD-based method outperforms standard CF methods. However, their evaluation setting is unrealistic: given a research paper, they only try to find one hidden reference from the reference list.

Link Ranking Algorithm-Based Approaches

Link ranking algorithms like PageRank [PBMW99], HITS [Kle99], or SALSA [LM00] have been successfully applied to rank websites based on their link structure. Since the link structure of the citation graph is similar, those algorithms are also suitable

for research paper recommendation. For example, the academic recommender service *TheAdvisor* by Küçüktunç et al. is based on PageRank [KSKC13]. Given a set of initial papers, it applies a random-walk-based PageRank variant on the citation graph associated with a large document collection. In their experiments, Küçüktunç et al., randomly hide 10% of a paper's reference list and try to find the hidden references. Note that this evaluation setting is not convincing. Since a research paper cites on average 30 documents [ZTLV13], Küçüktunç et al. on average try to find 3 hidden references for 27 given references.
Ekstrand et al. explore the application of PageRank, HITS, and SALSA for obtaining weights used in the rating matrix for collaborative filtering [EKS+10]. They found that collaborative filtering performs best in combination with PageRank.

Discussion

The major limitation of graph-based approaches is that their recommendations are restricted to the citation graph. Because of space limitations in research papers, missing citations, or errors in citing, the citation graph tends to be noisy and sparse [CSMG13]. Therefore, we assume that methods which only uses the citation graph for finding candidate documents might miss related papers. Furthermore, in graph-based candidate retrieval, often cited papers are more likely to be found and thus recommended [WB11]; as a result, they become more popular. This phenomenon is known as the Matthew effect, and it is often described with the phrase "the rich get richer and the poor get poorer" [Mer68].

Within the scope of our investigations, we want to compare our methods with one graph-based approach. Among the presented methods, the simple approach used in Sofia Search by Golshan et al. [GLT12] reproduces the way how humans would search for related work best. Therefore, we choose the approach by Golshan et al. [GLT12] as baseline; we will explain the method and our implementation of it in Section 3.1.2 in detail.

2.1.2 Content-Based Techniques

In contrast to the aforementioned methods, content-based approaches start with a piece of given text instead of a given set of documents. Generally speaking, three different content-based concepts have been applied in order to find related work.

Query-Based Methods

By submitting queries to a retrieval system, documents which cover the same topic can be obtained. Several researchers apply query-based approaches for finding related research papers.

Bethard and Jurasky propose a retrieval model for citation recommendation which recommends papers for a given query text, e. g., an abstract of a paper [BJ10]. The model uses precomputed metadata-based features on the document collection like citation count or popularity of authors, and combines them with content-based features obtained from the input text. The authors train the retrieval model on paper abstracts along with the corresponding reference lists. For the purpose of evaluation, Bethard and Jurasky utilize the ACL Anthology Reference Corpus [BDD+08]. They found that their retrieval model outperforms a baseline model which only uses text similarity.
Another query-based approach is developed by He et al. [HPK+10]. Starting with a manuscript including citation placeholders, they extract citation contexts as queries and obtain an initial set of papers by querying a reference search engine. Based on these papers, more candidate papers are retrieved by following the citation graph. He et al. explore several candidate ranking strategies and found that their non-parametric probabilistic model for ranking outperforms graph- and text-based baselines. In subsequently work, He et al. refine their method by automatic detection of citation contexts and placeholders [HKP+11]. However, the authors tackle a different problem: instead of a given manuscript including citation placeholders, we focus on a given set of documents.
Nascimento et al. propose to obtain candidate papers using a commercial search engine interface [NLSG11]. Given an input document, they first generate queries; then, they obtain candidate documents by submitting the queries individually to a search engine. Finally, they rank the candidates using text-based features. Nascimento et al. investigate different alternatives for query generation. They examine which part of a document should be used to extract queries and which kind of queries, i. e., nouns or bi-grams, are useful. In the experiments, the authors asked ten experts to first choose one input document and then to judge the relevance of the recommended papers. According to their experimental results, they suggest to use ten bi-grams extracted from title and abstract as queries. Note that queries containing only two words are very general and return a large number of results. According to the User-over-Ranking hypothesis, a users' information need is more likely to be satisfied for queries which return about as many results as the user can consider [HS11]. Therefore, the query generation method chosen by Nascimento et al. leaves some room for improvement.

Translation Models

Various approaches utilize translation models in order to find references for a given text. Translation models are based on a probability distribution which describes how likely a string in the source language is translated to a string in the target language [LYZ13]. In the context of citation recommendation, the source language could be a piece of text, and the target language could be citations.
Recently, Huang et al. propose to build a translation model using citations along with the citation contexts from a collection of research papers [HKC+12]. Given a query text,

Huang et al. compute the probability of citing a paper for each word in the text. However, translating only single words is a naive approach. A more advanced approach might be to first identify phrases in the text by using query segmentation techniques [HPBS12] and then translate these phrases into references. Huang et al. evaluate their approach on a dataset compiled from CiteSeerX and CiteULike; given a paper's text, they try to reproduce its reference list.
A similiar approach is proposed by Lu et al. [LHSY11]. While Huang et al. use references directly as target language, Lu et al. use the content of potential references as target language. Lu et al. evaluate their algorithm in comparison to two state-of-the-art methods by also taking papers' reference lists as ground truth. Note that the problem handled by Huang et al. and Lu et al. differs from our problem setting. While they try to find references for a given text, we want to find related papers for a given set of documents for further reading.
Tang et al. investigate the use of translation-based models for cross-language citation recommendation for given citation context [TWZ14]. They propose an embedding model trained on contexts and citations.

Topic Models

Topic modelling is a statistical approach which maps documents in a large collection to a low-dimensional representation in form of a set of topics [NAXC08]. Topic models have been applied for recommending references to given texts.
Nallapati et al. combine latent dirichlet allocation (LDA) with probabilistic latent semantic analysis (PLSA) in order to build a topic model from the texts and citations in a document collection [NAXC08]. This approach has been improved by Kataria et al. by using citation contexts in the document collection instead of full texts [KMB10]. Kataria et al. show that their model outperforms the model which Nallapati et al. propose.
Tang and Zhang use a two-layer Restricted Boltzmann Machine (RBM) in order to model the relationship between research papers and citation contexts [TZ09]. The model is then used to rank citations for a given text. Additionally, Tang and Zhang try to match recommended papers with sentences in the input text by calculating KL-divergence. They conduct experiments on a small dataset and found that their method performs better than a baseline algorithm using a language model and another baseline built on RBM.
Another method based on topic models is by El-Arini and Guestrin [EG11]. Given a large document collection, they extract a set of topics by selecting the most frequent non-stop words in the whole collection. Note that in the literature there are a lot of more sophisticated approaches known for topic modeling (e. g., [BNJ03]). For each topic, El-Arini and Guestrin build a graph containing papers dealing with this topic as nodes. Two nodes are connected if there is a citation or co-authorship relation; the edges are weighted by the topics frequency in the connected documents. According to

the topic's given in the input documents, El-Arini and Guestrin's method recommends papers.

Discussion

The drawback of citation recommendation approaches based on topic modeling or translation models is that they require a long training phase and they have to be retrained if new documents are added to the collection. Moreover, querying is a more natural strategy that humans would also follow in order to find related work. Therefore, we prefer a query-based method as representative of content-based citation recommendation methods. The method by Nascimento et al. [NLSG11] faces exactly the task of Related Work Search, is well documented and thus easy to implement. For these reasons, we choose this approach for our experiments and describe it in more detail in Section 3.1.1.

2.1.3 Combined Approaches

There are also methods which cannot be strictly categorized as metadata- or content-based since they use both approaches for candidate retrieval.
Wang and Blei address the problem of citation recommendation in the context of online reference management communities like *CiteULike* or *Mendeley* [WB11]. These communities enable researchers to manage papers they are interested in. Based on user ratings, Wang and Blei apply traditional collaborative filtering in which users correspond to users and papers correspond to items. In order to overcome the cold-start-problem, the authors use probabilistic topic models for articles which do not have user ratings. Wang and Blei show that the combination of collaborative filtering and probabilistic topic models outperforms its components.
In a very recent paper, Livne et al. present a system called *CiteSight*. It is supposed to support researchers by recommending references as they edit a manuscript [LGT+14]. Livne et al. use both graph- and content-based features proceeding from the current paper manuscript for retrieving candidates. For instance, they recommend papers which the author cited in the past or cited papers from references the author already added. For candidate ranking, Livne et al. use a prediction model based on decision trees including also both metadata- and content-based features.

2.2 Query Formulation

A common strategy for finding documents in a large collection is the submission of queries to a search engine. In order to find similar documents for a given document, a vocabulary is extracted and used for the generation of queries, which is known as query formulation. We can distinguish three different purposes of query formulation: (1) detection of near duplicates, (2) detection of text reuse, and (3) detection of topically similar documents [BC09]. First, in near duplicate detection the task is to find documents that have only marginally changes in the text. Second, text reuse detection aims to find documents restating the same topic with several reformulations and additions. Third, finding documents that reference the same topic is the most similar to the scenario of searching related work.

Near Duplicate Detection

Dasdan et al. investigate two approaches for finding near duplicates of a given document using only a search engine interface [DDKD09]. In their first method, they propose to generate queries by concatenating the terms from a document which are least frequent on the Web. The second query formulation approach by Dasdan et al. is characterized by random selection of consecutive terms in the input document. Their experiments prove that near duplicates can be found using their methods.

Text Reuse Detection

Bendersky and Croft suggest a query formulation technique in order to find candidate documents for text reuse detection [BC09]. Based on an input sentence, they extract unique noun phrases and named entities, called chunks. Then, each chunk is submitted as a quoted query to a search engine; the total number of results is recorded as the chunks' weight. Next, Bendersky and Croft generate queries by sorting the chunks according to their weight. Starting with all chunks as one long query, they iteratively drop the last chunk and get a shorter query. After each iteration step, the retrieved documents are added to the candidate documents which potentially contain text reuse. For in-depth text reuse analysis, the authors use sentence analysis and retrieval techniques on the obtained candidate documents.
Another query formulation strategy for candidate retrieval in text reuse detection is proposed by Hagen and Stein [HS11]. Given a set of keywords, they try to find a set of queries satisfying two constraints. First, each of the given keywords should be contained in the set of queries; they call this the *covering property*. Second, the number of documents returned by all queries together should not exceed a specified number. The latter constraint is motivated by the User-over-Ranking hypothesis; it states that queries returning about as many results as the user can consider are more likely to satisfy the

user's information need [SH11]. Since query submission to a web search engine is not for free but costly in terms of time and monetary charges, Hagen and Stein also focus on minimizing the number of submitted queries. Therefore, Hagen and Stein adapt the Apriori algorithm [AS94], in which the search space of all possible queries is traversed level-wise. In order to save queries, they propose to record the number of results of short queries for estimating the result list length of longer queries. Hagen and Stein show that their method outperforms the method by Bendersky and Croft [BC09] as well as the two approaches by Dasdan et al. [DDKD09] with respect to the quality of retrieved documents.

Search for Topical Similar Documents

Finding topically similar documents is the most difficult task compared to the aforementioned; it has been approached by several authors.
Yang et al. propose a strategy to generate queries from a document in order to find related blog posts [YBD+09]. Given an input document, they extract a number of noun phrases from the document's text. The noun phrases are ranked according to the frequencies of included words as well as the frequency of the complete phrase in the document collection. Additionally, Yang et al. obtain related terms which do not appear in the document by leveraging the hyperlink structure of Wikipedia articles. Then, they rank the terms using term frequency with respect to the underlying collection as well as mutual information. Next, they submit the top ranked terms as queries individually to a search engine and record the top search results. Yang et al. evaluate their method against human relevance judgments and found that their method indeed finds similar blog posts.
Lee and Croft first extract chunks from a short text by using conditional random fields [LC12]. They measure the importance of a chunk by various features like web frequency and suggest to generate queries by taking the two most important chunks.
Gollub et al. introduce the novel concept of keyqueries for describing a document's content [GHMS13]. A query is a keyquery for a document if it returns the document in the top-ranked results when submitted to a reference search engine. As a consequence, keyqueries strongly depend on the search engine, i. e., its indexed data collection and retrieval model. Gollub et al. propose an exhaustive search strategy for finding all keyqueries of a given document which is adapted from the approach by Hagen and Stein [HS11]. They provide pruning strategies for reducing the search space and thus, simplifying the search task. Moreover, they present two search strategies for finding a small number of diverse keyqueries. In their experimental evaluation, Gollub et al. focus on the number of submitted queries for finding one keyquery for a document.
Verberne et al. [VSK14] also tackle the problem of query term suggestion. Starting with a set of given documents, Verberne et al. extract all possible n-grams with at least one

and at most three words. Then, they use KL-divergence for ranking the extracted terms; the highest ranked terms are suggested as separate queries.

Discussion

The last task, i. e., search for topically similar documents, is highly related to the task of Related Work Search; therefore, these query formulation strategies are most suitable for finding related work. Among the presented methods, the concept of keyqueries by Gollub et al. [GHMS13] is the most promising. We assume that the other top-ranked results returned by a document's keyquery are highly related to the document. In order to investigate this assumption, we use the concept of keyqueries for candidate retrieval in our methods, which are described in Section 3.2.

3 Algorithms

This chapter describes our novel approach for recommending research papers as well as the algorithms that we use as baselines in our experiments. In Section 3.1 we first explain three baseline methods in detail. Our method, which utilizes the concept of keyqueries proposed by Gollub et al. [GHMS13], is presented in Section 3.2. Additionally, we investigate the combination of our novel approach with the baseline algorithms. In Section 3.3, we therefore describe our strategy to combine methods in order to find related work.

3.1 Baselines

We choose the baseline algorithms according to how humans search for related work. Humans usually apply three strategies in order to find related documents for a set of documents: (1) they formulate queries from given texts, (2) they follow the citation graph, and (3) they use the related articles feature of Google Scholar. For each of these three strategies we choose an algorithm that simulates the respective behavior. Thus, we utilize three baselines: a query-based technique (Section 3.1.1), a graph-based approach (Section 3.1.2), and a method that is based on Google Scholar's feature for finding related articles for a research paper (Section 3.1.3).

3.1.1 Simple Query Baseline

We choose the method proposed by Nascimento et al. [NLSG11] as query-based baseline. The following paragraphs explain the algorithm and our setting in detail.

Candidate Retrieval

In the use case addressed by Nascimento et al., only one single input document d_{i_1} is given. The candidate retrieval is based on queries and consists basically of three steps: query generation, query selection and query submission. First, Nascimento et al. use the title of the input paper d_{i_1} concatenated with its abstract in order to represent the document. In a preprocessing step, special characters, numbers, and stop words

are removed, and all terms are transformed to lowercase. In our implementation, we utilize the stop word list by Buckley et al. [BSAS95] for stop word removal, which is publicly available.[1] Then, all possible bi-grams are extracted as queries. In order to identify and remove duplicate queries, the queries are stemmed by using the Porter's algorithm [Por80].
In the second step, for each query q, or rather each bi-gram, a weight $weight(q)$ is computed. The calculation is based on term frequency $tf(t)$, which states how often the term t occurs in a document's title plus abstract. We define a term t to be a set of words, i. e., it can be a single word or a phrase. Nascimento et al. define $weight(q)$ as follows:

$$weight(q) = \gamma \cdot \frac{tf(q)}{tf\_phrase\_max} + (1 - \gamma) \cdot \frac{tf\_words(q)}{tf\_words\_max}. \tag{3.1}$$

The first summand of Equation 3.1 deals with the whole query, whereas the second summand weights the words within the query. The importance of the two summands is adjusted by the parameter γ, which we set to $\gamma = 0.66$ as proposed by the authors [NLSG11]. The variable tf_phrase_max refers to the highest frequency among all extracted bi-grams, and tf_words_max is the maximum of the following function:

$$tf\_words(q) = \frac{tf(w_1) + tf(w_2)}{tf\_word\_max}, \tag{3.2}$$

where w_1 and w_2 refers to the first and the second word respectively in the query and tf_word_max is the highest frequency of a single word in the document's title plus abstract.
In the third step, the 10 queries with the best weight are used for candidate retrieval. Each query is submitted individually to a search engine, and the top-50 search results for each query are added to the candidate set $\mathcal{C}$.

Note that the procedure for candidate retrieval is designed for only one input paper. We extend it to match our use case with a set of input papers $\mathcal{I} = \langle d_{i_1}, d_{i_2}, \ldots, d_{i_r} \rangle$ by applying it to every input paper $d_i \in \mathcal{I}$ and by adding the individually obtained candidate documents to $\mathcal{C}$.

Candidate Ranking

Nascimento et al. rank the candidate documents based on the text similarity to the input document. They suggest to represent a document's text, i. e., its title plus abstract, as a vector $\mathbf{t}$ using the vector space model with term frequency weights, in which each

[1] `www.lextek.com/manuals/onix/stopwords2.html`, Last accessed: August 31st, 2014

position describes the frequency of a term in the text. The text similarity of two texts t_1, t_2 is measured by the cosine similarity of their term frequency vectors $\mathbf{t_1}, \mathbf{t_2}$:

$$text\_similarity(t_1, t_2) = cosine\_similarity(\mathbf{t_1}, \mathbf{t_2}). \tag{3.3}$$

As usual, the cosine similarity of two vectors $\mathbf{t_1}, \mathbf{t_2}$ is defined as follows:

$$cosine\_similarity(\mathbf{t_1}, \mathbf{t_2}) = \frac{\mathbf{t_1} \cdot \mathbf{t_2}}{\|\mathbf{t_1}\|\|\mathbf{t_2}\|}. \tag{3.4}$$

In order to adopt the candidate ranking step to our use case, we define the relevance of a candidate document $d_c \in \mathcal{C}$ to the set of input documents $\mathcal{I}$ as maximum of the pairwise text similarity $d_c \in \mathcal{C}$ to each input document $d_i \in \mathcal{I}$:

$$relevance(d_c, \mathcal{I}) = \max_{d_i \in \mathcal{I}} text\_similarity(text(d_c), text(d_i)). \tag{3.5}$$

We utilize the maximum pairwise text similarity instead of the average because we want to assign a high relevance to candidate documents which are similar to at least one of the input documents. If the set of input documents have a high diversity, i. e., the input papers deal with different topics, the average function would decrease the relevance of a paper that is similar to only one input document. Since researchers might want to combine completely different approaches in order to solve their research task, it is not desirable to have a high relevance to all of the input documents.

Finally, the candidate documents are sorted by their relevance, and form the sorted set of output documents $\mathcal{O}$.

3.1.2 Sofia Search

As graph-based baseline we utilize the algorithm by Golshan et al. used in their system Sofia Search [GLT12]. We describe it in detail in the following paragraphs.

Candidate Retrieval

Starting with a list of input documents $\mathcal{I} = \langle d_{i_1}, d_{i_2}, \ldots, d_{i_r} \rangle$, cited and citing papers for each input document d_i are added to the candidate set $\mathcal{C}$. In order to enrich the candidate set $\mathcal{C}$, this routine can be repeated iteratively using already found candidate documents. For instance, consider the input document d_1, which cites the documents d_2, d_3 and is cited by the documents d_4, d_5. In the first iteration of the candidate retrieval step, cited and citing papers of the input documents are added to the candidate set $\mathcal{C}$. Thus, the set of candidate documents comprises four documents, namely $\mathcal{C} = \{d_2, d_3, d_4, d_5\}$. In the next iteration, the cited and citing papers of the candidate papers in $\mathcal{C}$ are added to the set $\mathcal{C}$, and so on.

Candidate Ranking

Golshan et al. define the relevance of a candidate document d_c with respect to the set of input documents $\mathcal{I}$ as maximum of the pairwise relevance between the candidate document $d_c \in \mathcal{C}$ and each input document $d_i \in \mathcal{I}$. The relevance of a pair of documents is defined as linear combination constructed from the following set of features:

- Content similarity of candidate and input document: It is specified as text similarity of documents; a document's text is preprocessed by removing stopwords and performing stemming using the Porter Stemmer [Por80].
- Frequency and placement of citations of the candidate paper in the input paper: It records how many times and in which section the candidate paper is explicitly cited in the input paper.
- Coauthorship similarity of candidate and input document: The set of authors of a paper and their coauthors is compared for two papers using the Jaccard coefficient; it is defined as the size of the intersection of two sets divided by the size of their union [TSK05].
- Temporal distance of candidate and input document: It is specified by the difference of publication years normalized by the difference of the current year and the year of the oldest publication in the collection.
- Citation neighborhood similarity of candidate and input document: The set of cited and citing papers of a document is compared for two papers using the Jaccard coefficient.
- Number of citations of the candidate document per year: The average citation count per year is normalized by the number of citations of the most cited paper per year in the collection.

Golshan et al. obtain the weights used in the linear combination by training their model on a data set, which contains input documents along with manually identified related documents.
Note that Golshan et al. provided us neither with information regarding the weights nor with their training data. Since we do not have a comparable data set, we cannot calculate the weights for the linear combination. As a consequence, we cannot combine the listed features used in the approach of Sofia Search in a meaningful way. Therefore, we only utilize the content similarity for ranking just like in the algorithms used in our experiments. Thus, the differences in the results produced by the approaches can be attributed to the candidate retrieval step on which we focus in this thesis. Just as with the method by Nascimento et al. [NLSG11], we define the relevance of a candidate document $d_c \in \mathcal{C}$ to the set of input documents $\mathcal{I}$ as maximum pairwise text similarity (see Equation 3.5). In contrast to the ranking method by Nascimento et al., we use

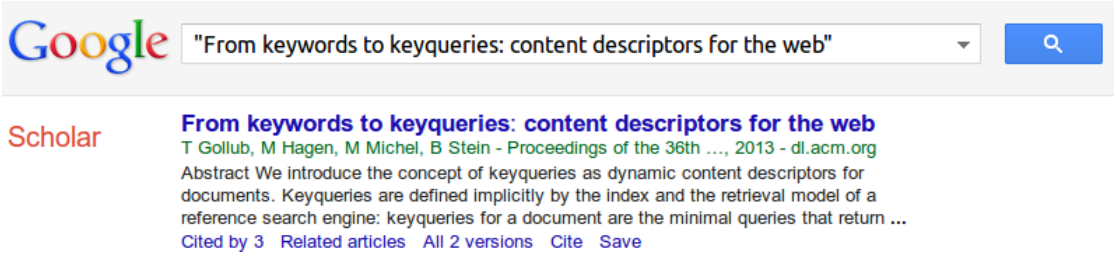

Figure 3.1: Example of a search result listed on the web page of Google Scholar.

the entire text of a document if it is available, otherwise we also concatenate title and abstract for text similarity computation.

For the candidate retrieval step of our implementation of Sofia Search, we choose to trace the citation graph with a recursion depth of 2 because a recursion depth of 1 would result in too few documents, and a depth of 3 would result in too many candidate documents.

3.1.3 Google Scholar

In a third baseline, we utilize the commercial search engine Google Scholar, i. e., its feature to find related articles. Note that Google Scholar is only capable of processing one input document at a time. In order to obtain related research papers for a set of input documents $\mathcal{I}$, we apply the strategy described in the following paragraphs.

Candidate Retrieval

For each input document $d_i \in \mathcal{I}$ we first issue its title to Google Scholar as quoted phrase and obtain search resulsts as illustrated in Figure 3.1. At the bottom of the search result entry a hyperlink to "Related articles", which refers to the website listing related articles, is included. We follow this hyperlink and record the titles of the top-100 related articles for the input document. Subsequently, we use the listed titles in order to check whether the corresponding documents are included in our data collection, and we add the found documents to the candidate set $\mathcal{C}$.

Candidate Ranking

Since we know neither Google Scholar's underlying retrieval model nor any implementation details, we can only use it as black box. However, we are convinced that the related articles returned by the search engine are ranked by relevance. Therefore, we do not change the ranking produced by Google Scholar but keep it. In order to obtain one single result list containing related articles for each input document, we apply an

Table 3.1: Example search result lists for 3 input documents.

Rank	d_{i_1}	d_{i_2}	d_{i_3}
1	d_{o_1}	d_{o_4}	d_{o_6}
2	d_{o_2}	d_{o_2}	d_{o_1}
3	d_{o_3}	d_{o_5}	d_{o_7}

alternating merging procedure. Note that there could be duplicate documents in the merged list since documents may be related to more than one input paper. We keep the first occurrence of each document and remove duplicates appearing later in the ranking. The following example illustrates our merging strategy. Given a set of input documents $\mathcal{I} = \langle d_{i_1}, d_{i_2}, d_{i_3} \rangle$, the documents d_o obtained from Google Scholar are listed in Table 3.1. For reasons of clarity, we only consider the top-3 related articles returned by each input document in this example. In order to obtain one final result list, we alternately merge the document lists in a rank-wise manner. We assume that the user specifies the input documents sorted by their importance. Therefore, we first add the top related article from d_{i_1}, then from d_{i_2} and finally from d_{i_3} to the set of output documents $\mathcal{O}$. Then, we continue with the second rank, and finally we process the third rank. As a result, the list of output documents contains the following documents $\mathcal{O} = \langle d_{o_1}, d_{o_4}, d_{o_6}, d_{o_2}, d_{o_2}, d_{o_1}, d_{o_3}, d_{o_5}, d_{o_7} \rangle$. In a pruning step, we remove duplicate documents from the list and keep the first occurrence. This leads to the sorted list of output documents $\mathcal{O} = \langle d_{o_1}, d_{o_4}, d_{o_6}, d_{o_2}, d_{o_3}, d_{o_5}, d_{o_7} \rangle$.

In the following section, we describe our novel keyquery-based approach for finding related research papers.

3.2 Keyquery-Based Approach

We introduce a novel approach for research paper recommendation, which is based on the concept of keyqueries by Gollub et al. [GHMS13]. A query q is a *keyquery* for a document d with respect to a reference search engine S if it fulfills the following two conditions: (1) d is among the top-k results returned by S on q, and (2) no subset $q' \subset q$ returns d in its top-k results when submitted to S [GHMS13]. The generality of a keyquery is defined by the parameter k, which is typically set to a small number like $k = 10$.

We assume that the search results returned by a document's keyqueries in the top ranks cover similar topics as the document itself. Therefore, we expect the keyqueries of a document to be useful for finding related research papers.

Given a document's vocabulary W, a set $\mathcal{Q}$ of keyqueries can be formulated. Note that the terms $w \in W$ can not only be single words but also complete phrases. The more terms w are included in $\mathcal{Q}$, the better $\mathcal{Q}$ describes the document's topics represented by the vocabulary W. Therefore, we try to find a set $\mathcal{Q}$ of keyqueries that contains as many terms w as possible.

The more general problem of finding a set of queries instead of a set of keyqueries has been addressed by Stein and Hagen [SH10]. They denote this problem as Query Cover. According to Stein and Hagen the task of Query Cover is to find a set $\mathcal{Q} = \{q_1, \ldots, q_m\}$ of queries $q_i \subseteq W$ with the following two characteristics: (1) In order to avoid redundancy, $\mathcal{Q}$ should be *simple*, which means that $q_i \not\subseteq q_j$ for any $q_i, q_j \in \mathcal{Q}$ with $i \neq j$. (2) A query $q \in \mathcal{Q}$ has to be *valid*, that is the length l_q of the result list returned by the search engine S on the query $q \in \mathcal{Q}$ has to be smaller than an upper bound l_{max} and greater than a lower bound l_{min}. Moreover, Stein and Hagen say a query q *covers* all the keywords contained in q, and a set of queries $\mathcal{Q}$ covers all keywords which are covered by all included queries $\bigcup_{q \in \mathcal{Q}} q$. They state that it is not always possible to cover W with a set of valid queries $\mathcal{Q}$, for example, when a single keyword itself is underflowing. Stein and Hagen define a keyword $w \in W$ to be *coverable* if and only if there is a valid query $q \subseteq W$ with $w \in q$.

We adapt the problem of Query Cover to keyqueries in order to utilize it for finding related research papers for a given set of documents. Therefore, three adjustments have to be made. First, each query $q \in \mathcal{Q}$ has to be a keyquery. Second, we define a keyquery for a set of documents: A query q is a keyquery for a set D of documents with respect to a reference search engine S iff: (1) each document $d \in D$ is among the top-k results returned by S on q, and (2) no subset $q' \subset q$ returns all documents $d \in D$ in its top-k results when submitted to S. Last, because of the additional restrictions made by the keyquery definition, we will not use an upper bound l_{max} on the result list length as defined in the problem of Query Cover.

We formally define the problem of computing a cover of keyqueries for a set of documents as follows:

Keyquery Cover

Given: (1) A set W of keywords.
(2) A set D of documents.
(3) Level k of keyquery generality.
(4) Lower bound l on the result list length.

Task: Find a simple family $\mathcal{Q} \subseteq 2^W$ of keyqueries for all $d \in D$ with respect to k and l covering the coverable keywords from W.

Typically, the parameters k and l are set to a small number like $k = 10$ and $l = 10$. That is, a keyquery for a document has to return at least 10 results and list the document in the top-10 results. Note that the vocabulary W could be extracted from the documents $d \in D$. Yet, we choose to use the vocabulary W as separate input parameter because we need an extra input parameter for vocabulary later.

We solve the problem of Keyquery Cover and use the obtained keyqueries in our novel approach in order to find candidate documents, which we then rate with a basic ranking procedure. The following paragraphs describe our keyquery-based approach in detail.

Candidate Retrieval

We first introduce our method for solving Keyquery Cover for a given vocabulary and a set of documents. Then, we explain how we obtain candidate documents.

Solving Keyquery Cover

Our approach for solving Keyquery Cover is listed as pseudocode in Algorithm 1. Basically, the algorithm consists of four steps. First, we remove all keywords $w \in W$ from the vocabulary W that return less than l results when submitted to the reference search engine S (lines 1-2). Queries q containing such keywords $w \subseteq q$ produce less than l results as well. Consequently, they cannot be a keyquery.

In the second step, the remaining terms $w \in W$ are traversed and candidate queries q_{cand} are generated by adding terms $w \in W$ (lines 3-4). If a candidate query q_{cand} is a keyquery for all documents $d \in D$ and the reference search engine S, it is added to the set of keyqueries $\mathcal{Q}$ and the candidate keyquery q_{cand} is emptied (line 5). Then the next candidate query q_{cand} is processed, and so on. This loop continues until the candidate query containing the last term $w \in W$ was processed.

Note that after this first iteration not all terms $w \in W$ necessarily are covered by the set of keyqueries $\mathcal{Q}$. If there are terms which are not yet included in a keyquery in $\mathcal{Q}$, i. e., if q_{cand} is not empty (line 6), we again go through the terms $w \in W$ (line 7). We consecutively add terms w_i to q_{cand} as long as a keyquery is found (lines 9-10). Note that according to the definition of keyqueries there is no subset $q' \subset q$ which is also a keyquery. Thus, keyqueries which are already in $\mathcal{Q}$ cannot be contained in the candidate query q_{cand}. Therefore, we omit terms $w \in W$ which would cause q_{cand} to contain a whole keyquery $q_{cand} \subseteq q$ (line 8).

After this iteration, we finish searching keyqueries although not all terms $w \in W$ may be covered by the set $\mathcal{Q}$ of keyqueries. However, any further iteration would only add keywords which are already included in q_{cand}.

The following example illustrates the process of the formulation of keyqueries. For reasons of clarity, we choose only one input paper in this example and a small vocabulary. Consider a paper d and the vocabulary $W = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$. Because

Algorithm 1 Solving Keyquery Cover

Input: Set W of keywords
Set D of documents
Level k of keyquery generality
Lower bound l on the result list length
Output: Set $\mathcal{Q}$ of keyqueries

1: **for all** $w \in W$ **do**
2: **if** w returns less than l search results **then** $W \leftarrow W \setminus w$
3: $q_{cand} \leftarrow \emptyset$
4: **for all** $w \in W$ **do**
5: $q_{cand} \leftarrow q_{cand} \cup w$
6: **if** q_{cand} is keyquery for all $d \in D$ **then** $\mathcal{Q} \leftarrow \mathcal{Q} \cup q_{cand}, q_{cand} \leftarrow \emptyset$
7: **if** $q_{cand} \neq \emptyset$ **then**
8: **for all** $w \in W$ **do**
9: **if** $\nexists\, q \in \mathcal{Q} : q \subset \{q_{cand} \cup w\}$ **then**
10: $q_{cand} \leftarrow q_{cand} \cup w$
11: **if** q_{cand} is keyquery **then** $\mathcal{Q} \leftarrow \mathcal{Q} \cup q_{cand}$, **break**
12: **return** $\mathcal{Q}$

the terms w_2 and w_5 return less than l results, they are removed from W. We iterate over $W = \langle w_1, w_3, w_4, w_6, w_7, w_8 \rangle$ and find that "$w_1 w_3 w_4$" and "$w_6 w_7$" are keyqueries for the document d and add them to the set $\mathcal{Q}$ of keyqueries. Since the term w_8 is not yet covered by $\mathcal{Q}$, we again build a candidate query q_{cand} by adding terms $w \in W$ from the beginning. We find that $w_8 w_1$ is a keyquery for d; therefore, we stop the iteration and return the set of keyqueries $\mathcal{Q} = \langle$"$w_1 w_3 w_4$", "$w_6 w_7$", "$w_8 w_1$"$\rangle$.
Finally, Algorithm 1 returns the set $\mathcal{Q}$ of keyqueries, which is then used to get the set $\mathcal{C}$ of candidate documents by taking the top results returned by each keyquery $q \in \mathcal{Q}$.
In the next paragraphs, it is explained how we use Algorithm 1 in order to find candidate documents for a set of input documents.

Computing the Set of Candidate Documents
Our algorithm for computing a set $\mathcal{C}$ of candidate documents for a given set $\mathcal{I}$ of input documents is listed in Algorithm 2. Generally speaking, the algorithm first tries to solve Keyquery Cover for all given input documents $d_i \in \mathcal{I}$ and adds the corresponding results to the set $\mathcal{C}$ of candidate documents. If there are not enough candidate documents after this step, Algorithm 2 tackles Keyquery Cover for all combinations with $|\mathcal{I}| - 1$ input documents, next with $|\mathcal{I}| - 2$ input documents, and so on.
First, the total number n of desired candidate documents is computed by multiplying the number $|\mathcal{I}|$ of input documents with the given number n_i of candidate documents per input document (line 1).

Algorithm 2 Computing the Set of Candidate Documents

Input: Set $\mathcal{I}$ of input documents
Number n_i of candidate documents per input document
Level k of keyquery generality
Lower bound l on the result list length
Output: Set $\mathcal{C}$ of candidate documents

1: $n \leftarrow |\mathcal{I}| \cdot n_i$
2: **for** $level = |\mathcal{I}|$ **down to** $level = 1$ **do**
3: **for all** $D : D \in \mathcal{P}(\mathcal{I}), |D| = level$ **do**
4: $W_{comb} \leftarrow$ combine vocabularies of documents in D
5: $\mathcal{Q}_{KQC} \leftarrow$ compute KEYQUERY COVER with W_{comb}, D, k, l
6: $\mathcal{Q}_{level} \leftarrow \mathcal{Q}_{level} \cup \{\mathcal{Q}_{KQC} \setminus \mathcal{Q}\}$
7: $n_{level} \leftarrow (n - |\mathcal{C}|) : |\mathcal{Q}_{level}|$
8: **for all** $q : q \in \mathcal{Q}_{level}$ **do**
9: $\mathcal{Q} \leftarrow \mathcal{Q} \cup q$
10: $\mathcal{C} \leftarrow \mathcal{C} \cup$ get top-n_{level} results of q
11: $\mathcal{Q}_{level} \leftarrow \emptyset$
12: **if** $|\mathcal{C}| = n$ **then break**
13: **return** $\mathcal{C}$

Then, we process all combinations of the given input set $\mathcal{I}$ in a level-wise manner by the number of elements (line 2). For this purpose, we utilize the power set $\mathcal{P}(\mathcal{I})$ of the set $\mathcal{I}$ of input documents (line 3), which comprises all subsets of $\mathcal{I}$, the empty set and $\mathcal{I}$ itself. In the first level, we tackle the combination with $level = |\mathcal{I}|$ elements. Note that there is only one such combination because no distinction is made with respect to the order of the elements. Then, we process all combinations with $|\mathcal{I}| - 1$ elements, and so on.
In each level, we select all sets D of documents from the power set $\mathcal{P}(\mathcal{I})$ that contain exactly the number of $level$ documents (line 3). For each set D, we combine the vocabularies W_{comb} of the contained documents $d \in D$ (line 4). Therefore, we first rank the terms according to their document frequency with respect to D, and second according to their mean rank in the vocabulary list extracted from each document $d \in D$.

The following example illustrates our strategy for the combination of vocabularies. Given a list of input documents $\mathcal{I} = \langle d_{i_1}, d_{i_2}, d_{i_3} \rangle$, the vocabulary for each document is listed in the left part of Table 3.2. For each term w_i in the given vocabularies, we compute its frequency, i. e., in how many of the given lists it occurs, and its mean rank as listed in the right part of Table 3.2. The combined vocabulary W_{comb} is obtained by sorting the terms w_i first by descending frequency and then by ascending mean rank. In the example, this leads to the list $W_{comb} = \langle w_2, w_3, w_1, w_4, w_5, w_6 \rangle$. We then solve KEYQUERY COVER for the combined vocabulary W_{comb} and the current set of documents D as described

Table 3.2: Example for combining lists of vocabulary. The left table shows vocabulary extracted from a set $\mathcal{I}$ of three input documents $\mathcal{I} = \langle d_{i_1}, d_{i_2}, d_{i_3} \rangle$. The variables $w_1, \ldots, w_6$ represent the extracted keywords. The right table lists statistics on the keywords with respect to all vocabulary lists. Note that the terms are sorted by descending frequency and ascending mean rank.

Rank	d_{i_1}	d_{i_2}	d_{i_3}
1	w_1	w_3	w_2
2	w_2	w_4	w_6
3	w_3	w_2	w_5
4	w_4	w_5	w_1

Term	Frequency	Mean Rank
w_2	3	2.0
w_3	2	2.0
w_1	2	2.5
w_4	2	3.0
w_5	2	3.5
w_6	1	2.0

in Algorithm 1 (line 5). Next, we add all keyqueries $\mathcal{Q}_{KQC}$ to $\mathcal{Q}_{level}$, which are not yet in the set $\mathcal{Q}$ of keyqueries (line 6). The generation of the set $\mathcal{Q}_{level}$ of keyqueries is finished when all combinations of the current level, i. e., all document sets with a length of $level$, are processed.
In the next step, the number n_{level} of candidate documents per keyquery of the current level is computed. We calculate how many candidate documents can be added to $\mathcal{C}$ and divide this number by the number $|\mathcal{Q}_{level}|$ of keyqueries obtained in the current level (line 7). Then, we add all keyqueries $q \in \mathcal{Q}_{level}$ to $\mathcal{Q}$, and we add the corresponding top n_{level} results of each keyquery to the candidate set $\mathcal{C}$ (lines 8-10). Note that not each keyquery $q \in \mathcal{Q}_{level}$ necessarily returns as many results as it could contribute, i. e., n_{level} results, to the candidate set $\mathcal{C}$. Consequently, $\mathcal{C}$ may not contain the desired number n of documents after the first level. Furthermore, it is possible that for none of the combinations of the current level any keyqueries can be found. As a result, no candidate documents can be added for that level. At the end of each level, we clear the set $\mathcal{Q}_{level}$ of keyqueries (line 11). If the number $|\mathcal{C}|$ of candidate documents matches the desired number n, the algorithm stops (line 12); otherwise, the next level is processed. Finally, the set $\mathcal{C}$ of candidate documents is returned by Algorithm 2 (line 13).
In our experiments we set the level of keyquery generality to $k = 10$ and the lower bound on the result list length to $l_{min} = 10$. Moreover, we set the number of candidate documents per input document to $n_i = 100$ to be comparable with the aforementioned baselines that also retrieve 100 candidate documents per input document.

Note that the result of the algorithm is strongly dependent on the vocabulary extracted from the input documents. In this thesis, we investigate the effects of two different strategies for selecting a document's vocabulary; we refer to them as Strategy A and Strategy B. In Strategy A, we take the top-20 keyphrases, i. e., terms with at least two

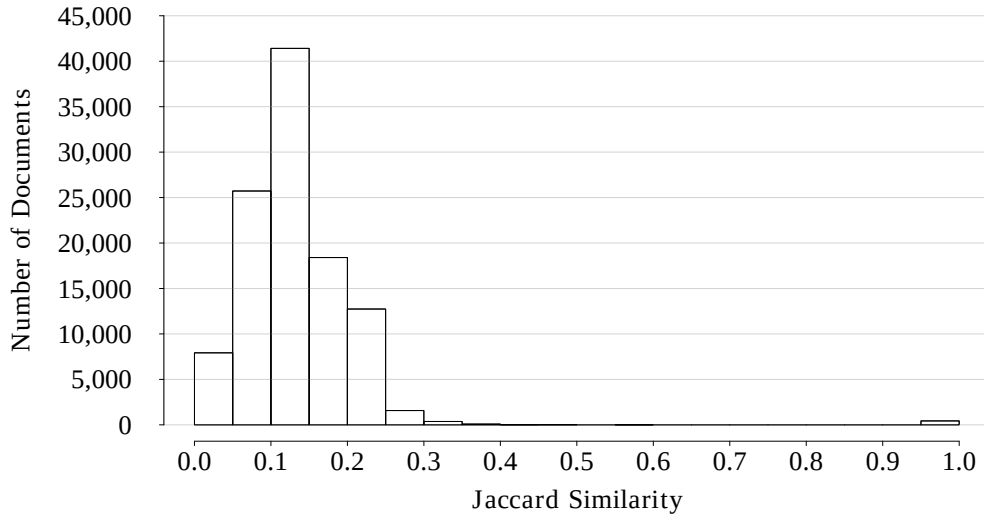

Figure 3.2: Distribution of Jaccard similarities of the vocabularies W_A and W_B for documents of the WCSP that are available as PDF-file.

words, obtained by applying a keyphrase extractor to a document's text. We choose KP-Miner by El-Beltagy and Rafea [ER09] since it is the best unsupervised keyphrase extractor for research papers according to the SemEval Competition 2010 [KMKB10]. We refer to Algorithm 2 using vocabulary selection Strategy A as *KQC/A*.
Strategy B for the selection of vocabulary is more complex. First, the top-50 keyphrases out of a document's text are extracted by using KP-Miner and the paper's keywords that have been specified by the author are added to this set. We then keep the 20 keyphrases that often appear as a paper's keyword specified by a paper's author in the underlying data collection, namely in the Webis Computer Science Paper Corpus (WCSP). Since a paper's keywords are specified by its author, we assume that they are very meaningful. Subsequently, we sort the remaining keyphrases by their frequency in descending order and take the top-20 phrases as vocabulary of the given document. We refer to Algorithm 2 using this second vocabulary selection method as *KQC/B*.
Table 3.3 shows the vocabularies W_A and W_B obtained with strategy A and B for the research paper "Latent dirichlet allocation" by Blei et al. [BNJ03]. For Strategy B, the keyword frequencies in the WCSP of the keyphrases are also listed. Note that terms which describe the same topic can be included in one vocabulary, e. g., both terms "latent dirichlet allocation" and "lda" are contained in vocabulary W_B. In future work, this can be avoided by using a dictionary of abbreviations and synonyms. However, the table indicates that the two methods extract different vocabularies for this example.

Table 3.3: Vocabularies W_A and W_B extracted from the research paper "Latent dirichlet allocation" by Blei et al. [BNJ03]. The emphasized terms are present in both vocabularies W_A and W_B.

	Vocabulary W_A		Vocabulary W_B
Rank	Term	Frequency	Term
1	generative probabilistic model	1267	*machine learning*
2	edu computer science	1042	classification
3	unigrams model	671	*collaborative filtering*
4	underlying set	318	*text classification*
5	mixture components	268	*dimensionality reduction*
6	text modeling	238	algorithm
7	text corpora	121	model
8	*collaborative filtering*	81	*latent dirichlet allocation*
9	*parameter estimation*	77	*latent semantic indexing*
10	finite mixture	73	distribution
11	*latent dirichlet allocation*	49	lda
12	discrete data	31	mixture model
13	*text classification*	28	*parameter estimation*
14	*machine learning*	25	corpus
15	empirical bayes	23	topic
16	infinite mixture	23	lsi
17	document modeling	23	generative model
18	*latent semantic indexing*	19	representations
19	plsi model	17	maximum likelihood
20	*dimensionality reduction*	13	log

We can measure the similarity of the two vocabularies, i. e., two sets of terms, using the Jaccard coefficient [TSK05]. For two sets W_A and W_B the Jaccard coefficient is defined as the size of the intersection divided by the size of the union of the sets:

$$J(W_A, W_B) = \frac{|W_A \cap W_B|}{|W_A \cup W_B|}. \tag{3.6}$$

In our example in Table 3.3, the vocabularies W_A and W_B share 7 phrases which are emphasized in Table 3.3. Therefore, the intersection of vocabularies W_A and W_B contains 7 elements, and the union contains 33 elements. As a result, the Jaccard coefficient for this example is $J(W_A, W_B) = \frac{7}{33} = 0.21$.
The distribution of the Jaccard similarities for all documents in the WCSP that are available in fulltext is shown in Figure 3.2. The figure illustrates that the Jaccard

coefficient is between 0.0 and 0.3 for most of the documents. Thus, the two strategies indeed produce different vocabularies.

Candidate Ranking

In the previous paragraphs, we described our technique for obtaining a set $\mathcal{C}$ of candidate documents in detail. We rank the candidate documents in $\mathcal{C}$ by applying the same ranking procedure as in our implementation of Sofia Search: we compute the pairwise text similarity of a candidate document d_c to the set $\mathcal{I}$ of input documents and take the maximum value as a relevance score (see Equation 3.5).

3.3 Combination Technique

Beside the aforementioned methods, we investigate the combination of our keyquery-based approach with the presented baseline algorithms. Therefore, we combine the result lists produced by the methods as previously described for the lists of related articles retrieved from Google Scholar for several input documents (see Section 3.1.3). The following paragraphs summarize our strategy for combining several Related Work Search approaches. For reasons of clarity, we explain our combination strategy for two approaches: Method 1 and Method 2.

Candidate Retrieval

First, we obtain two separate candidate sets $\mathcal{C}_1$ and $\mathcal{C}_2$ using the respective approach for candidate retrieval of the two methods.

Candidate Ranking

We rank the two sets $\mathcal{C}_1$ and $\mathcal{C}_2$ of candidate documents individually according to their maximum text similarity to the set of input documents $\mathcal{I}$ as described before (see Equation 3.5). As a result, we get two sets of output documents $\mathcal{O}_1$ and $\mathcal{O}_2$. Finally, we mix $\mathcal{O}_1$ and $\mathcal{O}_2$ by alternatingly adding documents to the merged list of output documents $\mathcal{O}$. First, we add a document from $\mathcal{O}_1$ and then from $\mathcal{O}_2$ in a rank-wise manner. Since a document can be found by both methods, it is possible that the merged list $\mathcal{O}$ contains duplicate documents. We address this issue by keeping the document that is higher ranked and remove all following duplicate documents.

Note that the strategy is also applicable for more than two methods as was illustrated before with an example of the ranking procedure for the Google Scholar baseline (see Table 3.1).

Summary

In this chapter, we have presented three baselines and two versions of a novel keyquery-based algorithm for finding related research papers for a given list of input papers. Furthermore, we have described a technique to combine several approaches. The next chapter analyzes the performance of the presented methods.

4 Evaluation

In this chapter, we describe the experimental evaluation of the Related Work Search approaches described in Chapter 3. In order measure the effectiveness of these methods, we conduct a Cranfield-style experiment. We use an index of a large collection of research papers, which builds the first component of our experiment. Since the other two components, namely topics and relevance judgments, are not available for this collection, we conduct a user study to obtain them. Based on the topics and relevance judgments the lists of recommended papers produced by several Related Work Search approaches are evaluated.

In Section 4.1, our experimental design is specified in detail. The results of our experiments are presented and analyzed in Section 4.2.

4.1 Experimental Design

Generally speaking, there are two different approaches to evaluate algorithms for research paper recommendation. A widely used method is to use the reference lists of papers as ground truth for the purpose of evaluation. In this approach, usually some references are hidden from the list and tried to recover with a Related Work Search algorithm. In contrast, the second evaluation method uses relevance judgments assigned by humans. These judgments state whether a document is relevant to a specific topic or not.
Note that the first method is based on the citation graph. As stated before, the citation graph tends to be noisy and sparse because of space limitations and missing or erroneous citations [CSMG13]. Consequently, reference lists are not a good basis for evaluation. Thus, we choose the second approach and utilize human relevance judgments.

In the following sections, we first analyze the collection which is used in our experiments, namely the Webis Computer Science Paper Corpus (Section 4.1.1). Then, we state how this collection was made accessible, i. e., how it was indexed (Section 4.1.2). Last, our user study for obtaining test cases and human relevance judgments is described (Section 4.1.3).

Table 4.1: Metadata record of the paper "Finding text reuse on the web" by Bendersky and Croft [BC09] in the WCSP.

Field	Content
ACM ID	1498835
Type	PDF
Title	Finding Text Reuse on the Web
Authors	Michael Bendersky, W. Bruce Croft
Conference	WSDM
Series	WSDM '09
Year	2009
Abstract	With the overwhelming number of reports on [...]
Keywords	Text reuse, information flow, web search
Index Terms	Information Systems, INFORMATION STORAGE AND RETRIEVAL, Information Search and Retrieval
Reference Count	23
References	1092473, 1341557, 1390432, [...]
Citation Count	18
Citations	2487688, 1840829, 2399184, [...]

4.1.1 The Webis Computer Science Paper Corpus

We conduct our experiments on the *Webis Computer Science Paper Corpus* (WCSP), which comprises 186,797 research papers from the domain of Computer Science published by ACM. Note that the WCSP has been used for the evaluation of the concept of keyqueries [GHMS13].

Construction of the WCSP

The WCSP was constructed over the course of several years by the Webis group. The construction was divided into two steps. First, a seed of 34,280 research papers from 20 different conferences have been collected (see Table A.1 for further information on the seed documents). In the second step, cited and citing papers of these seed documents were added if they were available in the ACM digital library. As a result the WCSP contains 186,797 documents.
The Webis group stored not only the PDF-files of the documents but also the metadata that was available on the web page of the ACM digital library. Since the metadata available on ACM was specified by humans, the metadata contained in the WCSP is of good quality. For example, in Table 4.1 the metadata for the paper "Finding text reuse on the web" by Bendersky and Croft [BC09] is listed. The first line states the ACM ID of

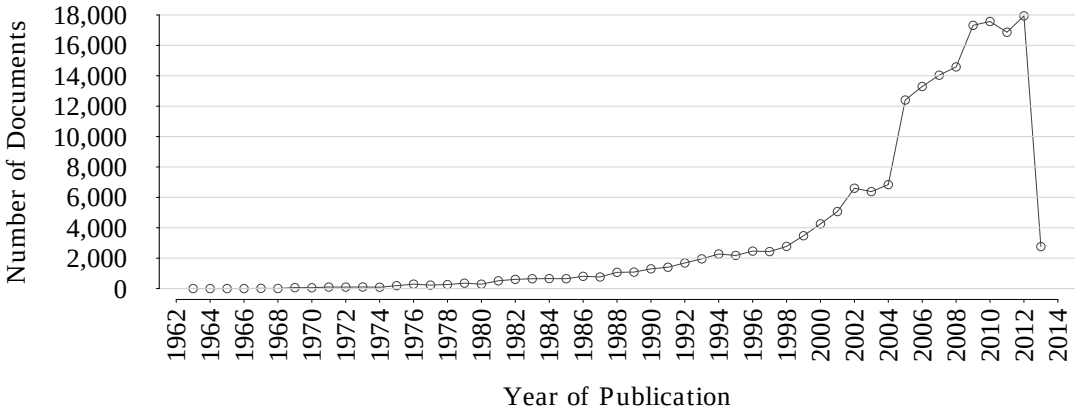

Figure 4.1: Distribution of publication years of documents in the WCSP.

the paper, which is used as an unique identifier in the WCSP. The field "Type" indicates that for the document not only metadata but also a PDF-file is available. This also holds for 56.9 % (106,525 of 186,797) of the documents in the WCSP. The remaining 43.1 % (80,272 of 186,797) of the documents are only represented as metadata since the PDF-file was not accessible. In addition to bibliographic data like title, authors, conference, series and year, the metadata comprises the abstract of the paper, the keywords specified in it and the index terms according to the *ACM Computing Classification System*. Moreover, information regarding cited and citing papers are included. Note that only papers that can be found in the ACM digital library, i. e., papers with an ACM ID, are included as citation or reference of a document in the WCSP. Consequently, not all actual references and citations of a paper are contained in the corpus. For instance, the paper listed in Table 4.1 actually has 30 references, as can be seen in the respective PDF-file, but only 23 of them are recorded in the WCSP since the other 7 references do not have an ACM ID. Furthermore, it is to say that not all of the recorded references and citations are present in the WCSP. In fact, the WCSP only contains all cited and citing papers with an ACM ID for the seed documents.

Key Figures

In the following paragraphs, we present some key figures and statistics of the documents included in the WCSP regarding three criteria: (1) year of publication, (2) publication venue, and (3) citation information.

Year of Publication
The documents included in the WCSP have been published in the years 1962–2013. The oldest publication in the corpus is "Dynamic programming treatment of the travelling

Table 4.2: The five most frequent conferences and journals in the WCSP. The value in column $|D|$ refers to the number of documents in the WCSP that have been published in the respective publication venue.

Conferences		Journals	
|D|	Acronym	|D|	Title
6,756	CHI	1,807	SIGPLAN Notices
4,033	ICSE	1,344	Theoretical Computer Science
3,934	SIGMOD	1,194	Computer Communications
3,778	CIKM	1,173	Computer Netwetworks
3,114	SIGIR	1,088	SIGCOMM Computer Communication Review

salesman problem" by Bellman [Bel62]. Since the Webis group finished the construction of the corpus in 2013, no more recent papers are included. Figure 4.1 shows the distribution of publication years of the documents in the WCSP. The majority of the documents contained in the corpus, or more precisely 71.5 % (133,621 of 186,797), have been published in the last ten years.

Publication Venue
The WCSP comprises papers which have been published in 3,232 different conferences and 749 different journals. The five most frequent conferences and journals in the corpus are listed in Table 4.2. About 3.6 % (6,756 of 186,797) of the documents in the WCSP have been published in the proceedings of the "International Conference on Human Factors in Computing Systems" (CHI). The most frequent journal is the monthly letter "SIGPLAN Notices", which is the publication venue of 1.0 % (1,807 of 186,797) of the articles in the WCSP.

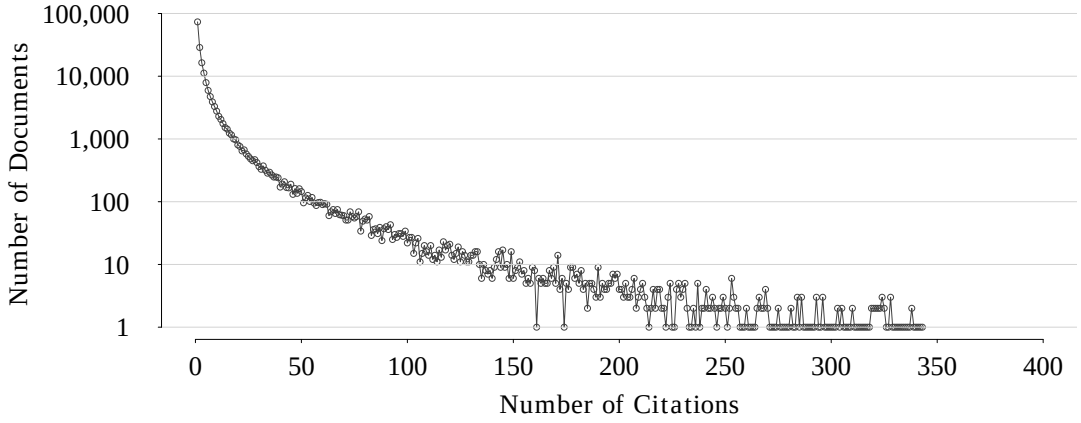

Figure 4.2: Distribution of the number of citations of a document that are present in the WCSP. For the purpose of illustration, we only show the interval $[0, 400]$ on the x-axis. Note that the y-axis is scaled logarithmically.

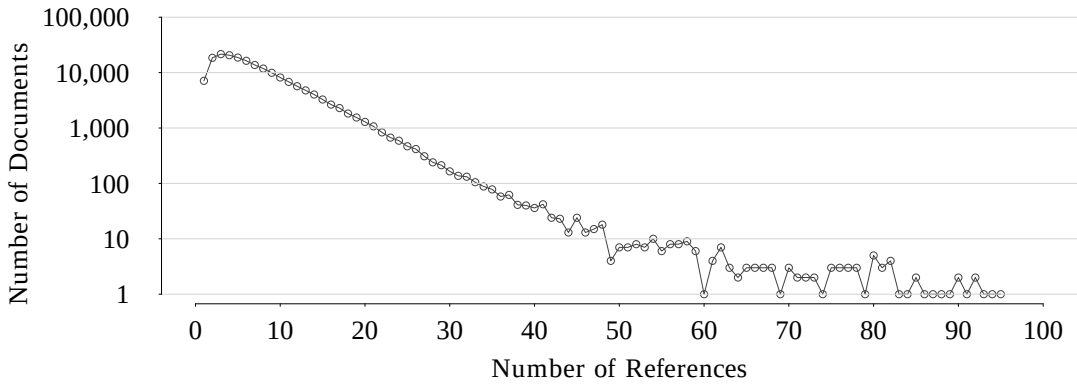

Figure 4.3: Distribution of the number of references of a document that are present in the WCSP. For the purpose of illustration, we only show the interval $[0, 100]$ on the x-axis. Note that the y-axis is scaled logarithmically.

Citation Information
The distribution of the number of references is illustrated in Figure 4.3. Note that the number of documents refers to the documents which are present in the WCSP. For the purpose of illustration, the x-axis is limited to $[0, 100]$. There are also documents with more than 100 references in the WCSP; for example, the journal article "Algorithms and data structures for external memory" by Vitter [Vit06] has 142 references in the WCSP. On average a document in the corpus has 6.5 references that are present in the WCSP and is also cited by 6.5 documents that are included in the WCSP. The distribution of the cited papers contained in the WCSP can be seen Figure 4.2. It turns out that many of the documents have no citing papers that are contained in the corpus, i. e., 39.4 % (73,527 of 186,797). Almost half of the documents, or more precisely 45.5 % (84,916 of 186,797), have between one and ten citing documents that are in the corpus. However, there are also documents with a lot of citations. For instance, Table 4.3 lists the top five cited papers in the WCSP. The most cited in the corpus is "Fast algorithms for mining association rules in large databases" by Agrawal and Srikant [AS94] with 2,043 citing documents.

Table 4.3: The five most cited papers in the WCSP. Note that the values in column "Citations" refer to the number of citing papers that are present in the WCSP.

Citations	Title
2,043	Fast algorithms for mining association rules in large databases
2,021	Mining association rules between sets of items in large databases
1,775	Chord: a scalable peer-to-peer lookup service for internet applications
1,326	R-trees: a dynamic index structure for spatial searching
1,322	A scalable content-addressable network

4.1.2 Experiment Setup

In order to run the Related Work Search algorithms described in Chapter 3, we use a retrieval system that indexes the documents of the WCSP. Specifically, they were indexed with the open source information retrieval software library Lucene by Apache[1]. Since Lucene supports fielded data, all fields of a document available in the WCSP as described previously (see Table 4.1) can be indexed. The ranking function applied in the retrieval system is BM25F, which is an extension of BM25 [RZT04]. Note that both ranking functions are based on the bag-of-words model in which a text is represented as a set of its words. Therefore, the relevance of a query to a document is computed according to terms, i. e., words or phrases, of the query appearing in the document. Whereas BM25 is only applicable to documents in form of one text, BM25F is capable of documents composed from several fields such as title, abstract and text.

In the used retrieval system, Lucene's implementation of BM25F is utilized, which defines the relevance $R(q, d)$ of a query q to a document d as follows [PPFF09]:

$$R(q,d) = \sum_{t \in q} idf(t) \cdot \frac{weight(t,d)}{k_1 + weight(t,d)}\,, \tag{4.1}$$

where k_1 is a free parameter that is usually set to $k_1 = 2$. The equation includes two further terms to be defined.

First, the inverse document frequency $idf(t)$ is calculated as:

$$idf(t) = \log \frac{N - df(t) + 0.5}{df(t) + 0.5}\,, \tag{4.2}$$

where N refers to the number of documents in the collection, and $df(t)$ states in how many documents the term t appears.

Second, $weight(t, d)$ defines the weight of a term t over the fields f in the document d:

$$weight(t,d) = \sum_{f \in d} \frac{tf(q,f) \cdot boost_f}{(1 - b_f) + b_f \cdot \frac{|f|}{|f|_{ave}}}\,, \tag{4.3}$$

where $tf(q, f)$ specifies the term frequency of term t in field f of document d. The variable $|f|$ refers to the length of the field f, and $|f|_{ave}$ refers to the average length of the field f in the collection. Note that using BM25F it is possible to assign different levels of importance to the fields $f \in d$ of a document d, which is accomplished by the boost factor $boost_f$. The parameter b_f is a free parameter, and it is usually set to $b_f = 0.75$ which we adapt.

[1] `http://lucene.apache.org/`, Last accessed: August 18th, 2014

In the used retrieval system, not all of available fields of a document are used since not all fields contain information on a document's topic. For example, the names of a document's authors or its citation count do not directly contain information about the topics handled in the document. Thus, only the following three fields are considered for the ranking function of BM25F: title, abstract and text. Moreover, the title is set to be the most important field followed by the abstract and finally by the content of the document.

4.1.3 User Study

As stated above, we want to analyze our algorithms for research paper recommendation by means of user evaluation. Therefore, we apply the well-known *Cranfield paradigm* for measuring the effectiveness of a retrieval system [Voo01]. Experiments following the Cranfield paradigm utilize a test collection, which consists of three components: (1) a set of documents, (2) a set of information needs called *topics*, and (3) a set of relevance judgments [Voo01]. The relevance judgments for one topic state which of the documents are relevant to the topic and which are not. In order to compare different retrieval approaches, the Cranfield paradigm has been applied in several evaluation conferences, e. g., in the *Text REtrieval Conference* (TREC) and in the *Conference and Labs of the Evaluation Forum* (CLEF) [Voo01].

As we already stated in previous paragraphs, we use the WCSP as a set of documents for the purpose of evaluation. Thus, the WCSP forms the first component of our Cranfield-style experiment. None of the two other components, namely topics and relevance judgments, is available for the WCSP. Therefore, we conduct a user study to obtain these two components. We first ask users to specify a topic, i. e., a set of input documents. Next, we obtain related documents for the input papers by applying several Related Work Search algorithms. The related documents are then judged by the users with respect to their relevance. In order to ensure a comfortable workflow for the user, we built a web interface for conducting the user study. The interface summarizes all information that the user needs on two web pages. Moreover, the web interface of the user study allows a user to participate in our study without being on site.

Procedure

The user study consists of two steps, which we denote as Step 1 and Step 2. Basically, we obtain topics in Step 1, and we collect relevance judgments in Step 2. For each step we provide a separate web interface, which can be seen in Figure 4.4 and Figure 4.5 illustrating Step 1 and Step 2 respectively.

In Step 1, we first ask the participants to enter a research task they are familiar with (see "Research Task" in Figure 4.4). We request a familiar research task because expert knowledge is required later in order to judge the relevance of the recommended documents.

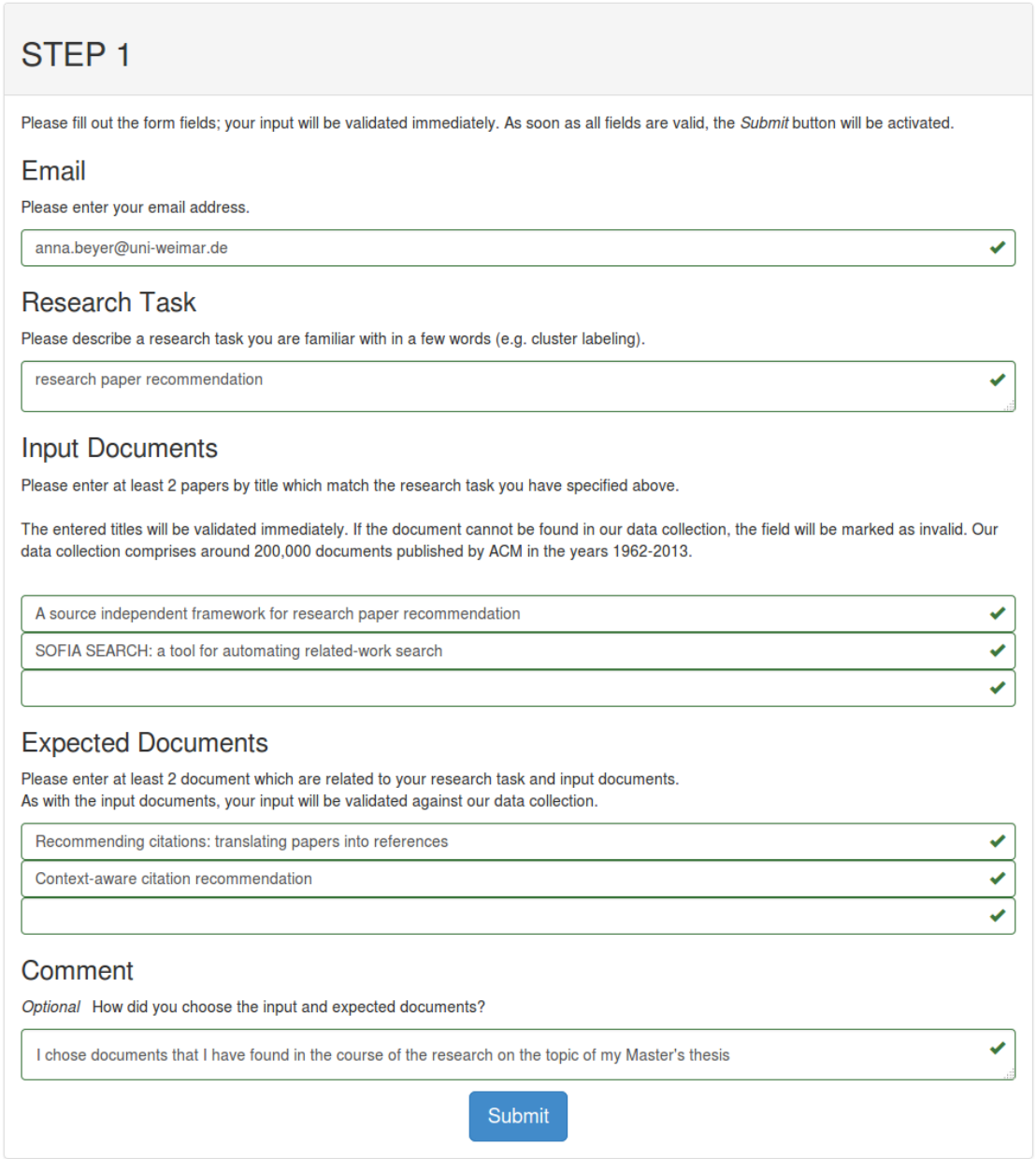

Figure 4.4: The web interface for obtaining topics in Step 1 of our user study.

Next, the users have to enter at least two input documents by title, which match their research task (see "Input Documents" in Figure 4.4). While the user is entering a title of a research paper, it is automatically checked whether the specified paper exists in the WCSP or not. If the document cannot be found in the collection, the user is notified to enter another title. Additionally, the participants have to name at least two papers that they expect to be found (see "Expected Documents" in Figure 4.4). Last, the users could optionally describe how they chose input documents and expected documents (see "Comment" in Figure 4.4).

After a user has completed Step 1, we obtain related papers for the specified set $\mathcal{I}$ of input documents by applying five Related Work Search algorithms, where three are baseline algorithms (see Section 3.1):

(1) Version of Sofia Search by Golshan et al. [GLT12]
(2) Simple query-based method by Nascimento et al. [NLSG11]
(3) Baseline that uses Google Scholar's feature to find related articles,

and two are novel keyquery-based methods (see Section 3.2):

(5) KQC/A
(6) KQC/B.

Since the Google Scholar baseline could not be automated due to the restrictions of the commercial search engine, we cannot fully automate the process of the user study experiment. Thus, we run the algorithms separately and notify the users via email when Step 2 is available for their topic.

The purpose of Step 2 is to obtain relevance judgments. Since it would be to much effort to judge all of the 186,797 documents in the WCSP for each topic, we only collect judgments for the documents that we need for the evaluation of the five algorithms. Therefore, we ask the users of our study to judge the top-10 documents returned by each of the algorithms on the topic they have specified in Step 1. If each algorithm returns different top-10 results in comparison to the other algorithms, a user has to judge 50 documents, which is an acceptable number. A larger number of documents would take to much effort as the judgment would take to much time and the users do not get any reward for their work. So, we present a list of at most 50 recommended papers, which contains the top-10 documents computed by the five algorithms, to the user (see "Related Documents" in Figure 4.5). In order to avoid bias caused by the documents' ranking, we display the papers in alphabetical order by title. For each paper the fields title, authors, publication venue, publication year and abstract are listed. Additionally, a hyperlink to the respective PDF-file is listed if the paper is available as PDF in the WCSP. Thus, the users could read the PDF-file if they need more detailed information that are not stated in the paper's abstract.

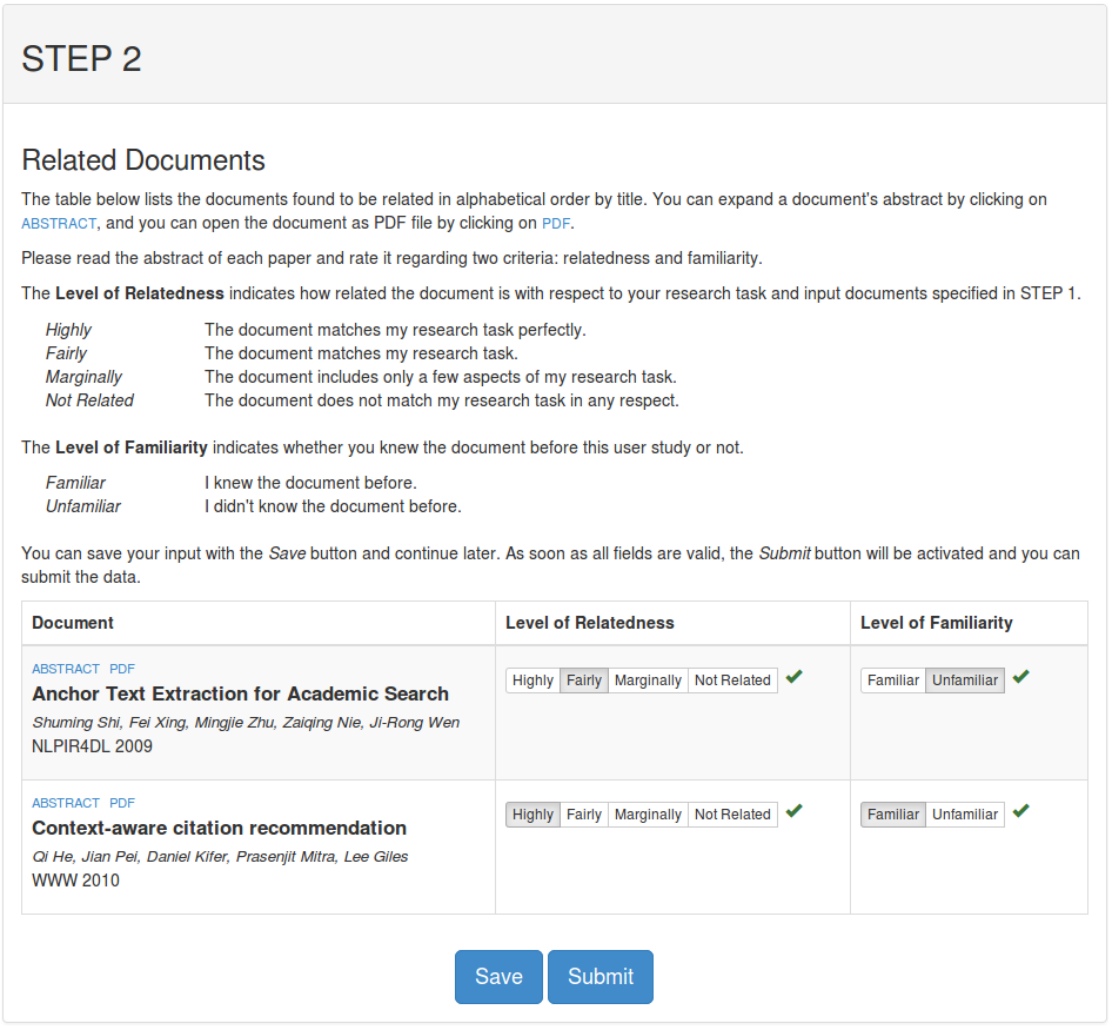

Figure 4.5: The web interface for obtaining relevance judgments in Step 2 of our user study.

Table 4.4: Description of the levels of relatedness and familiarity.

Judgment	Description
Highly	The document matches my research task perfectly.
Fairly	The document matches my research task.
Marginally	The document includes only a few aspects of my research task.
Not Related	The document does not match my research task in any respect.
Familiar	I knew the document before.
Unfamiliar	I didn't know the document before.

The users were asked to rate the papers according to two criteria: relatedness and familiarity. The level of relatedness defines how related, i. e., how topically similar, the listed document is with respect to the specified input documents and the respective research task. We decide to rate the relatedness on a 4-point scale because binary judgments would not be detailed enough and a 3-point scale would allow to choose a neutral judgment from which no trend with respect to the relatedness could be derived. Therefore, the relatedness is rated on the following 4-point scale: *highly*, *fairly*, *marginally*, and *not related*. Moreover, we obtain additional judgments with respect to the level of familiarity; it is rated as *familiar* if the user did know the paper before or as *unfamiliar* if not. By the combination of the two criteria, we want to determine which documents are relevant but unfamiliar to the user. The verbal description of the levels of relatedness and familiarity are listed in Table 4.4.

To summarize, this section has explained the user study that we conduct in order to obtain topics and judgments for our Cranfield-style experiment. In the next section, we first describe the results of our user study and then analyze the performance of the Related Work Search algorithms.

4.2 Results

This section presents the results of our experiments. First, some key figures of the conducted user study (Section 4.2.1) are summarized. In the second part, we explain the metrics that we use in order to measure the performance of the Related Work Search algorithms and present the results (Section 4.2.2).

4.2.1 Characteristics of the User Study

Our study has involved 10 research associates and experienced Master's students from the degree program Computer Science and Media at the Bauhaus-Universität Weimar. We have obtained topics in Step 1 and judgements in Step 2 of the study (see Section 4.1.3).

Topics

Each of the participants has stated one information need in Step 1 in our study; consequently, we have 10 topics each with a set $\mathcal{I}$ of at least two input documents. Table A.2 in the appendix provides an overview of the topics, which contain a short description of the research task and the respective input documents. The set of topics comprises various research tasks from different fields of Computer Science. Note that topic 006 and 007 are the only topics that deal with a similar research task. As can be seen in Table A.2 in the appendix, 7 of the 10 topics, i. e., topics 001, 002, 003, 004, 007, 008 and 009, comprise two input documents. For the remaining topics, i. e., topic 005, 006 and 010, three input documents have been specified.

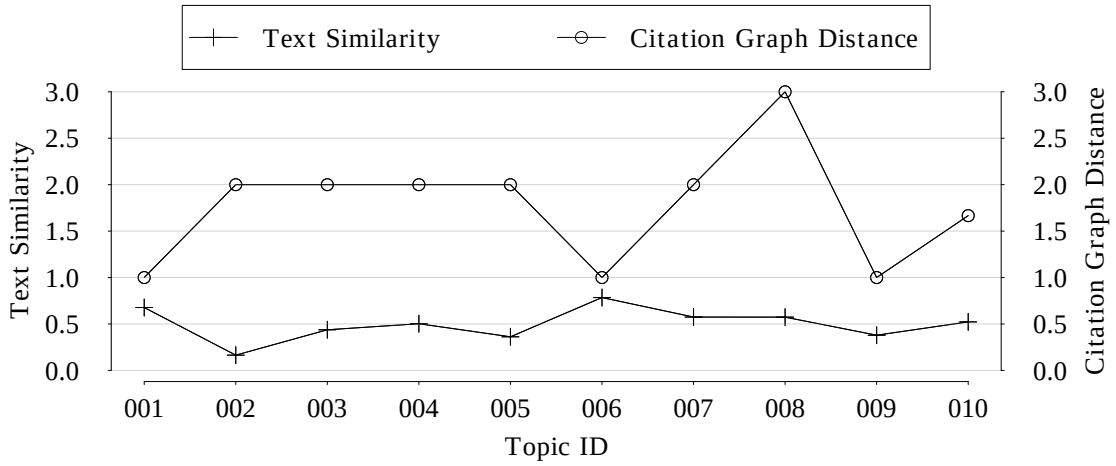

Figure 4.6: Average pairwise text similarity and average pairwise distance in the citation graph of the set of input documents for a topic.

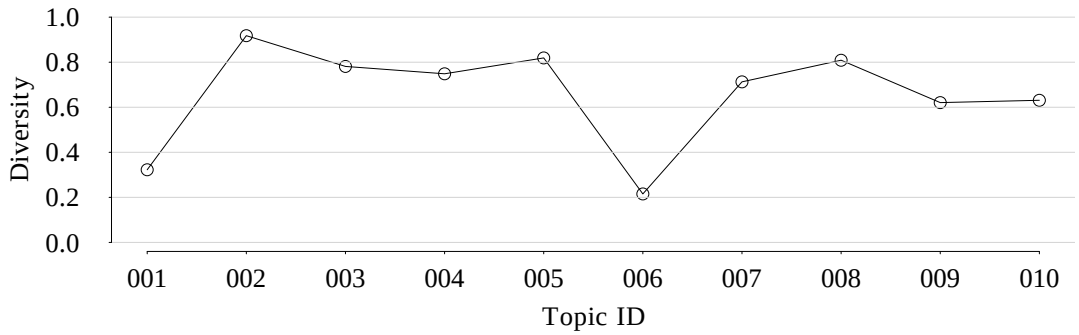

Figure 4.7: Diversity of the set of input documents for each topic.

In order to classify the difficulty of a topic, we analyze the set $\mathcal{I}$ of input documents. Figure 4.6 illustrates the average pairwise text similarity and the average pairwise distance in the citation graph for each topic. Note that the citation graph distance refers to the length of the shortest path connecting two documents in the citation network. As can be seen in Figure 4.6, the two input documents of topic 008 (see Table A.2) have the highest citation graph distance with a value of 3. This means that the two documents are connected over a path of length 3 in the citation network. On the other hand, the input papers of the topics 001, 006, and 009 have an average citation graph distance of 1, which means there is a direct citation relation between them. Note that topic 006 has even three input documents (see Table A.2) that are all directly connected in the citation graph. So, they are more likely to deal with similar topics than the input documents specified in topic 008. On average, the input papers $d_i \in \mathcal{I}$ of a topic are connected over a path of length 1.8.
The text similarity is computed as cosine similarity of the term frequency vectors of two texts as stated in Chapter 3. On average, a set $\mathcal{I}$ of input documents has a text similarity of 0.5. The input papers of the topics 001 and 006 are the most similar, whereas the documents specified in topic 002 are the least similar. We assume that a high text similarity of two documents indicates a high topical similarity.

The citation graph distance of two documents and their text similarity can be used to measure the diversity of these documents. A low text similarity of two documents indicates that they do not share many terms; hence, they handle diverse topics. Documents that have a large citation graph distance are more likely to handle different topics than documents which are directly cited. According to these assumptions, we define the $diversity(\mathcal{I})$ of a set $\mathcal{I}$ of input documents as follows:

$$diversity(\mathcal{I}) = \frac{1}{\binom{|\mathcal{I}|}{2}} \cdot \sum_{i,j \in [1,|\mathcal{I}|]: i<j} \left(1 - \frac{text\_similarity(d_i, d_j)}{citation\_graph\_distance(d_i, d_j)}\right), \tag{4.4}$$

Table 4.5: Overlap of top-10 recommended papers over all topics for all pairs of results lists produced by the respective algorithms.

	Sofia Search	Nascimento	Google Scholar	KQC/A	KQC/B
Sofia Search	100	26	27	55	55
Nascimento	26	100	16	25	26
Google Scholar	27	16	100	30	24
KQC/A	55	25	30	100	49
KQC/B	55	26	24	49	100

where $d_i, d_j \in \mathcal{I}$ is a pair of documents from $\mathcal{I}$. The summand in the equation refers to the diversity of the pair $d_i, d_j \in \mathcal{I}$ of input documents. The $diversity(\mathcal{I})$ of a set $\mathcal{I}$ of input documents is the average pairwise diversity of the included input documents. Two documents presumably deal with similar topics if they have a high text similarity and a low citation graph distance. In this case, the fraction of text similarity and citation graph distance in the equation becomes high. We subtract the fraction from 1 in order to map high similarity to a low value of diversity.
We assume that the diversity of $\mathcal{I}$ reveals how difficult a topic is, i. e., topics with low diversity are easier to solve than topics with high diversity.
Figure 4.7 illustrates the diversity for each set $\mathcal{I}$ specified per topic obtained in the user study. It turns out that the input documents of topic 006 are the least diverse with a diversity of 0.22. The small diversity value is caused by a high text similarity and a small citation graph distance (see Figure 4.6) which indicate that the input documents of topic 006 are very similar. Therefore, we assume that topic 006 is an easy one. In contrast, topic 002 has a very high diversity of 0.92 and is therefore more difficult than topic 006. The average diversity of all sets $\mathcal{I}$ of input documents is 0.66. In the course of the performance analysis, we will later refer to the diversity of the set $\mathcal{I}$ of input documents of a topic.

We use the 10 topics collected in our user study to run five Related Work Search approaches: (1) Sofia Search, (2) Nascimento, (3) Google Scholar, (4) KQC/A, and (5) KQC/B (see Chapter 3). As a result, we have obtained five sets $\mathcal{O}$ of output papers. As discussed previously, we took the top-10 documents recommended by each of the five algorithms and constructed a merged list. If each algorithm returns distinct top-10

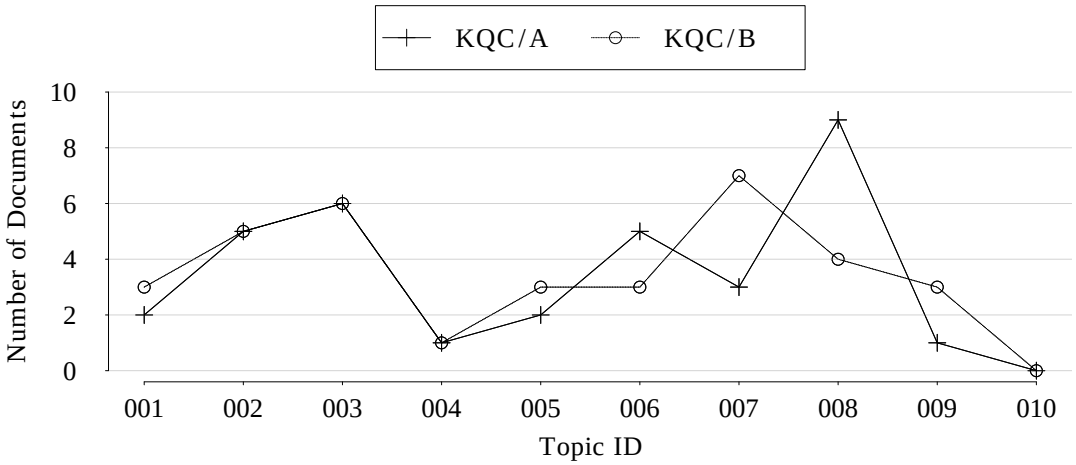

Figure 4.8: Number of documents that have been listed in the top-10 results of one of the keyquery-based methods but not in the top-10 results produced by the three baseline algorithms.

results in comparison to the results lists of the other algorithms, a user has to judge 50 documents. However, in reality the sets are not distinct. Table 4.5 lists the overlap of the top-10 recommended papers for each combination of algorithms. Note that the values in Table 4.5 are summed up over all 10 topics. That is, the value in each cell can be at most 100 since it describes the overlap of the top-10 results of two algorithms for 10 different topics. In fact, the main diagonal of the symmetric matrix presented in Table 4.5 is filled with 100. On average, the top-10 recommended papers of two different algorithms share 3.5 documents. In total, the highest overlap of 5.5 documents is found in the result lists of Sofia Search and KQC/A, and the lists of Sofia Search and KQC/B. Thus, the two keyquery-based methods found on average around half of the documents that the graph-based approach of Sofia Search has found as well and vice versa. Consequently, about half of the documents that are listed in the top-10 results of a keyquery-based method are not listed in the top ranks of the graph-based approach. Note that KQC/A and KQC/B also have found different documents in the top-10 results than Google Scholar has found. The smallest overlap of top-10 documents is reported for Google Scholar and Nascimento; their results lists share on average only 1.6 documents in the top-10 ranks.

In short, Table 4.5 reveals that the result lists produced by the five different Related Work Search algorithms indeed overlap. Another interesting question is: How many documents have been found by a keyquery-based method but not by the other methods? Figure 4.8 illustrates the number of documents that have been ranked in the top-10 results of one of the keyquery-based methods, i. e., KQC/A or KQC/B, but not in the

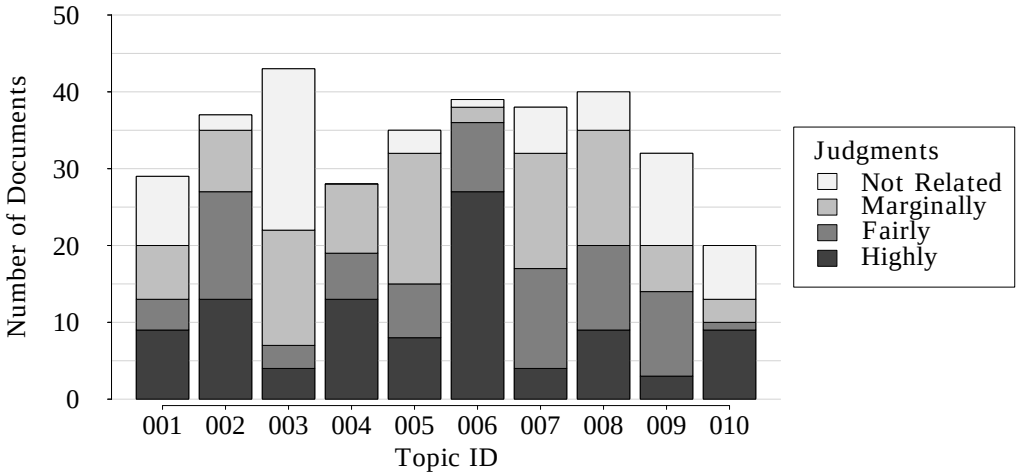

Figure 4.9: Distribution of relevance judgments for each topic.

top-10 results produced by the three baselines: Sofia Search, Nascimento and Google Scholar. Figure 4.8 reveals that both keyquery-based methods indeed find documents which are not listed in the 10 top-ranked results of any of the other three methods. On average KQC/A found 3.4 and KQC/B found 3.5 documents that the baseline methods miss in the top-10 results.

Judgments

In Step 2 of our user study, we collected judgments for the recommended documents. The distribution of judgments of relatedness is illustrated in Figure 4.9. The number of judgments per topic refers to the number of different documents found by the five Related Work Search algorithms. The figure shows that the users have judged different amounts of documents. For example, the participant who treated topic 003 had to judge 43 documents. This means that the all top-10 documents recommended by the five algorithms only overlap in 7 documents. In contrast, the five algorithms found very similar papers for topic 010, where the user only had to judge 22 papers. Overall, 341 documents have been judged by 10 users; therefore, a user has judged on average 34 documents. Furthermore, Figure 4.9 shows how many documents received which judgment of relatedness per topic. For instance, many of the documents recommended for topic 006 are judged as highly relevant, whereas many of the documents retrieved for topic 003 are judged as not related. However, the figure does not reveal any general trend regarding the distribution of the levels of relatedness within the list of recommended documents for a topic.

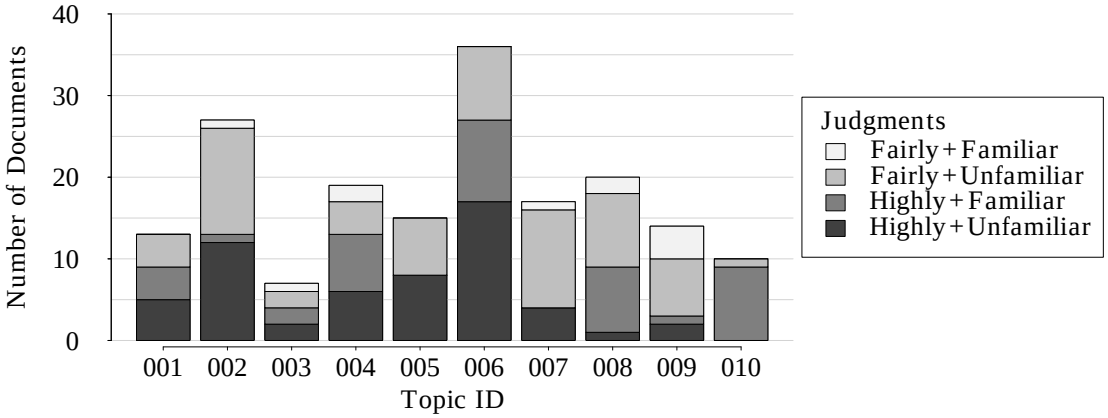

Figure 4.10: Number of highly or fairly related documents distinguished by their level of familiarity.

In our user study, we asked the users to additionally specify whether they knew each of the recommended documents before the study or not. The idea was to find *unfamiliar relevant papers*, i. e., documents that are highly or fairly related to the specified research task and that the user did not know before the study. Figure 4.10 illustrates the level of familiarity of the documents that were judged as highly or fairly relevant. In total, 86.1 % (68 of 79) of the documents that are judged as fairly related were unfamiliar to the user. Moreover, the users did not know 57.6 % (57 of 99) of the highly relevant papers before the study. All in all, 125 unfamiliar relevant papers have been found by the five algorithms.

Discussion

To sum up, this section has presented and analyzed the topics and judgments obtained in our user study. These data sets form the two missing components of our Cranfield-style experiment. Thus, we can now measure the effectiveness of the five Related Work Search approaches: (1) Sofia Search, (2) Nascimento, (3) Google Scholar, (4) KQC/A, and (5) KQC/B. Additionally, we are able to evaluate methods that combine any of these algorithms. In Section 3.3, we described a technique to combine methods by merging their result lists in an alternating manner. As a result, the merged list does not contain new documents. Moreover, in the top-10 results of the combined list are only documents listed that are also in the 10 top-ranked results of the individual methods. Since we have judgments for the top-10 results of the five individual algorithms and the top-10 results of combined methods do not contain further documents, we can evaluate the combined methods as well.

In the following section, we describe in detail how we use the obtained relevance judgments to measure the effectiveness of the five individual Related Work Search algorithms and a couple of combined methods.

4.2.2 Performance

In Cranfield-style experiments, the effectiveness of a retrieval system is often evaluated by using the metrics *precision* and *recall* [Voo01]. Precision is the ratio of relevant documents among the retrieved documents, and recall is the ratio of retrieved documents among all relevant documents in the collection. These two metrics are applied on the whole result list and do not consider the order of documents returned by the retrieval system. Since previous research found that users focus on the top-ranked results [JGP+05, ABDR06], we state that the ranking of documents produced by a Related Work Search algorithms has to be considered in the course of the evaluation. Therefore, we apply two other metrics that take the ranking of the result list into account, namely *average precision* and *normalized discounted cumulative gain*. Because of the restrictions in our user study, we only have relevance judgments for the top-10 documents returned by each of the algorithms. For this reason, we will only consider the top-10 results in the course of our ranking-based evaluation. In addition to the ranking-based metrics, we analyze the recall of the result lists with respect to the expected papers that the users have specified in Step 1 of our study and with respect to the unfamiliar relevant papers that we have found by analyzing the judgments obtained in the user study. In the following paragraphs, we outline the calculation of the used evaluation metrics.

Average Precision

The average precision (AP) of an ordered result list up to a rank k is computed by the average of the precision at each rank multiplied with the relevance of the document at this rank:

$$AP@k = \frac{1}{min(k, |relevant\ documents|)} \cdot \sum_{i=1}^{k} Precision@i \cdot relevance(i)\,, \qquad (4.5)$$

where k is the number of top results to be considered. $Precision@i$ is the precision up to rank i, i. e., the ratio of relevant documents among the top-i retrieved documents. Moreover, $relevance(i)$ refers to the relevance of the document at rank i. The sum of the precision values is normalized by the minimum of k and the number of relevant documents that have been found for one topic overall, i. e., the size of the union of relevant documents found by all five algorithms.
The measure AP is designed for the use of binary judgments; that is, a document is

either relevant or not. Since we have obtained relevance judgments for the relatedness of a document on a 4-point scale, we need to transform them into binary judgments. According to the verbal description of the levels of judgments (see Table 4.4), we define highly and fairly related papers to be relevant, and we define marginally and not related documents to be irrelevant. For the purpose of calculation, we assign the numerical value of 1 to relevant documents and the numerical value of 0 to irrelevant documents. We formally state:

$$relevance(judgment) = \begin{cases} 1, & \textit{if} \text{ highly } \textit{or} \text{ fairly} \\ 0, & \textit{if} \text{ marginally } \textit{or} \text{ not related,} \end{cases} \tag{4.6}$$

where *judgment* is the relevance judgment regarding the relatedness of a recommended document.

We first apply the measure of AP@10 to the result list produced on every topic by the five individual algorithms. Additionally, we examine the combination of the best query-based algorithm, which turns out to be KQC/A, with the graph-based method of Sofia Search and the method based on Google Scholar. The resulting AP-values are illustrated in Figure 4.11. The dashed line reports the mean of AP-values per topic; it reveals that the mean AP-values per topic are not similar among the topics but they are scattered. For example, the mean AP of topic 003 is only 0.15, whereas in topic 006 a mean value of 0.93 is reached. Note that we previously attempt to classify a topic's difficulty by the diversity of the respective set of input documents (see Section 4.2.1). According to this classification, topic 006 is an easy topic because its set of input documents has a low diversity. This statement is confirmed by the high mean AP-value that is achieved in topic 006. However, for the other topics the statement cannot be approved. For instance, topic 004 reaches the second best mean AP-value although it was classified as one of the more difficult topics. As a consequence, the diversity of the set of input papers appears not to be a meaningful measure for the difficulty of a topic.
Furthermore, the AP-values of the algorithms per topic illustrated in Figure 4.11 do not show any obvious trend regarding a best performing algorithm. Instead, the algorithms perform different among the topics. For instance, Nascimento reached the best AP in the topics 001, 005, and 009, while it produced the worst AP for the topics 004, 007, and 010. Another example for the varying performance are the AP-values of Google Scholar. For topic 003 it returned clearly the most relevant documents, whereas for the topics 001, 005, 008 and 009 it returned the most irrelevant documents.
As stated previously, we assume that the combination of several methods can outperform its individual components. The combination of KQC/A and Google Scholar, in Figure 4.11 denoted as *KQC/A+Google*, outperforms its parts in 4 topics, namely topic 001, 002, 008, and 010.

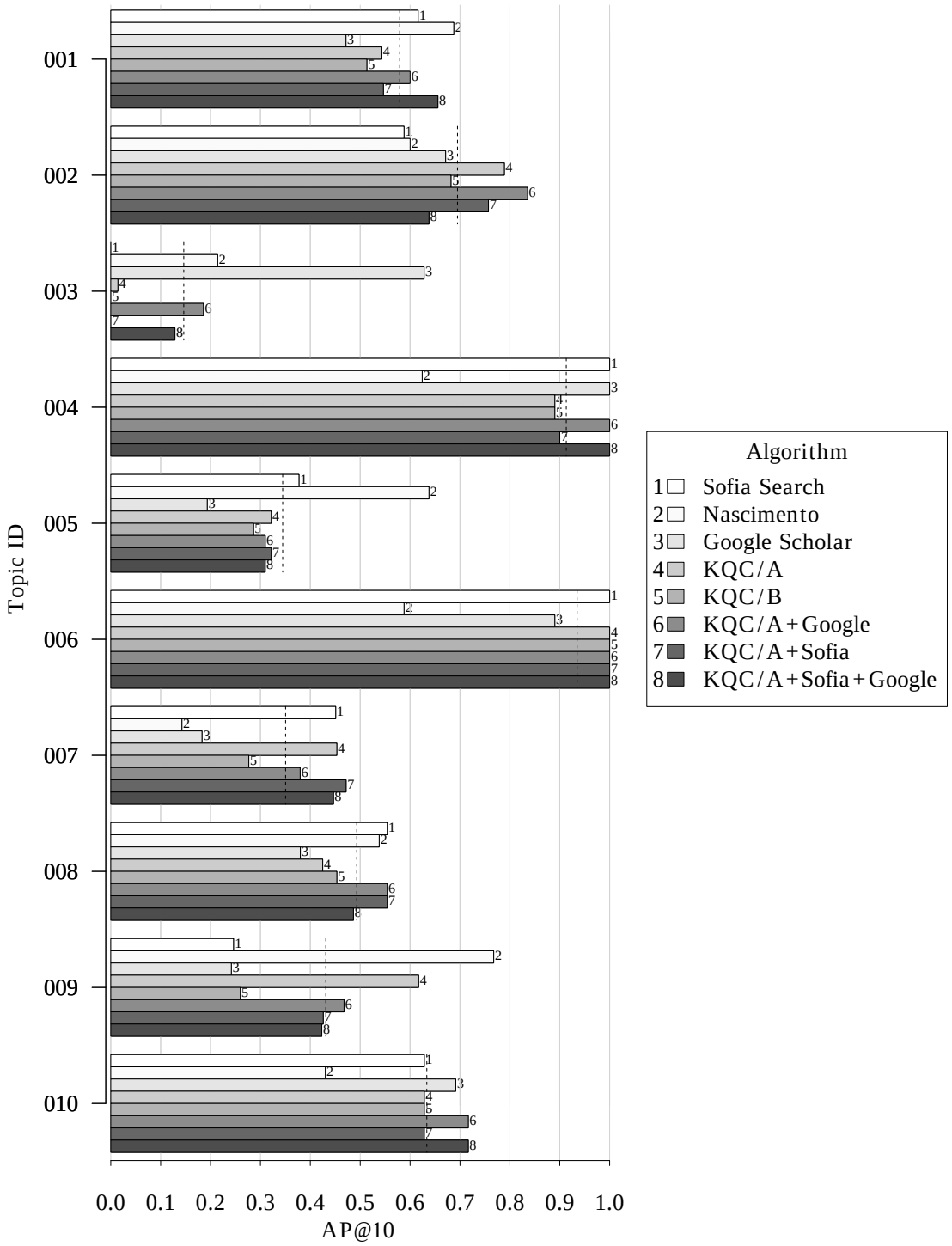

Figure 4.11: Average precision of the top-10 documents returned by each algorithm per topic.

Table 4.6: Mean performance values achieved by the different algorithms over all topics.

Algorithm	AP@10	$nDCG$@10	$Recall_{ED}$@50	$Recall_{URD}$@10
1 Sofia Search	0.546	0.598	0.503	0.371
2 Nascimento	0.523	0.563	0.300	0.344
3 Google Scholar	0.535	0.631	**0.647**	0.261
4 KQC/A	**0.568**	**0.632**	0.460	**0.378**
5 KQC/B	0.499	0.530	0.373	0.354
6 KQC/A+Google	**0.605**	**0.679**	**0.667**	0.353
7 KQC/A+Sofia	0.561	0.621	0.560	**0.369**
8 KQC/A+Sofia+Google	0.580	0.660	0.647	0.317

Moreover, the combination of KQC/A and Sofia Search, which is denoted as *KQC/A+Sofia*, is only once slightly better than its components: KQC/A and Sofia Search both reach AP@10=0.45 in topic 007 where KQC/A+Sofia performs with AP@10=0.47 slightly better. The third combination *KQC/A+Sofia+Google*, which combines KQC/A with Sofia Search and Google Scholar, outperforms its parts in topics 001 and 010. Since it is difficult to determine the best performing algorithm from Figure 4.11, we list the MAP@10 per algorithm over all topics in Table 4.6. Among the five individual algorithms, the approach of KQC/A reaches the highest value with MAP@10=0.568. The second best method with respect to MAP@10 is Sofia Search, and the third best method is Google Scholar. Note that KQC/B produced the worst value with MAP@10=0.499, which indicates that the vocabulary used in KQC/B is less suitable than the vocabulary used in KQC/A. In the bottom rows of Figure 4.11, the combined approaches are listed. The combination KQC/A+Google performs best among the combined methods, and it reaches the highest MAP@10=0.605 in general. Note that the combination KQC/A+Sofia does not outperform its components whereas the last combination of KQC/A+Sofia+Google does.

In order to check the significance of the mean values listed in Table 4.6, we apply a significance test. Therefore, we first need to check whether the samples are normally distributed; a sample comprises the 10 calculated performance values produced by one algorithm for the 10 topics. For this purpose, we apply the Shapiro-Wilk test [RW11] and found that not all of the samples are normally distributed. Consequently, we have to apply a non-parametric test [LFH10]. Since the values are computed on the same sample and we want to do a pairwise comparison, the *Wilcoxon signed rank test* is a suitable significance test for our scenario [LFH10]. More specifically, we perform an one-sided paired Wilcoxon signed rank test on each pair of performance values in order to determine if one algorithm has a significantly higher mean performance value than

another. The p-values of the significance test for the AP@10-values are reported in the appendix in Table A.3. Moreover, we define a significance level of 0.05, which means that the difference in the mean values is significant if $p < 0.05$. It turns out that the approach of KQC/A and all three of the combined methods perform significantly better than KQC/B. For none of the other pairs of values a statistical significance can be found.

To sum up, our keyquery-based method KQC/A performs best among the individual methods with respect to AP@10. The combination of KQC/A and Google Scholar retrieved the most relevant documents among all tested algorithms and thus outperforms its components. Not each of the analyzed combinations performs better than the individual algorithms. It appears that the documents contributed by Sofia Search impair the quality of the ranking since the approaches including Sofia Search perform worse than the respective algorithms without Sofia Search.

Normalized Discounted Cumulative Gain

A further metric that can be applied in a Cranfield-style experiment is normalized discounted cumulative gain (nDCG). The metric of nDCG is based on two basic assumptions [JK02]: (1) documents have different levels of relevance and users prefer documents with higher relevance, and (2) documents that are listed later in the ranking are less likely to be examined by the user.

While the aforementioned metric AP is only capable of binary relevance judgments, nDCG is suitable for non-binary judgments [MRS08]. Therefore, by using nDCG we can exploit the detailedness of the judgments of relatedness obtained in our user study. The nDCG of a sorted result list is computed as follows [JK02]:

$$nDCG@k = \frac{DCG@k}{IDCG@k}\,, \tag{4.7}$$

where $DCG@k$ is the discounted cumulative gain (DCG). Moreover, $IDCG@k$ refers to the ideal DCG, i. e., the DCG produced by the optimal ranking where all obtained judgments are sorted in decreasing order. The DCG@k is defined as:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{relevance(i)} - 1}{\log_2(i+1)}\,, \tag{4.8}$$

where $relevance(i)$ is the relevance of the document at rank i. In order to use the obtained judgments in the calculation of nDCG, we assign the following numerical values:

$$relevance(judgment) = \begin{cases} 3, & \textit{if highly} \\ 2, & \textit{if fairly} \\ 1, & \textit{if marginally} \\ 0, & \textit{if not related.} \end{cases} \tag{4.9}$$

We compute nDCG@10 on the result lists of the five individual algorithms and three combined methods. The IDCG@10 for a topic is calculated over all available judgments for this topic, i. e., the judgments collected for documents in the top-10 results of each algorithm. Figure 4.12 presents the nDCG@10-values per topic and algorithm. First, it should be noted that the distribution of the nDCG-values illustrated in Figure 4.12 looks quite similar to the distribution of the AP-values in Figure 4.11. The mean nDCG-values per topic, which are illustrated by the dashed line, are also scattered, and there is no obvious best performing algorithm for all topics. Yet, there are slightly differences in the two figures. For instance, in topic 004 the ranking of the algorithms differs for the two metrics AP and nDCG. Whereas four of the algorithms, namely Sofia Search, Google Scholar, KQC/A+Google, and KQC/A+Sofia+Google, reach a perfect score of AP@10=1.0, Google Scholar clearly outperforms the other algorithms with respect to nDCG@10. These differences arise partly from the different fineness of the used judgments: for the calculation of AP the judgments were binarized while for the computation of nDCG the 4-point scale judgments were used.
We provide the mean values of nDCG@10 in Table 4.6. The top-10 documents recommended by KQC/A are most relevant among the individual methods with a nDCG-value of 0.632. The second best method is Google Scholar with nDCG@10=0.631 followed by Sofia Search with nDCG@10=0.598. The very best result is reached by the combined method KQC/A+Google. As with the AP-values, we perform a Wilcoxon signed rank test with a significance level of 0.05 in order to examine the significance of the differences in the results. The corresponding p-values are listed in the appendix in Table A.4. We found that the two combinations KQC/A+Google and KQC/A+Sofia+Google perform significantly better than Sofia Search. Moreover, the mean nDCG@10 produced by KQC/A+Sofia+Google is significantly higher than the mean nDCG@10 produced by KQC/A. Last, almost all methods perform significantly better than KQC/B.
Note that the algorithm rankings in Table 4.6 produced by the mean values of AP and nDCG differ. Whereas the ranking produced by AP@10 is $\langle 6, 8, 4, 7, 1, 3, 2, 5 \rangle$ according to the numbers listed in Table 4.6, the ranking produced by the mean nDCG@10 is $\langle 6, 8, 4, 3, 7, 1, 2, 5 \rangle$. These differences are not only caused by the dissimilar calculation but also by the different levels of relevance judgments. As stated above, we categorized the four levels of relatedness into two groups in order to compute the AP-values.

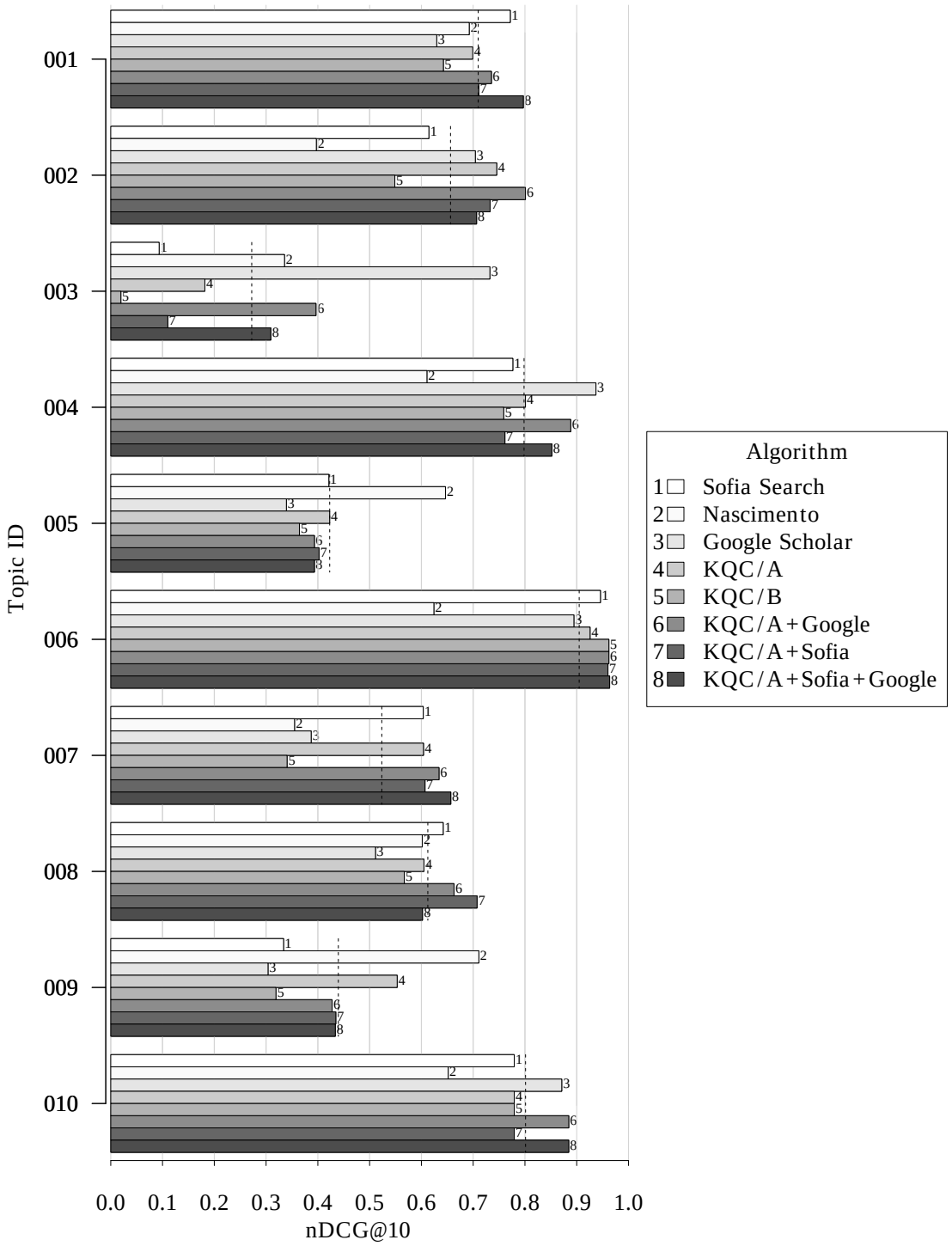

Figure 4.12: Normalized discounted cumulative gain of the top-10 documents returned by each algorithm per topic.

Consequently, some information with respect to the relatedness is lost in the course of the calculation of AP. Therefore, one could state that nDCG@10 is the more meaningful metric for a Cranfield-style experiment with four levels of judgments. Yet, it is to say that the metric of nDCG places focus on the relevance of the documents in the top ranks, which is accomplished by the use of the logarithm function in Equation 4.7. Therefore, we could not state which of the two metrics is more meaningful.
To summarize, the best performing individual methods with respect to nDCG@10 are KQC/A and Google Scholar. The combination of these two methods leads to the highest nDCG@10-value, which reveals that the mixed result list contains more or higher related documents than the individual ones. As with the AP-values, the combined approaches including Sofia Search perform worse than the respective methods without Sofia Search.

Recall with Respect to Expected Papers

The third measure that we use is recall. In contrast to the two aforementioned metrics, it is independent of the ranking of documents in the result list. We utilize the expected documents (ED) specified by the users in the study in order to compute the recall:

$$Recall_{ED}@k = \frac{|ED \cap \textit{top-k retrieved documents}|}{|ED|}. \tag{4.10}$$

Since the computation of the recall is not dependent on the obtained relevance judgments, we can compute recall for any k. However, we choose $k = 50$ because we assume a human would consider the top-50 documents. This means if a participant in the user study has stated two expected documents, and they are listed among the top-50 documents the $Recall_{ED}@k$ will be 1.0. The users have specified a different number of expected documents. For the most topics, or more precisely for the topics 001, 004, 005, 007, 008, 009, and 010, the least required number of two expected documents were entered. The user who stated topic 002 expected three documents to be found, and for the topics 003 and 006 even five expected documents were entered.

Table 4.6 presents the mean $Recall_{ED}@50$-values for each algorithm over all topics. The best $Recall_{ED}@50$ among the individual methods is reached by Google Scholar with a score of 0.647. Moreover, the combination KQC/A+Google reached an even better result with $Recall_{ED}@50$=0.667. One explanation for this can be that the search engine Google Scholar was used previously to find related documents for the specified research task.
We conducted a paired Wilcoxon signed rank test; the respective p-values can be found in the appendix in Table A.5. We found that all three combined methods as well as Google Scholar perform significantly better than Nascimento and KQC/B with a significance level of 0.05. Moreover, the mean $Recall_{ED}@50$ of KQC/A+Sofia+Google is significantly greater than the mean of KQC/A.

Recall with Respect to Unfamiliar Relevant Papers

In our user study, we also asked the participants to state whether they did know a document before the study or not. As stated previously, we use this information to analyze unfamiliar relevant documents (URD), i. e., papers that are highly or fairly related and that the user did not know before.

We want to examine which algorithm returns the most unfamiliar relevant papers. Therefore, we compute the recall with respect to the unfamiliar relevant papers per topic for each algorithm:

$$Recall_{URD}@k = \frac{|URD \cap top{-}k \ retrieved \ documents|}{|URD|}. \tag{4.11}$$

Since we have obtained the unfamiliar relevant papers by examining the judgments for the top-10 documents of the Related Work Search algorithms, we set $k = 10$. The mean $Recall_{URD}$@10-values are listed in Table 4.6. KQC/A has found with, on average, 37.8 % the most unfamiliar relevant documents, and Google Scholar returns the fewest with 26.1 %. This supports the assumption that the users were familiar with the documents found by Google Scholar. We also performed a significance test for the $Recall_{URD}$@10-values. As the p-values in A.6 in the appendix reveal, only the difference between KQC/A+Sofia+Google and Google Scholar is statistically significant.

4.2.3 Discussion

To summarize, we have analyzed eight Related Work Search algorithms: five individual and three combined methods. In our experiments, the novel keyquery-based method KQC/A has produced on average the best top-10 results among the individual algorithms. The results of the combination of KQC/A and Google Scholar contained even more or higher related documents. Besides, the values show that the vocabulary used in KQC/A leads to better results than the vocabulary used in KQC/B.
Moreover, we found that Google Scholar achieves the best recall with respect to the expected papers specified by the participants in the study. This indicates that users are more familiar with the documents found by Google Scholar than by any other of the techniques. The low recall value of Google Scholar with respect to the unfamiliar relevant papers, i. e., documents that are highly or fairly related and that the user did not know before the study, supports this assumption. On the other hand, the keyquery-based method KQC/A has found the most unfamiliar relevant papers that users have not found in previous researches, where they presumably used commercial search engines and graph-based approaches

Note that not all pairwise differences of the listed mean values in Table 4.6 are statistically significant which might be because of the small sample size of 10 topics. Therefore, we cannot assume that all results obtained in our experiments are valid in general. The generally valid statement is that KQC/A and all combined methods outperform KQC/B. In brief, a user study comprising more topics is needed in order to prove the statistical significance of the mean values.
Besides, our evaluation depends strongly on the top-10 documents that the algorithms returned. This restriction is due to the limited number of relevance judgments obtained in our user study. As stated previously, a human might consider more than the 10 top-ranked documents in the course of the search for related work. Therefore, a more meaningful evaluation involves more documents of the result list. As a consequence, more relevance judgments have to be obtained by conducting a more costly user study, which is beyond the scope of this thesis.

In conclusion, our experiments indicate that our novel keyquery-based method KQC/A is comparable to current state-of-the-art methods for finding related research papers (see the first research question stated in Chapter 1). Therefore, the application of the concept of keyqueries appears to be valuable for the task of Related Work Search. Moreover, the results of our experiments indicate that the combination of keyquery-based and graph-based methods does not outperform its components (see the second research question stated in Chapter 1). Since we could not prove statistical significance in the differences of the obtained mean values, our findings have to be confirmed in a large-scale evaluation using more topics and considering more documents in the list of recommended papers.

5 Conclusion

In this thesis, we investigated methods for the recommendation of related work. We specifically address the use case in which the user has already found a set of interesting documents in the course of an initial research and wants to find papers that are related to them. Accordingly, we formally stated the problem of Related Work Search: Given a set of input documents, the task is to find a set of related papers. Basically, this problem can be divided into two steps: (1) candidate retrieval, and (2) candidate ranking. In the first step, candidate documents are retrieved, which are ranked in the second step with respect to their relevance. We focused on the candidate retrieval as a first crucial step in the process of finding related work. More specifically, we analyzed the applicability of the concept of keyqueries proposed by Gollub et al. [GHMS13] for this step. A keyquery for a document is a query that returns the document in the top-ranked results when it is submitted to a reference search engine. The other documents that are also listed on top ranks cover topics similar to the paper itself and thus can be used as candidate documents for the search of related work.

We proposed a novel keyquery-based method for the recommendation of related work. In this approach, the vocabulary extracted from a set of input documents is utilized to formulate keyqueries. Then, the other results returned by the keyqueries are used as a set of candidate documents. Subsequently, the candidate set is ranked based on the text similarity of a candidate document to the specified set of input papers. Moreover, we investigated the suitability of two different vocabularies used for the formulation of keyqueries. This led to two versions of our keyquery-based approaches, namely KQC/A and KQC/B. The vocabulary utilized in KQC/A contains keyphrases extracted from the set of input papers. In KQC/B the vocabulary also comprises extracted keywords but they are restricted to keywords that often appear in the collection as a paper's keyword specified by the paper's author. That is, the vocabularies used in KQC/A and KQC/B are not completely distinct but overlap in several keyphrases.

In the course of our evaluation, we implemented three state-of-the-art algorithms as baselines: (1) a version of the graph-based approach by Golshan et al. [GLT12], (2) a simple query-based method by Nascimento et al. [NLSG11], and (3) an approach based on the scholarly search engine Google Scholar. These methods represent the ways of how humans would search for candidate documents for a given set input documents: following the citation graph, formulating queries from given texts, and using the related articles feature of Google Scholar.

In order to measure the effectiveness of our novel approach in comparison to the baseline algorithms, we conducted a Cranfield-style experiment. Therefore, we have used an index of a large collection, namely the WCSP, that contains about 187,000 Computer Science research papers. This document collection formed the first component of our experiment. Since the other two components, topics and corresponding relevance judgments, were not available for the WCSP, we conducted a user study in order to obtain them.
Our user study has involved 10 experts in Computer Science. Each of them stated a topic containing at least two input documents in the first step of the study; as a result, we had a set of 10 topics. Then, we have applied our two keyquery-based algorithms and the baselines on the topics and obtained recommended papers. In the second step of the study, we asked the users to judge the top-10 documents recommended by the algorithms according to their relevance to the input papers.
Based on the obtained relevance judgments, we evaluated the ranking of related documents produced by the algorithms. The results of the experiments show that KQC/A produced a significantly better ranking than KQC/B; consequently, the vocabulary used in KQC/A is more valuable for keyquery-based recommendation of related work than the vocabulary used in KQC/B. Moreover, the results indicate that our novel keyquery-based method KQC/A outperforms current state-of-the-art methods for finding related research papers. Furthermore, the combination of our keyquery-based method KQC/A and the graph-based method of Sofia Search did not outperform its components in our experiments. Since we could not prove the statistical significance of the last two statements, our findings have to be confirmed in a large-scale evaluation using more topics and considering more documents in the list of recommended papers.

Moreover, the results of the experiments should be confirmed by using another data collection. For this purpose, the iSearch collection [LLLI10] might be suitable since it comprises a large set of research papers from the domain of physics and a set of topics with corresponding relevance judgments. Yet, the iSearch collection does not contain information about cited and citing papers of a document. Consequently, citation information has to be extracted first in order to apply graph-based methods on the iSearch collection.
In this thesis, we proposed a keyquery-based method along with two different strategies for the extraction of a document's vocabulary. Future research could examine more algorithms using keyqueries. For example, another approach could be to formulate keyqueries involving not only the vocabulary extracted from the text but also the names of a paper's authors.
Furthermore, keyqueries depend strongly on the retrieval model of the reference search engine. Therefore, one future research question could be: Which is the best retrieval model of the reference search engine in order to find candidate papers in the result lists of documents' keyqueries.

In our investigations, we have focused on the retrieval of candidate documents, and we have only applied simple ranking strategies. Therefore, in future research more convincing features for ranking should be investigated. For instance, the features used for candidate ranking of the original version of Sofia Search (see Section 3.1.2) seem to be valuable. Additionally, the quality and impact of a paper's publication venue could be considered for the ranking of candidate papers. A further feature could examine how often a paper is found as a candidate document by different methods. For instance, a paper that is found by a graph-based and a keyquery-based approach might be more valuable than a paper that is found by only one of the algorithms.
Note that the work at hand has only evaluated the effectiveness of keyquery-based approaches in comparison to state-of-the-art methods in order to analyze the applicability of keyqueries for the task of Related Work Search. Future research should also investigate the efficiency of the different algorithms.

A Appendix

Seed Documents of the WCSP

Table A.1 lists a few characteristics of the 34,280 seed documents of the WCSP, which built the basis for the retrieval of 152,517 further documents. The seed documents were in the years 1967 – 2012 in the proceedings of 20 different publications.

Table A.1: Characteristics of WCSP seed documents.

Acronym	Years		$\|D\|$
	From	To	
ACL	1979	2012	2,061
CHI	1981	2012	4,729
CIKM	1993	2012	3,018
FSE	2010	2012	104
ICML	2004	2009	844
ICSE	1976	2011	2,899
KDD	1999	2012	1,603
OSDI	2004	2006	28
PLDI	1988	2012	807
PODS	1982	2012	971
SIGCOMM	1977	2012	1,154
SIGIR	1971	2012	2,830
SIGMETRICS	1976	2012	1,217
SIGMOD	1975	2012	2,562
SODA	1990	2012	2,444
SOSP	1967	2011	536
STOC	1969	2012	2,619
UIST	1988	2012	890
VLDB	1975	2007	706
WWW	2001	2012	2,258
			34,280

User Study Topics

Table A.2 lists the 10 topics obtained in the course of our user study. The column ID refers to the topic ID that we have assigned. In the second column, the specified research task is listed in boldface, and the respective input documents are listed by title.

Table A.2: The 10 topics obtained in our user study.

ID	Research Task and Input Documents
001	**Off-screen pointer techniques**
	• Halo: a technique for visualizing off-screen objects
	• Wedge: clutter-free visualization of off-screen locations
002	**Curse of dimensionality**
	• On the effects of dimensionality reduction on high dimensional similarity search
	• The concentration of fractional distances
003	**Automatic summarization of perspectives**
	• Staying informed: supervised and semi-supervised multi-view topical analysis of ideological perspective
	• Summarizing contrastive viewpoints in opinionated text
004	**Efficient linked open data processing**
	• Hexastore: sextuple indexing for semantic web data management
	• Binary RDF for scalable publishing, exchanging and consumption in the web of data
005	**External hashing**
	• Dynamic external hashing: the limit of buffering
	• Linear hashing with separators - a dynamic hashing scheme achieving one-access
	• External memory algorithms and data structures: dealing with massive data
006	**Search result clustering**
	• Scatter/Gather: a cluster-based approach to browsing large document collections
	• Reexamining the cluster hypothesis: scatter/gather on retrieval results
	• Web document clustering: a feasibility demonstration
007	**Document clustering and labeling**
	• Enhancing cluster labeling using wikipedia
	• Topical clustering of search results
008	**Research paper recommendation**
	• A source independent framework for research paper recommendation
	• SOFIA SEARCH: a tool for automating related-work search
009	**Exploratory search**
	• Model-driven formative evaluation of exploratory search: a study under a sensemaking framework
	• Exploratory search: from finding to understanding
010	**Query segmentation**
	• Towards optimum query segmentation: in doubt without
	• Query segmentation based on eigenspace similarity
	• Unsupervised query segmentation using click data: preliminary results

Significance Tests

Table A.3 – Table A.6 list the p-values that we have obtained in the course of the significance testing of the performance values reported in Table 4.6 in Section 4.2.2. We have conducted a Wilcoxon signed rank test for each pair of performance values produced by the algorithms over all topics. More specifically, we tested whether the mean value produced by the algorithm listed in the row is significantly greater than the mean value produced by the algorithm listed in the column. Therefore, the matrices are not symmetric. Moreover, we have utilized a significance level of 0.05; that is, the difference of a pair of mean values is statistically significant if $p < 0.05$. In Table A.3 – Table A.6, the p-values that are smaller than 0.05 are printed in bold.

Table A.3: The p-values obtained in the course of significance testing of the mean AP@10-values reported in Table 4.6.

	Sofia Search	Nascimento	Google Scholar	KQC/A	KQC/B	KQC/A+Google	KQC/A+Sofia	KQC/A+Sofia+Google
Sofia Search	1.000	0.423	0.172	0.528	0.054	0.883	0.663	0.925
Nascimento	0.615	1.000	0.577	0.754	0.423	0.688	0.688	0.784
Google Scholar	0.857	0.461	1.000	0.880	0.713	0.922	0.903	0.951
KQC/A	0.528	0.278	0.142	1.000	**0.026**	0.639	0.466	0.882
KQC/B	0.962	0.615	0.323	0.983	1.000	0.991	0.993	0.997
KQC/A+Google	0.147	0.348	0.096	0.406	**0.012**	1.000	0.318	0.853
KQC/A+Sofia	0.417	0.348	0.116	0.600	**0.011**	0.723	1.000	0.946
KQC/A+Sofia+Google	0.102	0.246	0.062	0.143	**0.005**	0.201	0.071	1.000

Table A.4: The p-values obtained in the course of significance testing of the mean $nDCG$@10-values reported in Table 4.6.

	Sofia Search	Nascimento	Google Scholar	KQC/A	KQC/B	KQC/A+Google	KQC/A+Sofia	KQC/A+Sofia+Google
Sofia Search	1.000	0.348	0.500	0.857	**0.009**	0.986	0.857	0.981
Nascimento	0.688	1.000	0.722	0.884	0.385	0.903	0.812	0.935
Google Scholar	0.539	0.312	1.000	0.722	0.161	0.903	0.722	0.947
KQC/A	0.172	0.138	0.312	1.000	**0.006**	0.884	0.239	0.968
KQC/B	0.994	0.652	0.862	0.995	1.000	1.000	0.995	0.997
KQC/A+Google	**0.019**	0.116	0.116	0.138	**0.001**	1.000	0.161	0.819
KQC/A+Sofia	0.172	0.216	0.312	0.797	**0.006**	0.862	1.000	0.958
KQC/A+Sofia+Google	**0.024**	0.080	0.065	**0.042**	**0.005**	0.221	0.053	1.000

Table A.5: The p-values obtained in the course of significance testing of the mean $Recall_{ED}$@50-values reported in Table 4.6.

	Sofia Search	Nascimento	Google Scholar	KQC/A	KQC/B	KQC/A+Google	KQC/A+Sofia	KQC/A+Sofia+Google
Sofia Search	1.000	0.068	0.873	0.392	0.093	0.978	0.909	0.931
Nascimento	0.952	1.000	0.990	0.920	0.727	0.994	0.972	0.990
Google Scholar	0.167	**0.018**	1.000	0.080	**0.039**	0.585	0.259	0.707
KQC/A	0.709	0.118	0.947	1.000	0.179	0.947	0.970	0.957
KQC/B	0.956	0.333	0.972	0.901	1.000	0.994	0.974	0.994
KQC/A+Google	0.050	**0.010**	0.500	0.080	**0.010**	1.000	0.231	0.707
KQC/A+Sofia	0.211	**0.039**	0.806	0.173	**0.042**	0.865	1.000	0.828
KQC/A+Sofia+Google	0.102	**0.018**	0.500	**0.061**	**0.010**	0.500	0.219	1.000

Table A.6: The p-values obtained in the course of significance testing of the mean $Recall_{URD}$@10-values reported in Table 4.6.

	Sofia Search	Nascimento	Google Scholar	KQC/A	KQC/B	KQC/A+Google	KQC/A+Sofia	KQC/A+Sofia+Google
Sofia Search	1.000	0.722	0.080	0.705	0.209	0.708	0.583	0.664
Nascimento	0.312	1.000	0.089	0.466	0.277	0.277	0.466	0.542
Google Scholar	0.935	0.947	1.000	0.946	0.908	0.929	0.896	0.962
KQC/A	0.394	0.600	0.071	1.000	0.140	0.624	0.292	0.824
KQC/B	0.860	0.761	0.117	0.911	1.000	0.819	0.735	0.883
KQC/A+Google	0.428	0.777	0.092	0.458	0.221	1.000	0.265	0.791
KQC/A+Sofia	0.500	0.600	0.147	0.819	0.337	0.799	1.000	0.819
KQC/A+Sofia+Google	0.400	0.542	**0.049**	0.223	0.147	0.295	0.221	1.000

Bibliography

[ABDR06] Eugene Agichtein, Eric Brill, Susan T. Dumais, and Robert Ragno. Learning user interaction models for predicting web search result preferences. In Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, and Kalervo Järvelin, editors, *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 3–10. ACM, 2006.

[AGS11] Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*. ACM, 2011.

[AJM04] Yuan An, Jeannette Janssen, and Evangelos E. Milios. Characterizing and mining the citation graph of the computer science literature. *Knowl. Inf. Syst.*, 6(6):664–678, 2004.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[BC09] Michael Bendersky and W. Bruce Croft. Finding text reuse on the web. In Baeza-Yates et al. [BYBRNC09], pages 262–271.

[BDD+08] Steven Bird, Robert Dale, Bonnie J. Dorr, Bryan R. Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R. Radev, and Yee Fan Tan. The ACL anthology reference corpus: a reference dataset for bibliographic research in computational linguistics. In *LREC*. European Language Resources Association, 2008.

[Bel62] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962.

[BJ10] Steven Bethard and Dan Jurafsky. Who should I cite: learning literature search models from citation behavior. In Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors, *CIKM*, pages 609–618. ACM, 2010.

[BLG+13] Joeran Beel, Stefan Langer, Marcel Genzmehr, Bela Gipp, Corinna Breitinger, and Andreas Nürnberger. Research paper recommender system evaluation: a quantitative literature survey. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, pages 15–22. ACM, 2013.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[BSAS95] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using SMART: TREC 3. *NIST special publication sp*, pages 69–69, 1995.

[BYBRNC09] Ricardo A. Baeza-Yates, Paolo Boldi, Berthier A. Ribeiro-Neto, and Berkant Barla Cambazoglu, editors. *Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009*. ACM, 2009.

[CSMG13] Cornelia Caragea, Adrian Silvescu, Prasenjit Mitra, and C. Lee Giles. Can't see the forest for the trees? A citation recommendation system. In Downie et al. [DMC+13], pages 111–114.

[DDKD09] Ali Dasdan, Paolo D'Alberto, Santanu Kolay, and Chris Drome. Automatic retrieval of similar content using search engine query interface. In David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin, editors, *CIKM*, pages 701–710. ACM, 2009.

[Del11] Scott E. Delman. ACM Aggregates Publication Statistics in the ACM Digital Library. *Communications of the ACM*, 54(7):12–12, 2011.

[DMC+13] J. Stephen Downie, Robert H. McDonald, Timothy W. Cole, Robert Sanderson, and Frank Shipman, editors. *13th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '13, Indianapolis, IN, USA - July 22 - 26, 2013*. ACM, 2013.

[EG11] Khalid El-Arini and Carlos Guestrin. Beyond keyword search: discovering relevant scientific literature. In Apté et al. [AGS11], pages 439–447.

[EKS+10] Michael D. Ekstrand, Praveen Kannan, James A. Stemper, John T. Butler, Joseph A. Konstan, and John Riedl. Automatically building research reading lists. In Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker, editors, *RecSys*, pages 159–166. ACM, 2010.

[ER09] Samhaa R. El-Beltagy and Ahmed A. Rafea. KP-Miner: a keyphrase extraction system for English and Arabic documents. *Inf. Syst.*, 34(1):132–144, 2009.

[GHMS13] Tim Gollub, Matthias Hagen, Maximilian Michel, and Benno Stein. From keywords to keyqueries: content descriptors for the web. In Gareth J. F. Jones, Paraic Sheridan, Diane Kelly, Maarten de Rijke, and Tetsuya Sakai, editors, *Proceedings of the 36th International ACM SIGIR conference on research and development in Information Retrieval*, pages 981–984. ACM, 2013.

[GLT12] Behzad Golshan, Theodoros Lappas, and Evimaria Terzi. SOFIA SEARCH: a tool for automating related-work search. In K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman, editors, *SIGMOD Conference*, pages 621–624. ACM, 2012.

[HKC+12] Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C. Lee Giles, and Lior Rokach. Recommending citations: translating papers into references. In wen Chen et al. [wCLWZ12], pages 1910–1914.

[HKP+11] Qi He, Daniel Kifer, Jian Pei, Prasenjit Mitra, and C. Lee Giles. Citation recommendation without author supervision. In Irwin King, Wolfgang Nejdl, and Hang Li, editors, *WSDM*, pages 755–764. ACM, 2011.

[HPBS12] Matthias Hagen, Martin Potthast, Anna Beyer, and Benno Stein. Towards optimum query segmentation: in doubt without. In wen Chen et al. [wCLWZ12], pages 1015–1024.

[HPK+10] Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and C. Lee Giles. Context-aware citation recommendation. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *WWW*, pages 421–430. ACM, 2010.

[HS11] Matthias Hagen and Benno Stein. Candidate document retrieval for web-scale text reuse detection. In Roberto Grossi, Fabrizio Sebastiani, and Fabrizio Silvestri, editors, *SPIRE*, volume 7024 of *Lecture Notes in Computer Science*, pages 356–367. Springer, 2011.

[JGP+05] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 154–161. ACM, 2005.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[KMB10] Saurabh Kataria, Prasenjit Mitra, and Sumit Bhatia. Utilizing context in generative bayesian models for linked corpus. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[KMKB10] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Semeval-2010 task 5: automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics, 2010.

[KSKC13] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V. Catalyürek. Theadvisor: a webservice for academic recommendation. In Downie et al. [DMC+13], pages 433–434.

[LC12] Chia-Jung Lee and W. Bruce Croft. Generating queries from user-selected text. In *Proceedings of the 4th Information Interaction in Context Symposium*, IIIX '12, pages 100–109, New York, NY, USA, 2012. ACM.

[LFH10] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research methods in human-computer interaction*. Wiley Publishing, 2010.

[LGT+14] Avishay Livne, Vivek Gokuladas, Jaime Teevan, Susan Dumais, and Eytan Adar. CiteSight. In *SIGIR*, 2014.

[LHSY11] Yang Lu, Jing He, Dongdong Shan, and Hongfei Yan. Recommending citations with translation model. In Craig Macdonald, Iadh Ounis, and Ian Ruthven, editors, *CIKM*, pages 2017–2020. ACM, 2011.

[LLLI10] Marianne Lykke, Birger Larsen, Haakon Lund, and Peter Ingwersen. Developing a test collection for the evaluation of integrated search. In *Advances in Information Retrieval*, pages 627–630. Springer, 2010.

[LM00] Ronny Lempel and Shlomo Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks*, 33(1-6):387–401, 2000.

[LYZ13] Yingming Li, Ming Yang, and Zhongfei (Mark) Zhang. Scientific articles recommendation. In Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi, editors, *CIKM*, pages 1147–1156. ACM, 2013.

[Mer68] Robert K Merton. The matthew effect in science. *Science*, 159(3810):56–63, 1968.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[NAXC08] Ramesh Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. Joint latent topic models for text and citations. In Ying Li, Bing Liu, and Sunita Sarawagi, editors, *KDD*, pages 542–550. ACM, 2008.

[NLSG11] Cristiano Nascimento, Alberto H. F. Laender, Altigran Soares da Silva, and Marcos André Gonçalves. A source independent framework for research paper recommendation. In Glen Newton, Michael J. Wright, and Lillian N. Cassel, editors, *JCDL*, pages 297–306. ACM, 2011.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: bringing order to the web. 1999.

[Por80] Martin F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[PPFF09] Joaquín Pérez-Iglesias, José R. Pérez-Agüera, Víctor Fresno, and Yuval Z. Feinstein. Integrating the probabilistic models BM25/BM25F into Lucene. *CoRR*, abs/0911.5046, 2009.

[RRT08] Anna Ritchie, Stephen Robertson, and Simone Teufel. Comparing citation contexts for information retrieval. In James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury, editors, *CIKM*, pages 213–222. ACM, 2008.

[RW11] Nornadiah Mohd Razali and Yap Bee Wah. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, 2(1):21–33, 2011.

[RZT04] Stephen E. Robertson, Hugo Zaragoza, and Michael J. Taylor. Simple BM25 extension to multiple weighted fields. In David A. Grossman, Luis Gravano, ChengXiang Zhai, Otthein Herzog, and David A. Evans, editors, *CIKM*, pages 42–49. ACM, 2004.

[SH10] Benno Stein and Matthias Hagen. Making the most of a web search session. In Jimmy Xiangji Huang, Irwin King, Vijay V. Raghavan, and Stefan Rueger, editors, *Web Intelligence*, pages 90–97. IEEE Computer Society, 2010.

[SH11] Benno Stein and Matthias Hagen. Introducing the User-over-Ranking hypothesis. In Paul Clough, Colum Foley, Cathal Gurrin, Gareth J. F. Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Murdock, editors, *ECIR*, volume 6611 of *Lecture Notes in Computer Science*, pages 503–509. Springer, 2011.

[SK13] Kazunari Sugiyama and Min-Yen Kan. Exploiting potential citation papers in scholarly paper recommendation. In Downie et al. [DMC+13], pages 153–162.

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining.* Addison-Wesley, 2005.

[TWZ14] Xuewei Tang, Xiaojun Wan, and Xun Zhang. Cross-language context-aware citation recommendation in scientific articles. In Shlomo Geva, Andrew Trotman, Peter Bruza, Charles L. A. Clarke, and Kalervo Järvelin, editors, *SIGIR*, pages 817–826. ACM, 2014.

[TZ09] Jie Tang and Jing Zhang. A discriminative approach to topic-based citation recommendation. In Thanaruk Theeramunkong, Boonserm Kijsirikul, Nick Cercone, and Tu Bao Ho, editors, *PAKDD*, volume 5476 of *Lecture Notes in Computer Science*, pages 572–579. Springer, 2009.

[Vit06] Jeffrey Scott Vitter. Algorithms and data structures for external memory. *Foundations and Trends in Theoretical Computer Science*, 2(4):305–474, 2006.

[Voo01] Ellen M. Voorhees. The philosophy of information retrieval evaluation. In Carol Peters, Martin Braschler, Julio Gonzalo, and Michael Kluck, editors, *CLEF*, volume 2406 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2001.

[VSK14] Suzan Verberne, Maya Sappelli, and Wessel Kraaij. Query term suggestion in academic search. In Maarten de Rijke, Tom Kenter, Arjen P. de Vries, ChengXiang Zhai, Franciska de Jong, Kira Radinsky, and Katja Hofmann, editors, *ECIR*, volume 8416 of *Lecture Notes in Computer Science*, pages 560–566. Springer, 2014.

[WB11] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In Apté et al. [AGS11], pages 448–456.

[wCLWZ12] Xue wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki, editors. *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*. ACM, 2012.

[YBD+09] Yin Yang, Nilesh Bansal, Wisam Dakka, Panagiotis G. Ipeirotis, Nick Koudas, and Dimitris Papadias. Query by document. In Baeza-Yates et al. [BYBRNC09], pages 34–43.

[ZTLV13] Xiaodan Zhu, Peter Turney, Daniel Lemire, and Andre Vellino. Measuring academic influence: not all citations are equal. In *Journal of the American Society for Information Science and Technology*, 2013.