Supporting Scholarly Search with Keyqueries

Matthias Hagen(✉), Anna Beyer, Tim Gollub, Kristof Komlossy, and Benno Stein

Bauhaus-Universität Weimar, Weimar, Germany
{matthias.hagen,anna.beyer,tim.gollub,kristof.komlossy,
benno.stein}@uni-weimar.de

Abstract. We deal with a problem faced by scholars every day: identifying relevant papers on a given topic. In particular, we focus on the scenario where a scholar can come up with a few papers (e.g., suggested by a colleague) and then wants to find "all" the other related publications. Our proposed approach to the problem is based on the concept of keyqueries: formulating keyqueries from the input papers and suggesting the top results as candidates of related work.

We compare our approach to three baselines that also represent the different ways of how humans search for related work: (1) a citation-graph-based approach focusing on cited and citing papers, (2) a method formulating queries from the paper abstracts, and (3) the "related articles"-functionality of Google Scholar. The effectiveness is measured in a Cranfield-style user study on a corpus of 200,000 papers. The results indicate that our novel keyquery-based approach is on a par with the strong citation and Google Scholar baselines but with substantially different results—a combination of the different approaches yields the best results.

1 Introduction

We tackle the problem of automatically supporting a scholar's search for related work. Given a research task, the term "related work" refers to papers on similar topics. Scholars collect and analyze related work in order to get a better understanding of their research problem and already existing approaches; a survey of the strengths and weaknesses of related work forms the basis for placing new ideas into context. In this paper, we show how the concept of keyqueries [9] can be employed to support search for related work.

Search engines like Google Scholar, Semantic Scholar, Microsoft Academic Search, or CiteSeerX provide a keyword-based access to their paper collections. However, since researchers usually have limited knowledge when they start to investigate a new topic, it is difficult to find all the related papers with one or two queries against such interfaces. Keyword queries help to identify a few promising initial papers, but to find further papers, researchers usually bootstrap their search from information in these initial papers.

Every paper provides two types of information useful for finding related work: content and metadata. Content (title, abstract, body text) is a good resource

N. Ferro et al. (Eds.): ECIR 2016, LNCS 9626, pp. 507–520, 2016.
DOI: 10.1007/978-3-319-30671-1_37

for search keywords. Metadata (bibliographic records, references) can be used to follow links to referenced papers. Recursively exploring the literature via queries or citations to and from some initial papers is common practice, although rather time-consuming. Support is provided by methods that automate the above procedure: graph-based methods exploit the citation network and content-based methods can generate queries.

Recently, Gollub et al. proposed the concept of keyqueries [9]: A *keyquery* for a document is a query that returns the document in the top result ranks against a (reference) search engine. Assuming that the top results returned by a document's keyqueries cover similar topics, we view the concept of keyqueries as promising for identifying related papers. Moreover, we believe that keyqueries can identify papers that graph-based methods might miss since the citation graph tends to be noisy and sparse [4]. Based on these assumptions, we address the research questions of whether keyqueries are useful for identifying related work and whether they complement other standard approaches.

Our contributions are threefold: (1) We develop a keyquery-based method identifying related work. (2) For the evaluation, we implement three strong baselines representing standard approaches: the graph-based Sofia Search by Golshan et al. [10], the query-based method by Nascimento et al. [24], and the "related articles"-feature of Google Scholar. (3) We conduct a Cranfield-style user study to compare the different approaches.

2 Related Work

Methods for identifying related work can be divided into citation-graph-based and content-based approaches. Only few of the content-based methods use queries such that we also investigate query formulation techniques for similar tasks.

2.1 Identifying Related Research Papers

Many variants of related work search are known: literature search, citation recommendation, research paper recommendation, etc. Some try to find references for a written text, others predict further "necessary" references given a subset of a paper's references. In our setting, the task is to find related papers according to a given set of papers. This represents the everyday use case of enlarging an initial related work research.

Related Work Search
Given: An input list $\mathcal{I} = \langle d_1, d_2, \ldots, d_n \rangle$ of papers
Task: Find an output list $\mathcal{O} = \langle d'_1, d'_2, \ldots, d'_m \rangle$ of related papers.

Given the initial knowledge of a scholar specified as the list $\mathcal{I}$ of input papers (could also be just one), most approaches to Related Work Search retrieve *candidate papers* in a first step. In a second step, the candidates are ranked to generate the final output list $\mathcal{O}$. More than 80 approaches are known for the problem of identifying related papers [1]. We concentrate on the recent and better performing approaches and classify them by the employed candidate retrieval method: *content* and/or *citations* can be used.

Citation-Graph-Based Methods. The network of citations forms the *citation graph*. If d cites d', we call d a *citing paper* of d' and d' a *cited paper* of d. Several approaches apply collaborative filtering (CF) using the adjacency matrix of the citation graph as the "rating" matrix [4,28]. A limitation of CF is that it has problems for poorly connected papers, known as the "cold-start-problem" [31]. Ekstrand et al. thus explore the additional application of link ranking algorithms like PageRank [6] that can also be applied stand-alone [19]. Since the citation graph tends to be noisy and sparse [4], by design, graph-based approaches favor frequently cited papers [31]. Methods that only use the citation graph easily miss papers that are rarely cited (e.g., very recent ones).

As a graph-based baseline, we select Sofia Search [10]. It very closely mimics the way how humans would identify candidates from the citation graph. Starting from an initial set of papers, the approach follows all links to cited and citing papers up to a given recursion depth or until a desired number of candidates is found. Note that this procedure conforms with CF methods and link ranking to some extent. Papers citing similar papers are linked via the cited papers, such that the CF candidates are included. Setting the recursion depth large enough, also all interesting results from PageRank random walk paths will be found—except the low-probability "clicks" on some random non-linked papers. Thus, Sofia Search forms a good representative of graph-based approaches.

Content-Based Techniques. Content-based approaches utilize the paper texts to find related work. Translation models are used to compute the probability of citing a paper based on citation contexts [15], the content of potential references [21], or via an embedding model [30]. Similar ideas are based on topic models: LDA was combined with PLSA to build a topic model from texts and citations [23]. Later improvements use only citation contexts instead of full texts [17,29]. Drawbacks of translation- and topic-model-based approaches are the long training phase and that re-training is necessary whenever papers dealing with new topics are added to the collection. Besides such efficiency aspects, topic and translation models do not resemble human behavior, and they cannot be used with the keyword query interfaces of existing scholarly search engines. We thus choose a query-based baseline to represent the content-based approaches.

While there are complete retrieval models for recommending papers for a given abstract using metadata and content-based features [3], we prefer a standard keyword query baseline since it can be used against any scholarly search interface. Nascimento et al. [24] propose such a method: given a paper, ten word bigrams are extracted from the title and the abstract and submitted as separate queries. Note that queries containing only two words are very general and return a large number of results. Still, Nascimento et al.'s idea is close to human behavior and forms a good content-based baseline.

Combined Approaches. Several methods combine citations and content. For instance, by querying with sentences from a given paper and then following references in the search results [13,14]. However, submitting complete sentences leads to very specific queries returning very few results only; a drawback

we will avoid in our query formulation. Other combined approaches use topic models for citation weighting [7] or overcoming the cold-start-problem of papers without ratings in online reference management communities like CiteULike or Mendeley [31]. The CiteSight system [20] is supposed to recommend references while writing a manuscript using both graph- and content-based features to recommend papers the author cited in the past, or cited papers from references the author already added. Since our use case is different and considering only cited papers appears very restrictive, we do not employ this approach either.

As a representative of the combined approaches, we use Google Scholar's feature "related articles" to form an often used and very strong baseline. Even though the underlying algorithms are proprietary, it is very reasonable that content-based features (e.g., text similarity) and citation-based features (e.g., number of citations) are combined.

2.2 Query Formulation

Since the existing query techniques for related work search are rather simplistic, we briefly review querying strategies for other problems that inspired our approach.

There are several query-by-document approaches that derive "fingerprint" queries for a document in near-duplicate, text reuse, or similarity detection [2,5,32]. Hagen and Stein [12] further improve these query formulation strategies trying to satisfy a so-called *covering property* and the User-over-Ranking hypothesis [27]. Recently, Gollub et al. also introduced the concept of keyqueries for describing a document's content [9]. 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. Instead of just representing a paper by its keyqueries (as suggested by Gollub et al.), we further generalize the idea and assume that the other top-ranked results returned by a paper's keyqueries are highly related to the paper. We will adjust the keyquery formulation to our case of potentially more than one input paper and combine it with the covering property of Hagen and Stein [12] to derive a keyquery cover for the input papers.

3 Baselines and Our Approach

After describing three baselines, we introduce our novel keyquery-based approach and a straightforward interleaving scheme for combining the results of different methods.

3.1 Baselines

The baselines are chosen from the literature to mimic the strategies scholars employ: formulating queries, following citations, and Google Scholar's "related articles."

As a representative of the content-based strategies, we select a method by Nascimento et al. [24]. The approach submits as its ten queries the ten distinct consecutive bigrams of non-stopword terms from a paper's title and abstract that have the highest normalized tf-weights. For each query, the top-50 results are stored. The combined candidate set is then ranked according to the tf-weighted cosine similarity to the input paper. Note that the candidate retrieval and ranking are designed for only one input paper. We adapt the process to our use case by applying the retrieval phase for every input paper individually and then ranking the combined result sets of all input papers by the highest similarity to any of the input papers. The reason for taking the highest similarity and not the average is the better performance in pilot experiments. We also tried to combine the titles and abstracts to a single meta-paper in case of more than one input paper. Taking the highest similarity for individual input papers showed the best performance.

As a representative of the citation graph approaches, we select Sofia Search [10]. For each input paper d, both cited and citing papers are added to a candidate set $\mathcal{C}$. This routine is iterated using the candidates as new starting points until either enough (or no more) candidates are found or a specified recursion depth is reached (typically 2 or 3). Again, we rank the candidates by their highest similarity to any of the input papers.

Our third baseline is formed by Google Scholar's "related articles" feature. For a given research paper, a link in the Google Scholar interface yields a list of about 100 related articles. We collect all these related articles for each input paper individually, treating the underlying retrieval model as a black box. Since Google Scholar already presents ranked results, we do not re-rank them. In case of more than one input paper, we use a simple interleaving strategy: first the first rank for the first input paper, then the first rank for the second input paper, then the first rank for the third input paper, etc., then the second rank for the first input paper, etc. In case that a ranked result is already contained in the merged list it is not considered again.

3.2 New Keyquery-Based Approach

Our new approach combines the concepts of keyqueries [9] and query covers [12]. Generalizing the original single-document notion [9], a query q is a *keyquery* for a document set D with respect to a reference search engine S, if it fulfills the following conditions: (1) every $d \in D$ is in the top-k results returned by S on q, (2) q has at least l results, and (3) no subset $q' \subset q$ returns every $d \in D$ in its top-k results. The generality of a keyquery is defined by the parameters k and l (e.g., 10 or 50). We argue that the top-ranked results returned by a keyquery for D cover topics similar to the documents themselves—rendering keyqueries a promising concept for identifying related work.

The power set $\mathcal{Q} = 2^W$ forms the range of queries that can be formulated from the extracted keyphrases W of a document set D. A query q is said to *cover* the terms it contains. The more terms $w \in W$ are covered by q, and the more documents $d \in D$ are among q's top results, the better the query describes

Algorithm 1. Solving KEYQUERY COVER

Input: Sets D of documents and W of keywords, keyquery generality parameters k and l
Output: Set Q of keyqueries covering W

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

the topics represented by D's vocabulary W. The covering property [12] states that (1) in a proper set Q of queries for W, each term $w \in W$ should be contained in at least one query $q \in Q$ (i.e., the queries "cover" W in a set-theoretic sense) and (2) Q should be *simple* (i.e., $q_i \not\subseteq q_j$ for any $q_i, q_j \in Q$ with $i \neq j$), to avoid redundancy. The formal problem we tackle then is:

KEYQUERY COVER
Given: (1) A vocabulary W extracted from a set D of documents
(2) Levels k and l describing keyquery generality
Task: Find a simple set $Q \subseteq 2^W$ of queries that are keyquery for every $d \in D$ with respect to k and l and that together cover W.

The parameters k and l are typically set to 10, 50, or 100 but it will not always be possible to find a covering set of queries that are keyqueries for all documents in D. In such a case, we strive for queries that are keyqueries for a $|D| - 1$ subset of D.

Solving KEYQUERY COVER. Our approach has four steps (pseudocode in Algorithm 1). First, all keywords $w \in W$ are removed that return less than l results when submitted to the search engine S (lines 1–2); queries including such terms can not be keyqueries.

In a second step, the remaining terms $w \in W$ are iterated and added to an intermediate candidate query q (lines 3–5). If q is a keyquery for all papers $d \in D$, it is added to the set Q of keyqueries and q is emptied (line 6). Then the next query is formed from the remaining terms, etc., until the last term $w \in W$ has been processed.

After this first iteration, not all terms $w \in W$ are necessarily covered by the set Q of keyqueries. In this case, q is not empty (line 7) and we again go through the terms $w \in W$ (line 8) and consecutively add terms w to q—as long as no keyquery is found and simplicity of Q is not violated (line 9). According to the keyquery definition, keyqueries that are already in Q must not be contained

in the candidate query q. We hence omit terms $w \in W$ that would cause q to contain a keyquery $q' \in Q$.

After this iteration, we do not further deepen the search but output Q although still not all terms $w \in W$ may be covered. This heuristic serves efficiency reasons and our experiments show that often all possible keywords are covered.

Solving Related Work Search. The pseudocode of our algorithm using keyquery covers for related work search is given as Algorithm 2. Using the Keyquery Cover algorithm as a subroutine, the basic idea is to first try to find keyqueries for all given input papers, and to add the corresponding results to the set $\mathcal{C}$ of candidates. If there are too few candidates after this step, Keyquery Cover is solved for combinations with $|\mathcal{I}| - 1$ input papers, then with $|\mathcal{I}| - 2$ input papers, etc.

The vocabulary combination (line 3) is based on the top-20 keyphrases per paper extracted by KP-Miner [8], the best unsupervised keyphrase extractor for research papers in SemEval 2010 [18]. The terms (keyphrases) in the combined vocabulary list W are ranked by the following strategy: First, all terms that appear in all papers ranked according to their mean rank in the different lists. Below these, all terms contained in $(|D| - 1)$-sized subsets ranked according to their mean ranks, etc.

In the next steps (lines 4–6), the Keyquery Cover-instance is solved for the subset D and its combined vocabulary W, the found candidate papers are added to the candidate set $\mathcal{C}$. In case that enough candidates are found, the algorithm stops (line 7). Otherwise, some other input subset is used in the next iteration. If not enough candidates can be found with keyqueries for more than one paper, the keyqueries for the single papers form the fallback option (also applies to single-paper inputs).

In our experiments, we will set $k, l = 10$ and $c = 100 \cdot |\mathcal{I}|$ (to be comparable, also the baselines are set to retrieve 100 candidate papers per input paper).

Algorithm 2. Solving Related Work Search

Input: List $\mathcal{I}$ of input papers, number c of desired related papers, keyquery generality parameters k and l
Output: Set $\mathcal{C}$ of candidate related papers

1: **for** $i \leftarrow |\mathcal{I}|$ down to 1 **do**
2: **for all** $D : D \in 2^{\mathcal{I}}, |D| = i$ **do**
3: $W \leftarrow$ combine vocabularies of documents in D
4: $Q \leftarrow$ Keyquery Cover(D, W, k, l)
5: $C \leftarrow$ combined at most top-l results of each $q \in Q$
6: $\mathcal{C} \leftarrow \mathcal{C} \cup C$
7: **if** $|\mathcal{C}| \geq c$ **then break**
8: **return** $\mathcal{C}$ ranked by highest similarity to any document in $\mathcal{I}$

3.3 Combining Approaches

To combine different Related Work Search algorithms (e.g., Google Scholar's related articles and our keyquery approach), we use a simple interleaving procedure. First, the results of the individual approaches are computed and ranked as described above. We then interleave the ranked lists by first taking the first rank from the first approach, then the first rank from the second approach and so on, then the second rank from the first approach, etc. Already contained papers are not added again.

4 Evaluation

To experimentally compare the algorithms, we conduct a Cranfield-style experiment on 200,000 computer science papers. Topics and judgments are acquired in a user study.

4.1 Experimental Design

In general, there are two different approaches to evaluate algorithms for Related Work Search. A widely used method is to take the reference lists of papers as ground truth for the purpose of evaluation. Some of the references of a single input paper are hidden and it is measured whether the Related Work Search algorithm is able to re-identify these. The second possible method uses relevance judgments assigned by scholars. These judgments then state whether a recommended paper is relevant to a specific topic or not. Since the first method is rather biased towards the citation graph and also not really representing the use case we have in mind, we choose the second approach and utilize human relevance judgments from a carefully designed user study.

Paper Corpus. We crawled a corpus of computer science papers starting from the 35,000 papers published at 20 top-tier conferences like SIGIR, CHI, CIKM, ACL, STOC, and iteratively including cited and citing papers until the desired size of 200,000 papers was reached. In the crawling process, not all papers had a full text available on the web (for 57 % of the corpus, we have an associated full text). In this case, only the abstracts and metadata were obtained when possible (contained for 43 % of the corpus) but often not even abstracts were available and the respective papers were not included. On average, only 7.0 of the 13.9 references in a paper and 6.8 of the 9.5 citations to a paper could be crawled. Thus, only about 60 % of the cited/citing papers are contained in our corpus. Not surprisingly, especially older papers often were not available. The papers included in our corpus have been published in the years 1962–2013 (more than 75 % published after 2000). Starting from the 20 seed conferences, we included papers from about 1,000 conferences/workshops and 500 journals.

Experimental Setup. Sofia Search is run on the citation information contained in the metadata of the papers. Note that due to the non-availability of 40 % of the cited/citing papers, our corpus might not be optimally suited for Sofia Search. Still, this was not intended in the corpus design and could not be avoided—it rather represents the realistic web scenario but should be kept in mind when analyzing Sofia Search's performance.

In order to run the query-based baseline and our keyquery cover algorithm, we indexed the corpus papers using Lucene 5.0 while treating title, abstract, and body text as separate fields. In case that no full text is available in the corpus, only title and abstract are indexed. The retrieval model is BM25F [26] with different boost factors: the title is the most important followed by the abstract and then the body text.

Whenever a corpus paper with only title and abstract is used as an input paper for our keyquery-based algorithm, the keyphrase extraction is done on these two fields only and text similarity of a candidate paper is also only measured against these two fields (similar to the Sofia Search and Nascimento et al. baselines).

User Study Design. Topics (information needs) and relevance judgments for a Cranfield-style analysis of the Related Work Search approaches are obtained from computer science students and scholars (other qualifications do not match the corpus characteristics). A study participant first specifies a topic by selecting a set of input papers (could just be one) and then judges the found papers of the different approaches with respect to their relevance. These judgments are the basis for our experimental comparison.

In order to ensure a smooth work flow, we have built a web interface that also allows a user to participate without being on site. The study itself consists of two steps with different interfaces. In the first step, a participant is asked to enter a research task they are familiar with and describe it with one or two sentences. Note that we request a familiar research task because expert knowledge is later required in order to judge the relevance of the suggested papers. After task description, the participants have to enter titles of input papers related to their task. While the user is entering a title, a background process automatically suggests title auto completions from our corpus. Whenever a title was not chosen from the suggestions but was manually entered, it is again checked whether the specified paper exists in our corpus or not. If the paper cannot be found in the collection, the user is notified to enter another title. After this two-phase topic formation (written description + input papers), the participants have to name at least one paper that they expect to be found (again with the help of auto completion). Last, the users should describe how they have chosen the input and expected papers to get some feedback of whether for instance Google Scholar was used which might bias the judgments.

After a participant has completed the topic formation, the different recommendation algorithms are run on the input papers and the pooled set of the top-10 results of each approach is displayed in random order for judgment (i.e., at most 40 papers to judge). For each paper, the fields title, authors, publication

venue, and publication year are shown. Additionally, links to fade in the abstract and, if available in our corpus, to the respective PDF-file are listed. Thus, the participants can check the abstract or the full text if needed.

The participants assessed two criteria: relevance and familiarity. Relevance was rated on a 4-point scale (highly relevant, relevant, marginally, not relevant) while familiarity was a two-level judgment (familiar or not). Combining the two criteria, we can identify good papers not known to the participant before our study. This is especially interesting since we asked for research topics the participants are familiar with and in such a scenario algorithms identifying relevant papers not known before are very promising.

Characteristics of the User Study. In total, 13 experienced scholars and 7 graduate students have "generated" and judged 42 topics in our study—in a previous study we had another 25 topics with single-paper inputs only [11]. The scholars typically chose topics related to one of their paper projects while the graduate students chose topics from their Master's theses. On average, the topic creation took about 4 min while the judgment took about 27 min. For 23 topics, one or two input papers are given, while for 19 topics even three or up to five input papers were specified. For most topics (80 %), two or more expected papers were entered. The number of different papers returned by the four algorithms varies highly. For the topic with the most different results, the participant had to judge 37 papers, while the topic with the least different results required only 13 judgments. On average, a participant had to judge 28 papers. In total, 31 % of the results were judged as highly relevant, 22 % as relevant, 24 % as marginally relevant, and 23 % as not at all relevant to the topic. Interestingly, 79 % of the papers that are judged as relevant and 59 % of the highly relevant papers were unfamiliar to the user before the study. We will evaluate the algorithms with respect to their ability of retrieving unexpected good results when discussing the experimental results.

4.2 Experimental Results

We employ the standard retrieval effectiveness measures of nDCG [16] and precision for the top-10 results, and recall for evaluating the ability to retrieve the expected and also the relevant unexpected papers. The experimental results are reported in Table 1 (Left). The top four rows contain the results for the four individual algorithms while the bottom three rows contain the most promising combinations that can also be evaluated due to the pooling strategy. In the top-k results of our combination scheme described in Sect. 3.3 only results from the top-k of the combined algorithms are included.

General Retrieval Performance. We measure nDCG using the 4-point scale: high as 3, relevant as 2, marginal as 1, and not relevant as 0, and precision considering highly and relevant as the relevant class. The mean nDCG@10 and *prec*@10 over all topics are given in the second and third columns of Table 1 (Left). The top-10 of our new keyquery-cover-based approach KQC are the most relevant

Table 1. (Left) Performance values achieved by the different algorithms averaged over all topics. (Right) Percentage of top-10 rank overlap averaged over all topics.

Algorithm	nDCG@10	$prec$@10	rec_{e}@50	rec_{ur}@10
Sofia Search	0.60	0.59	0.33	0.20
Nascimento	0.58	0.56	0.34	0.16
Google Scholar	0.60	0.59	**0.43**	**0.21**
KQC	**0.62**	**0.60**	0.37	0.16
KQC+Google	**0.65**	0.63	**0.48**	0.21
KQC+Sofia	0.61	0.60	0.39	0.19
KQC+Sofia+Google	**0.65**	**0.64**	**0.48**	**0.24**

	Sofia	Nasc.	Google	KQC
Sofia Search	100	35	17	43
Nascimento	35	100	15	36
Google Scholar	17	15	100	18
KQC	43	36	18	100

among the individual methods on a par with Google Scholar and Sofia Search; both differences are not significant according to a two-sided paired t-test ($p = 0.05$). The overall best result is achieved by the KQC+Sofia+Google combination that significantly outperforms its components. Another observation is that most methods significantly outperform the Nascimento et al. baseline.

Recall of Expected Papers. To analyze how many papers were retrieved that the scholars entered as expected in the first step of our study, we use the recall of the expected papers denoted as rec_{e}. Since the computation of rec_{e} is not dependent on the obtained relevance judgments, we can compute recall for any k. We choose $k = 50$ since we assume that a human would often not consider many more than the top-50 papers of a single related work search approach. The fourth column of Table 1 (Left) shows the average rec_{e}@50-values for each algorithm over all topics. The best rec_{e}@50 among the individual methods is the 0.43 of Google Scholar. The combinations including KQC and Google Scholar achieve an even better result of 0.48. An explanation for the advantage of Google Scholar—beyond the probably good black-box model underlying the "related articles" functionality—can be found in the participants' free text fields of topic formation. For several topics, the study participants stated that they used Google Scholar to come up with the set of expected papers and this obviously biases the results.

Recall of Unfamiliar but Relevant Papers. We also measure how many "gems" the different algorithms recommend. That is, how many highly or relevant papers are found that the user was not familiar with before our study. Providing such papers is a very interesting feature that might eventually help a researcher to find "all" relevant literature on a given topic. Again we measure recall, but since we have familiarity judgments for the top-10 only, we measure the recall rec_{ur}@10 of unexpected but relevant papers listed in the rightmost column of Table 1 (Left). Again, the combination KQC+Sofia+Google finds the most of the unfamiliar but relevant papers.

Result Overlap. Since no participant had to judge 40 different papers, the top-10 results cannot be completely distinct; Table 1 (Right) shows the percentage of overlap for each combination. On average, the top-10 retrieved papers of two

different algorithms share 2–4 papers meaning that the approaches retrieve a rather diverse set of related papers. This again suggests combinations of different approaches as the best possible system and combining the best query-based method (our new KQC), Google Scholar, and Sofia Search indeed achieves the best overall performance. Having in mind the "sparsity" of our crawled paper corpus' citation graph (only about 60 % of the cited/citing papers could be crawled), Sofia Search probably could diversify the results even more in a corpus containing all references. The combination of the three systems in our opinion also very well models the human way of looking for related work such that the KQC+Sofia+Google combination could be viewed as very close to automated human behavior.

A Word on Efficiency. The most costly part of our approach is the number of submitted queries. In our study, about 79 queries were submitted per topic; results for already submitted queries were cached such that no query was submitted twice (e.g., once when trying to find a keyquery for all input papers together, another time for a smaller subset again). On the one hand, 79 queries might be viewed more costly than the about 27 queries submitted by the Nascimento et al. baseline ($10 \cdot |\mathcal{I}|$ queries) or the about 30 requests submitted for the Google Scholar suggestions ($11 \cdot |\mathcal{I}|$ requests; one to find an individual input paper, ten to retrieve the 100 related articles). Also Sofia Search on a good index of the citation graph is much faster than our keyquery-based approach. On the other hand, keyqueries could be pre-computed at indexing time for every paper such that at retrieval time only a few postlists from a reverted index [25] have to be merged. This would substantially speed up the whole process, rendering a reverted-index-based variant of our keyquery approach an important step for deploying the first real prototype.

5 Conclusion

We have presented a novel keyquery-based approach to related work search. The addressed common scenario is a scholar who has already found a handful of papers in an initial research and wants to find "all" the other related papers—often a rather tedious task. Our problem formalization of Related Work Search is meant to provide automatic support in such situations. As for solving the problem, our new keyquery-based technique focuses on the content of the already found papers complementing most of the existing approaches that exploit the citation graph only. Our overall idea is to get the best of both worlds (i.e., queries and citations) from appropriate method combinations.

And in fact, in our effectiveness evaluations of a Cranfield-style experiment on a collection of about 200,000 computer science papers, the combination of keyqueries with the citation-based Sofia Search and Google Scholar's related article suggestions performed best on 42 topics (i.e., sets of initial papers). The top-10 results of each approach were judged by the expert who suggested the topic. Based on these relevance judgments, we have evaluated the different algorithms and identified promising combinations based on the rather different returned

results of the individual approaches amongst which keyqueries slightly outperformed Sofia Search and Google Scholar suggestions.

Our experiments should be confirmed using other topic sets and paper corpora. The iSearch collection [22] may be suitable: 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 judgments may have been obtained using keyword queries, a fact that could give an advantage to our keyquery approach. Another promising future direction is to evaluate other background retrieval models, as well as keyphrase selection and weighting methods since our keyqueries heavily depend on these. Also the candidate ranking deserves further analyses. We have adopted simple text-based cosine similarity, which is also used in many other approaches but taking the number of citations or the publication venue into account may further improve the rankings.

References

1. Beel, J., Langer, S., Genzmehr, M., Gipp, B., Breitinger, C., Nürnberger, A.: Research paper recommender system evaluation: a quantitative literature survey. In: RepSys Workshop, pp. 15–22 (2013)
2. Bendersky, M., Croft, W.B.: Finding text reuse on the web. In: WSDM, pp. 262–271 (2009)
3. Bethard, S., Jurafsky, D.: Who should I cite: learning literature search models from citation behavior. In: CIKM, pp. 609–618 (2010)
4. Caragea, C., Silvescu, A., Mitra, P., Giles, C.L.: Can't see the forest for the trees? a citation recommendation system. In: JCDL, pp. 111–114 (2013)
5. Dasdan, A., D'Alberto, P., Kolay, S., Drome, C.: Automatic retrieval of similar content using search engine query interface. In: CIKM, pp. 701–710 (2009)
6. Ekstrand, M.D., Kannan, P., Stemper, J.A., Butler, J.T., Konstan, J.A., Riedl, J.: Automatically building research reading lists. In: RecSys, pp. 159–166 (2010)
7. El-Arini, K., Guestrin, C.: Beyond keyword search: discovering relevant scientific literature. In: KDD, pp. 439–447 (2011)
8. El-Beltagy, S.R., Rafea, A.A.: KP-Miner: a keyphrase extraction system for English and Arabic documents. Inf. Syst. **34**(1), 132–144 (2009)
9. Gollub, T., Hagen, M., Michel, M., Stein, B.: From keywords to keyqueries: content descriptors for the web. In: SIGIR, pp. 981–984 (2013)
10. Golshan, B., Lappas, T., Terzi, E.: Sofia search: a tool for automating related-work search. In: SIGMOD, pp. 621–624 (2012)
11. Hagen, M., Glimm, C.: Supporting more-like-this information needs: finding similar web content in different scenarios. In: Kanoulas, E., Lupu, M., Clough, P., Sanderson, M., Hall, M., Hanbury, A., Toms, E. (eds.) CLEF 2014. LNCS, vol. 8685, pp. 50–61. Springer, Heidelberg (2014)
12. Hagen, M., Stein, B.: Candidate document retrieval for web-scale text reuse detection. In: Grossi, R., Sebastiani, F., Silvestri, F. (eds.) SPIRE 2011. LNCS, vol. 7024, pp. 356–367. Springer, Heidelberg (2011)
13. He, Q., Kifer, D., Pei, J., Mitra, P., Giles, C.L.: Citation recommendation without author supervision. In: WSDM, pp. 755–764 (2011)
14. He, Q., Pei, J., Kifer, D., Mitra, P., Giles, C.L.: Context-aware citation recommendation. In: WWW, pp. 421–430 (2010)

15. Huang, W., Kataria, S., Caragea, C., Mitra, P., Giles, C.L., Rokach, L.: Recommending citations: translating papers into references. In: CIKM, pp. 1910–1914 (2012)
16. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. **20**(4), 422–446 (2002)
17. Kataria, S., Mitra, P., Bhatia, S.: Utilizing context in generative Bayesian models for linked corpus. In: AAAI, pp. 1340–1345 (2010)
18. Kim, S.N., Medelyan, O., Kan, M.-Y., Baldwin, T.: SemEval-2010 task 5: automatic keyphrase extraction from scientific articles. In: SemEval 2010, pp. 21–26 (2010)
19. Küçüktunç, O., Saule, E., Kaya, K., Catalyürek, Ü.V.: TheAdvisor: a webservice for academic recommendation. In: JCDL, pp. 433–434 (2013)
20. Livne, A., Gokuladas, V., Teevan, J., Dumais, S., Adar, E.: CiteSight: supporting contextual citation recommendation using differential search. In: SIGIR, pp. 807–816 (2014)
21. Lu, Y., He, J., Shan, D., Yan, H.: Recommending citations with translation model. In: CIKM, pp. 2017–2020 (2011)
22. Lykke, M., Larsen, B., Lund, H., Ingwersen, P.: Developing a test collection for the evaluation of integrated search. In: Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., van Rijsbergen, K. (eds.) ECIR 2010. LNCS, vol. 5993, pp. 627–630. Springer, Heidelberg (2010)
23. Nallapati, R., Ahmed, A., Xing, E.P., Cohen, W.W.: Joint latent topic models for text and citations. In: KDD, pp. 542–550 (2008)
24. Nascimento, C., Laender, A.H.F., Soares da Silva, A., Gonçalves, M.A.: A source independent framework for research paper recommendation. In: JCDL, pp. 297–306 (2011)
25. Pickens, J., Cooper, M., Golovchinsky, G.: Reverted indexing for feedback and expansion. In: CIKM, pp. 1049–1058 (2010)
26. Robertson, S.E., Zaragoza, H., Taylor, M.J.: Simple BM25 extension to multiple weighted fields. In: CIKM, pp. 42–49 (2004)
27. Stein, B., Hagen, M.: Introducing the user-over-ranking hypothesis. In: Clough, P., Foley, C., Gurrin, C., Jones, G.J.F., Kraaij, W., Lee, H., Mudoch, V. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 503–509. Springer, Heidelberg (2011)
28. Sugiyama, K., Kan, M.-Y.: Exploiting potential citation papers in scholarly paper recommendation. In: JCDL, pp. 153–162 (2013)
29. Tang, J., Zhang, J.: A discriminative approach to topic-based citation recommendation. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 572–579. Springer, Heidelberg (2009)
30. Tang, X., Wan, X., Zhang, X.: Cross-language context-aware citation recommendation in scientific articles. In: SIGIR, pp. 817–826 (2014)
31. Wang, C., Blei, D.M.: Collaborative topic modeling for recommending scientific articles. In: KDD, pp. 448–456 (2011)
32. Yang, Y., Bansal, N., Dakka, W., Ipeirotis, P.G., Koudas, N., Papadias, D.: Query by document. In: WSDM, pp. 34–43 (2009)