Martin-Luther-Universität Halle-Wittenberg
Faculty of Data Science
Degree Programme Informatik

Total Recall via Keyqueries: A Case Study for Systematic Reviews

Bachelor's Thesis

Paul Alexander Cahn

1. Referee: Prof. Dr. Matthias Hagen
2. Referee: M.Sc. Maik Fröbe

Submission date: May 3, 2021

Declaration

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

Göttingen, April 26, 2021

Paul Alexander Cahn

Abstract

Researchers conduct systematic reviews to gain and build a comprehensive understanding of a studied field. During the screening of documents, researchers aim for total recall to ensure that all relevant documents are covered in their systematic review. Creating a systematic review is time-consuming and can take several years. Systems for technology-assisted systematic reviews incorporate user feedback, whether a document was relevant or not, to learn presenting more yet unknown potential relevant documents. We propose a system that automatically creates so-called keyqueries which rank the known relevant documents in the top results of a reference search engine. This keyquery approach is motivated by research on related work search, where keyqueries retrieve additional related work for a given set of documents. Therefore, we construct keyqueries for the documents labeled as relevant in the systematic review to identify new, potentially relevant documents. We compare our keyquery-based approach with four classical machine-learning approaches and the state-of-the-art approach on three simulated systematic reviews with biological research topics. The Evaluation shows that our keyquery-based approach outperforms our implementation of logistic regression and decision table, and is comparable to a random forest approach. The state-of-the-art and our naive Bayes approach both outperform our keyquery-based approach.

Contents

Chapter 1

Introduction

In a systematic review researchers try to gather all knowledge about a specific scientific topic. Systematic reviews can be used used for biomedical purposes. For example, researchers studying a disease must gain all the knowledge about the disease to work productively on improving approaches and also make sure they hold certain standards. A systematic review helps researchers to avoid redundant work and can support the efficacy of an approach [17].

Creating such a systematic review can take a few months up to several years since it is elaborate to find all relevant documents for a researcher's topic in a large corpus of documents. Relevant documents include for example small, new study cases which have not been cited much. Hence, finding and evaluating relevant documents is not an easy task, because researchers must examine every single potential relevant document. While creating a systematic review, researchers aim for total recall since they want to find all relevant documents.

The corpus of potential relevant documents is created by researchers using a complex query. We assume that all potential relevant documents for the researched field are in that corpus. Researchers must identify all relevant documents of the corpus so that they can be incorporated in the systematic review. Technology-assisted systems for systematic reviews can simplify the screening of potential relevant documents for systematic reviews. Ideally, the process of screening documents can be as follows: Based on feedback provided by researchers, a technology-assisted system extracts more potential relevant documents from the corpus. The researchers screen these extracted documents and the technology-assisted system again can extract potential relevant documents based on the available feedback. This process is repeated until enough or all relevant documents have been screened.

To do this, technology-assisted systems must deliver total recall to ensure that no relevant document is missing. In the past machine learning approaches were successfully used for technology-assisted reviews which usually applied

continuous active learning frameworks.

In continuous active learning users have a corpus of documents, from which they screen documents and give feedback on them. A machine-learning algorithm uses this feedback to rank the corpus from relevant to non-relevant. Then users screen and evaluate the top results of the ranked documents, and the newly labeled documents are used to update the machine learning algorithm. The relevant documents of the corpus tend to be in the top results of the ranked documents due to the predictions of the machine-learning approach. Such a continuous active learning framework was already presented in the year 1994 [15].

Suppose technology-assisted systems could correctly predict that there are no more relevant documents, researchers could create systematic reviews much faster because they must mostly read and evaluate the relevant documents. Many systems are trying to achieve total recall as soon as possible, so researchers screen most of the relevant documents prior to the irrelevant documents. The best of those technology-assisted systems use machine-learning based approaches with an continuous active learning framework, using identified relevant documents to predict if an unknown document is relevant.

We regard the problem of finding relevant work from a different angle and are using so-called keyqueries that have been proven useful in related work search [12]. A keyquery is a query generated from a document with the property to retrieve that document in the top results of a reference search engine. So instead of predicting unknown documents as relevant, we use keyqueries that retrieve the known relevant documents at the top positions of a search engine to get related work. The underlying hypothesis is, that the newly retrieved related work to the known relevant documents is also relevant. With more feedback on documents provided by the user more keyqueries can be created and more relevant documents can be found.

An overview over the related work is provided in Chapter 2. We will present the development of our keyquery-based system in Chapter 3, while we first discuss machine-learning approaches and then the keyquery-based approach. The performance of our keyquery-based approach against four machine-learning based approaches and the state-of-the-art approach is presented in Chapter 4. An outlook on future work directions regarding technology-assisted reviews via keyqueries and the conclusion is given in Chapter 5.

Chapter 2

Related Work

A wide range of systems is trying to help scientists create systematic reviews or achieve total recall in finding relevant literature. Such systems are mostly either search-based or machine-learning-based [26]. In this chapter we first introduce search-based approaches and then machine-learning-based approaches. After that we present the keyquery approach for finding related work, which we will use later for tackling the total recall task for systematic reviews.

The total recall task is to identify the smaller subset of all relevant documents in the set of all documents. In the total recall TREC conferences of 2015 and 2016 many participants tried to tackle the total recall task. Machine-learning based approaches tried to use logistic regression or random forest models with different features [11]. Search-based approaches used for example query expansion techniques where the original query is modified in order to find relevant documents [10]. Creating good search queries is difficult, therefore it is examined by researchers.

However humans are not very effective at creating queries because often they can not extract the essential terms in documents to create effective queries [26]. A simple approach to automatically create search queries was developed by Alharbi et al. for the CLEF eHealth task. The authors extract essential terms, like MeSH terms, from the title and abstract of documents. MeSH is medical terminology structured in a tree data structure, where more abstract terms are parent nodes of concrete terms regarding a search topic [16]. The extracted terms are the clauses of a query for a document. For each extracted query of a document they build tf-idf based feature vectors for this document. Then they ranked the documents by how close the vectors were to the vector of the tf-idf vector of the topic description. The closer they were, the higher the document was ranked. This approach is better than a random ordering of the documents [1]. The queries themselves are not modified and do not ensure that they are well build queries.

Scells et al. created an approach to automatically modify boolean queries to improve the search results. Their system optimizes an initial query in a way that it should return mostly relevant papers and few irrelevant ones. To do that, they presented a Query Transformation Chain. The clauses of a given query are modified until a stopping criterion is met. The authors present several transformation paradigms to change a query. With those new queries the retrieval of relevant documents shall be improved [19].

The first transformation is to switch AND and OR operators in the query. The second restricts clauses for specific paragraphs in documents. For example, a term must occur in the abstract of a document. Thirdly, the MeSH explosion. In MeSH explosion, a retrieval model uses a MeSH term of the query and all terms that are child nodes of this term in the MeSH tree. Fourth, the MeSH parents working the other way round using the parent nodes of a given MeSH term. Fifth, the system removes clauses of the query. Last, abstract concepts of medical terms are converted into a single keyword which is then added to the query. The process of transformations is stopped after a maximum of five transformations. To select the best query, they use a machine-learning approach with boosted regression trees [19]. This approach can only modify the initial query, so the modified query is strongly dependent on the initial query, which was designed by humans.

Surita et al. created a system which automatically creates questions as queries from documents. These questions indicate what the documents can answer. The questions are computed from a document by a trained deep learning model. Then they rank each question by a score. The score shows how well the question can summarize the entire corpus. They take the so-called BERTScore, which computes the similarity between the automatically generated question and the initial research question. The top three questions are considered as the ones, which can summarize the corpus the best. When these questions are used as queries, they should retrieve mostly relevant articles due to the topic's description represented in the questions [20].

In a technology-assisted review generation the researcher always gives feedback on presented documents. This feedback can be well used by machine-learning based approaches. Hence in the following paragraphs we will present some machine-learning-based approaches. One approach that supports researchers on the total recall task is the HiCal algorithm which uses the users' feedback to predict if an unknown document is relevant. HiCal extracts tf-idf based feature vectors of relevant labeled documents. After each feedback it reweighs the used logistic regression model. With more feedback on documents the algorithm can predict better if a given document is relevant or not. The predicted relevant documents are given back to the user to evaluate. In general a machine learning model classifies unlabeled data and ranks them from most

relevant to non-relevant. The HiCal algorithm is proven to be a solid baseline for the total recall task [4]. In the following paragraphs, this approach will be discussed in further detail. It is the state-of-the-art algorithm in the technology-assisted review generation.

HiCal uses a logistic regression classifier which is trained on the users' feedback. For the feature extraction they use only specific paragraphs like the abstract, the anchor texts, or a summary. Also the model is trained to improve the classifier continuously with every user feedback. A single iteration of the active learning loop is as follows: First, they take 100 random documents and label them as not relevant. They take the description of the researched topic as a relevant document. With these and the labeled documents from the user, they train a logistic regression classifier. Then they classify the unlabeled documents, sort them by the classifier's score and give the top results back to the user [23]. Screening whole documents to decide if a document is relevant takes a lot of time.

Hence the authors later developed a more sophisticated version of HiCal. In order to decrease the time for the users, only certain excerpts of the document are presented to them. The users then could label the documents based only on these excerpts. A user study found that this gives even better performance on correct labeled documents than if the user is confronted with the document's full text. This increases efficiency and the performance of the logistic regression model is improving when the feedback on the screened documents is done better [24]. The logistic regression classifier however is still updated after every single feedback of the user.

To renew the model so often is very costly. Therefore they developed heuristics for dynamic batch sizes, so-called refreshing methods. The first method is the BMI method, where the first batch is size 1. Using the formula $k \leftarrow k + \frac{k+9}{10}$ increases the batch size exponentially, while k is the batch size. The second method uses static batch sizes, the most inefficient method. The third method is called partial refresh. After m judgments of documents, all judgments are used to update the classifier and rerank the unlabeled documents. The l documents with the highest score are being stored in a partial refresh set. After each new judgment of the user, this labeled document is used to update the model. However, the model only evaluates the documents in the partial refresh set [7]. The fourth method is Precision Based Refreshing. When the logistic classifier's output is below a defined threshold, every judgment is taken to refresh the classifier and re-score the output documents. The last method is called recency weighting, and the recently judged documents are being used more frequently for training [7].

Their evaluation shows that all these methods have a comparable recall, so selecting the refreshing method does not significantly impact the results.

However, the BMI method is a much more efficient than the other methods [7]. If the relevance feedback improved, when the users are confronted only with excerpts, the machine-learning model may work well with excerpts as well. The logistic regression classifier is then trained with partly whole documents and partly with only sentences of documents that are considered to hold much content. The evaluation shows that the performance of the HiCal algorithm was similar to before but the runtime improved [25]. The results of all those variants of the HiCal approach perform well, the only drawback left is that it takes a lot of time for the machine-learning model to identify the last 10 to 15 percent of relevant documents.

To get these 10 to 15 percent of relevant documents, Zou et al. extended the HiCal system. To improve the HiCal system, the authors presented questions to the user, which the user answered with yes or no. With these answered questions, the authors want to find the remaining percentage of relevant documents in a corpus. The questions for their system are generated from an annotation of entities in the identified relevant documents. For this, they took documents the user labeled as relevant, and extracted entities with an entity-recognition system. Then they build simple questions like *Do you look for* $< entity >$?. Based on these answered questions and with the trained machine-learning model, the system provides the last relevant articles earlier than before [28]. This extension of the HiCal approach has some drawbacks that were tackled in the following approach.

Zou et al. tackled the problem of users not being able to give clear answers to the questions and the problem when to stop asking questions. First, the system asks a question that the user can answer with yes, no, or unsure. Second they implemented an automatic stopping criterion like a maximum of answered questions. Apart from that this new approach is only a slight improvement to the previous approach [27].

Rens van de Schoot and his team criticized approaches like HiCal for focusing only on one machine-learning algorithm. They also criticized other approaches for not being transparent enough. In their approach, they developed a system that can use several different machine learning algorithms depending on which one the user specifies [21].

They only use the title and the abstract because that is where researchers mostly decide whether a document is relevant or not. The corpus of potential relevant documents has mainly irrelevant documents. They undersample these irrelevant documents by giving relevant documents a higher weight for the training of the machine-learning model. Their experiments shows that after 30% of reviewed documents, around 90% of the relevant articles were shown to the user [21]. This approach might have one problem because users do not know which machine learning model is best suited for their task. Hence

it could be better if one takes several machine learning approaches but uses them as an ensemble of algorithms.

2.1 Related Work Search

Many assisting systems use machine learning algorithms or deep learning models, as [27], [21] or [14]. Lange et al. focused on the task of updating systematic reviews. For this, related work to relevant documents must be found. The authors tested a large number of machine learning and deep learning models to extract relevant related work to a systematic review. This new related work can then be used to update the systematic review. A model receives the papers of the systematic review and learns which papers are relevant. Based on this knowledge, the model can evaluate if a given unknown document is related or not. In their evaluation, they found that deep learning models do not outperform simpler machine-learning algorithms. Although the most noticeable impact on the outcome is the preprocessing of the data which can change an approaches performance significantly [14].

Another different approach of finding related work to a given set of documents comes from Stein et al. Here the authors used so-called keyqueries to find related work. A keyquery for a document is a query that returns the document in the top result ranks when given to a search engine [12].

Previous approaches to related work search focused either on citation-graph-based methods, content-based methods, or combined methods. Citation-graph-based methods follow links from given papers to cited papers until a condition is met [8]. Content-based methods use keywords from the set of given papers to retrieve similar ones [6]. Combined methods use both approaches. A drawback of these is that they lead to very specific papers [12].

The top-ranked results of a keyquery to a search engine are considered as related to the input paper because the keyquery is very precise to retrieve the wanted paper. The authors expand that approach to a set of documents. A query for a set of documents is a keyquery if and only if every document of the set is in the top k-results of the retrieved documents. Furthermore, there must be at least l results, and there is no other query with fewer keywords that can satisfy these requirements [12].

The system's output does not only depend on the used documents but also on the parameters k and l. l defines how many documents should be in the retrieval of the keyquery, and k defines the number of results considered top results [12].

From a set of documents, keywords are extracted that occur frequently in documents and occur in only few documents. Then the keywords that return

less than l results are removed. In the next step, keywords are iteratively added to a query until the keyquery criteria are fulfilled. If a query is a keyquery it is added to a set. This happens for all extracted keywords. The shortest keyqueries are then used to find related work. For this these keyqueries are given to a search engine which retrieves similar documents to the ones described in the keyquery [12].

Finding related work is very close to finding relevant documents for a given set of documents. Therefore we want to use this approach to tackle the total-recall task for systematic reviews. The techniques of finding related work have not yet been systematically tested for technology-assisted review creation. Therefore we introduce a system that uses keyqueries to help users attain total recall faster during systematic review creation in the following chapter.

Chapter 3

Keyqueries for Systematic Reviews

A systematic review is an evaluation of a researched topic according to a method or system [9]. A systematic review shows the latest state of science regarding a topic. Doing such a systematic review takes a lot of time because scientists must screen and evaluate all relevant documents. In this chapter, we are going to show how keyqueries can support scientists doing systematic reviews. Keyqueries are designed to find related work in a corpus. If they find relevant documents, users must mostly screen relevant documents. First, we show how the workflow for users trying to create a systematic review with a technological helping system looks like. After that we present several approaches that support screening relevant documents.

3.1 Human in the Loop

Technology-assisted systematic reviews usually employ the human-in-the-loop pattern to screen all relevant documents. In this pattern, users have a corpus with documents that contains all relevant documents regarding the researched topic. Users screen the documents of the corpus and label the documents as relevant or not relevant. A technology-assisting system uses the user's feedback and the query to propose a ranking of the remaining corpus in which relevant documents are on top. The more feedback the users give, the better the system can probably order unlabeled documents. After the reordering of the corpus, users label the top documents. The labeling of documents and reordering of the corpus is done in a loop until the system predicts that it has shown all relevant documents, the users think they have seen all relevant documents or the users have seen all documents.

Figure 3.1 shows the workflow of a technology-assisted review using the human in the loop pattern. To gain a complete overview of a topic users need a document corpus that contains all relevant documents. Users created

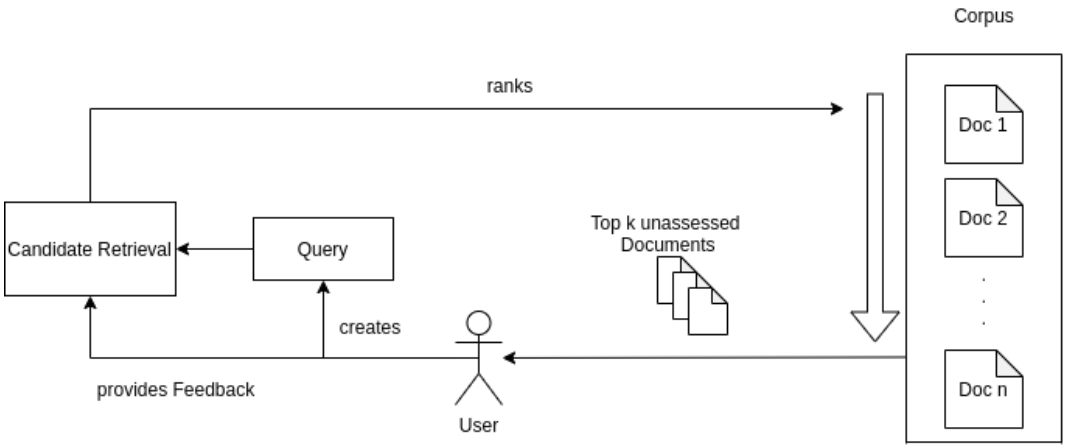

Figure 3.1: Human in the Loop Framework

that corpus by creating a query and getting its retrieved documents with a variety of search strategies and search engines. Typically such a corpus is vast and contains only a small minority of relevant documents. Users only want to screen those relevant documents. Hence the technology-assisting system tries to find those in the corpus and provides the reordered corpus in which documents are on top of which the system thinks they are relevant. This system, we call it Candidate Retrieval, uses the initial query and feedback on documents provided by the users for reordering the corpus. Users screen the top documents of the result of the candidate retrieval. The candidate retrieval uses the newly screened documents to reorder the corpus again in the next iteration.

3.2 Candidate Retrieval

As Figure 3.1 shows, the candidate retrieval is like a black box providing a modified or reranked version of the corpus. In the following section, we present systems that can provide such reranking methods. First, we show a system that reranks the remaining corpus based on machine-learning models. Other supporting systems mostly use machine-learning approaches for systematic review tasks. Secondly, we present a keyquery approach based on the keyquery definition of Stein et al. [12].

Approach	TP rate	TN rate	FP rate	FN rate
Naive Bayes	0,80	0,81	0,19	0,20
Decision Table	0,96	0,93	0,07	0,04
Logistic Regression	0,97	0,91	0,09	0,03
Random Tree	0,81	0,58	0,42	0,19
Random Forest	0,99	0,66	0,34	0,01
J48	0,96	0,91	0,09	0,04

Table 3.1: Machine Learning Pilot Study results achieved by different algorithms, with true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

3.2.1 Machine Learning Approaches

In technology-assisted reviews, machine-learning approaches are prevalent. A requirement for a machine learning approach is that it classifies documents as relevant and not relevant based on labeled documents. The labeled documents are used as training data for a machine-learning algorithm. We did a pilot study on several machine learning algorithms to find the ones that work the best for this task. These algorithms should learn based on user feedback whether unseen documents are relevant or not. Table 3.1 shows the results of this pilot study conducted on whole data sets. The true-positive rate indicates how many relevant documents were labeled correctly. The true-negative rate shows how many irrelevant documents were labeled as not relevant. The false-positive rate shows how many irrelevant documents were falsely predicted as relevant, and the false-negative rate shows how many relevant documents were classified as not relevant. One can see that logistic regression has the overall best results in our pilot study. The decision table has the second-best overall results. The random forest approach has the best true-positive and false-negative rate, which is good because we try to hold the false-negative rate as low as possible not to miss any relevant articles. Our candidate retrieval uses one of those three or naive Bayes, due to its simplicity, for different rankings of the remaining corpus.

When working with machine learning approaches, the most important task is how to preprocess the data and choosing which features are taken for training the model. For all documents, we removed all stopwords using an English dictionary. Then we used the porter stemmer to transform conjugated words to their base form. After that, the term frequency and the inverse document frequency for all terms in all documents are computed. The term frequency tf and the inverse document frequency idf are used as features for feature vectors

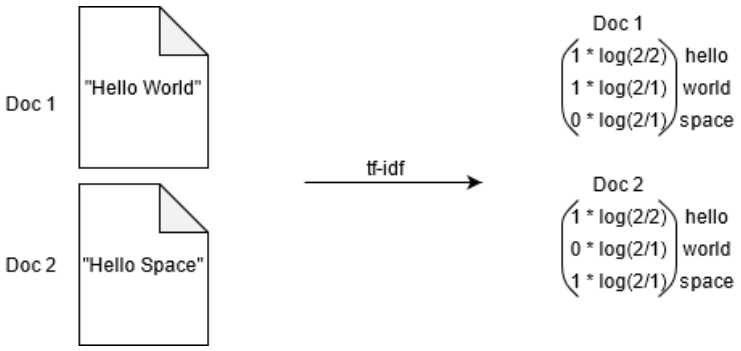

Figure 3.2: Data Preprocessing of two Documents

for a document. This transformation of a document to a feature vector is exemplified in Figure 3.2. The term frequency is the number of occurrences of that term in a document. The inverse document frequency is $\log(\frac{N_D}{f_t})$, where N_D is the number of documents and f_t is the number of documents in which term t occurs. The term frequency is multiplied by the inverse document frequency. This product will be computed for every term in every document and represented by a vector. In Figures 3.2 one can see word *hello* occurs one time in Document one. This is multiplied with inverse document frequency of *hello*, which is $\log(\frac{2}{2})$ which equals 0. Therefore the entry in the vector for the word *hello* of document one is zero. That indicates that *hello* is not a specific term for document one because it occurs in all other documents.

3.2.2 Keyquery Approach

In this section, our keyquery approach for technology-assisted systematic reviews is elucidated. First we show how a keyquery works, then we present how we can use it for technology-assisted review creation. Figure 3.3 illustrates how a keyquery works. Keyphrases are extracted into a vocabulary from relevant documents. A phrase is a group of words of a sentence that carries a special meaning. Keyphrases are phrases that contain information about the content of the document. These phrases are selected based on the tf-idf score, meaning that if a term or phrase has a high tf-idf score it is more likely to be taken as keyword or phrase. With this vocabulary, keyqueries are computed. First, the vocabulary's smallest phrases are taken and tested if they suffice the conditions of being a keyquery. A query is a keyquery if the relevant documents are in the top k retrieved documents of the query and if there are at least l retrieved documents. Suppose the vocabulary phrase does not suffice those conditions, more phrases are added to the initial phrase until it is a keyquery.

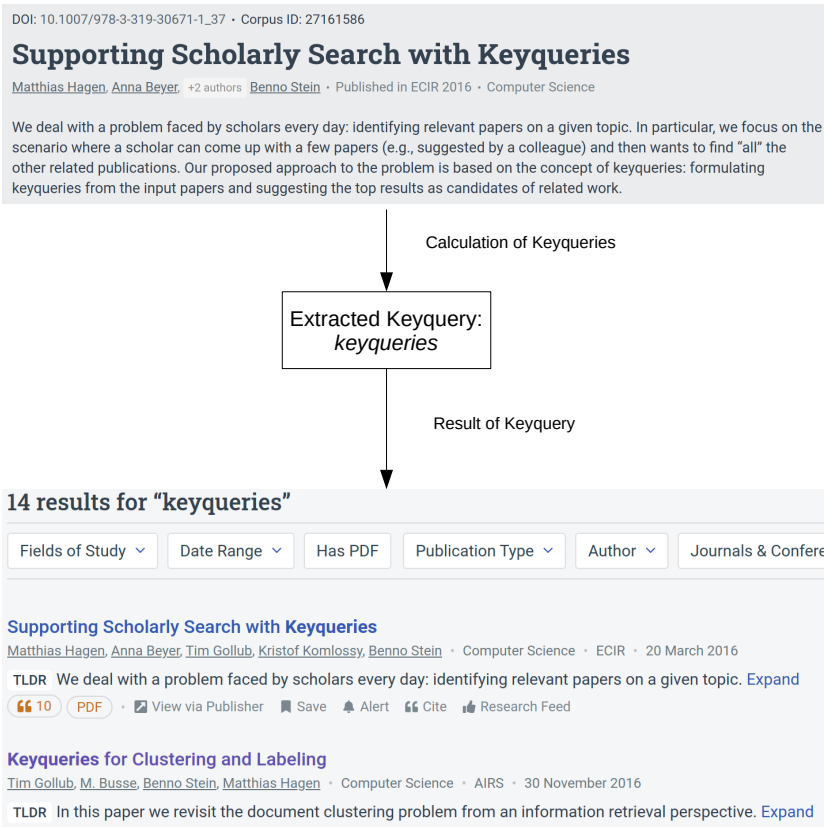

Figure 3.3: Example for keyquery functionality. At the top is a document, from this we extract a keyquery. The conditions for this keyquery are e.g.:
$k = 3$ (wanted document is in top 3 of the retrieval);
$l = 10$ (there are at least 10 results in the retrieval).
At the bottom is the result of a retrieval, where one can see that this keyquery suffices the given conditions.

In the example the document of which the keyquery is extracted from, is the first result of the retrieval and there are 14 results. So that *keyqueries* is a keyquery for the document "Supporting Scholarly Search with Keyqueries" where k equals 3 and l equals 10. A keyquery is designed to find similar documents as the ones the vocabulary is extracted from. We assume that similar documents to a set of relevant documents are also possibly relevant. Therefore we can use these keyqueries to retrieve relevant documents for creating systematic reviews. From relevant labeled documents, we extract keyphrases. With these, we create keyqueries and use them to retrieve documents from the corpus for each keyquery. The results of the keyqueries retrieval are combined so that from each result set the highest-ranked document is put in the final result. Then users can give feedback on the highest ranked documents and our approach can again create keyqueries, using the human in the loop pattern.

This approach can be improved by pseudo-relevance feedback. Pseudo-relevance feedback is the labeling of documents as relevant made by another technology-assisting approach. With pseudo-relevance feedback, our keyquery approach can create more and maybe better keyqueries and could retrieve relevant documents better. An ensemble of machine learning algorithms classifies the documents of the corpus using the users feedback. From this classification, the predicted relevant documents are given to the keyquery approach. This uses the pseudo-relevance feedback and the real feedback from the user to create keyqueries. However, it might be possible that the pseudo-relevance feedback is noisy, that some of the documents have the wrong label. With this feedback, the keyquery approach again creates keyqueries and follows the same procedure as the keyquery approach without pseudo-relevance feedback.

3.3 Methodology

This section describes the hyper-parameters of each algorithm we use to tune our approaches. For the machine-learning approaches we focus on parameters for data preprocessing and feature selection. As features we use the term frequency, tf-idf and for preprocessing we used undersampling techniques. For undersampling, we use a tool that removes not relevant documents until a desired ratio of relevant to not relevant documents is reached. That means that relevant documents have a greater influence on the training of the machine learning model compared to unbalanced data sets. The weighting factors are [0.5, 1.0, 1.5, 2.0]. A weighting factor of 2.0 means that there are twice as many not relevant documents as relevant ones, 0.5 means that there are twice as many relevant documents than irrelevant documents.

Our keyquery approach uses a wider range of parameters. First, we use

the hyper-parameter k and l in range [10, 20, 50, 100]. l states that there must be at least l results in the keyqueries retrieval, k states that all relevant documents must be under the top-k results. Another parameter is m. This parameter states the absolute ratio of relevant documents in the top-k result. For example, k has the value ten, and m value seven. Then there must be at least seven relevant documents in the top-k results. m is set to [10%, 30%, 50%, 70%, 100%] of the different k values. Also, the way of merging or interleaving the keyquery results is tested, because if many keyqueries are computed each has its own result retrieval. We test balanced and teamdraft interleaving. In balanced interleaving the top result of each ranking is taken for the final result ranking. As Radlinski described, balanced interleaving is not good in merging almost identical retrieval results [18]. Therefore we may get better results when using teamdraft interleaving, which randomly picks the ranking of which the next top result is taken from. Another parameter that could have an impact on the results is the removal of redundant keyqueries. Here keyqueries that return the same results as others will not be used. The last parameter we evaluate on our keyquery-based approach is the use of pseudo-relevance feedback, where our machine-learning based approaches provide potential relevant documents of which our keyquery-based approach creates keyqueries.

Chapter 4

Evaluation

This chapter evaluates and discusses the machine learning approaches and the keyquery approach that helps screening documents for creating systematic reviews. First, we describe our experiment setup, then evaluate the machine learning approaches and discuss the results. After that, we assess the keyquery approach.

4.1 Experiment Setup

For our evaluation, we used two datasets of the Julius-Kühn-Institute, short JKI. Their datasets were created during the generation of a systematic review for biological research. Those data sets are from 2018 and 2019 and are called CEEDER-2018 and CEEDER-2019. The documents are about ecological and environmental topics with the query concerning environmental management. Besides, we took a dataset from a genome TREC conference also from 2019 where the documents are mainly about genome editing with the query if there are documents about the traits of genome editing in plants and crops. There are about 27000 documents in CEEDER-2018, CEEDER-2019 contains about 14000 documents with each about 5% relevant documents, and in the genome TREC data set are about 5000 documents of which are 27% relevant. There is relevance judgment on each document in all sets. This judgment can simulate real users evaluating documents. Also, the initial query for each dataset is given. We chose CEEDER-2018 as our validation set to tune the parameters, and CEEDER-2019 and genome TREC are used as test sets.

Due to the number of documents and the real user judgment on the documents, we created an automatic human-in-the-loop system for testing purposes. This system simulates an actual user by labeling the top-k documents after the candidate retrieval has reordered the corpus. With this, tests can run much faster than if users have to screen the documents online. We assume

these relevance judgments to be correct.

As a performance measure, we take the work saved oversampling (WSS) measure. It indicates how much work has been saved using the technology-assisting system instead of screening the corpus without it. We apply the definition from Cohen et al. [2]. The formula is as follows:

$$WSS@Recall = \frac{True\ Negatives\ +\ False\ Negatives}{Total\ Number\ of\ Docs} - (1\ -\ Recall).$$

The $True\ Negatives + False\ Negatives$ are the documents that have not been judged yet. For example, we have 100 in a corpus of documents and a recall of 85% after 60 seen documents. Then the WSS@85% equals 25% because with the assisting system, users have seen 25% more relevant documents with the same amount of work. Hence the higher the WSS value, the better because a high WSS value means that more relevant documents were screened to achieve this recall with less effort. In creating systematic reviews, it is crucial to have a high recall. Therefore we evaluate our systems with a recall of 85%, 90%, and 95% on the WSS measure. In addition to the WSS measure we plotted results with the used effort against recall.

To compare our approaches against others, we took two approaches as baselines. The first is BM25 retrieval, where every document is sorted by a score [13]. The second approach is a machine-learning-based algorithm from Cormack et al. called HiCal [3], which was discussed in the related work Chapter 2. Here the structure is similar to our machine learning approach, using the human-in-the-loop pattern and a logistic regression classifier.

4.2 Evaluation of our Approaches

In the following sections we discuss our experiments regarding the machine-learning approaches and the keyquery-based approach. First we tune parameters for our machine-learning based approaches on CEEDER-2018. As parameters we first use different features for training the machine-learning models and secondly we use undersampling methods to reweigh the dataset. Then we discuss their performance on the other datasets. After that we tune our keyquery-based approach on regard to the parameters k, l and m. Then we test different interleaving methods and at last we test if pseudo-relevance feedback is improving the performance of the keyquery-based approach. In the end we discuss its performance on CEEDER-2019 and genome TREC dataset.

4.2.1 Evaluation of Machine Learning Approaches

As Lange et al. stated, that improving feature selection and data preprocessing is more important then selecting and finetuning a sophisticated machine-learning model.[14]. Therefore, we focused on feature selection and data preprocessing and not on finetuning the different machine learning models. We used three different features. The first feature is boolean term occurrence. For each term it is checked if it occurs in a document. The second is the term frequency (tf), where it is checked how often a term occurs in a document. The last is the term frequency multiplied by the inverse document frequency (tf-idf). The inverse document frequency is defined as follows: $log\frac{N}{\sum_{D:t\in D} 1}$ where N is the number of documents and $\sum_{D:t\in D} 1$ is the number of documents in which term t occurs.

We evaluated four machine-learning approaches with these features on CEEDER-2018: A naive Bayes approach, a logistic regression approach, a random forest approach and a decision table approach. Table 4.1 shows the results with regard to the WSS score. The decision table has the worst performance of all used approaches and the different features did not have an impact on the results. The third best of our approaches is the logistic regression approach. Here the results differ only a little between the different features too. Interestingly our is significantly worse than HiCal which also uses a logistic regression model. The second best of our approaches is the random forest approach with the boolean term occurrence feature. Our naive Bayes approach significantly outperforms all of our own approaches but performs worse than HiCal. This is interesting because naive Bayes is a much simpler machine learning model than logistic regression.

The approaches with the overall best results are visualized. If the results were equal or it was not obviuous which one the best results were, we used the approaches with the tf-idf feature. We did this because we assumed that more complex features may lead to better results in future experiments on other datasets.

In Figure 4.1 one can see results of the discussed approaches where the recall is computed as a function against the effort. Contrary to our expectations, the different features did not significantly impact the results. One can see that naive Bayes and random forest are nearly identical to HiCal in the beginning but then their recall curve flattens faster. So they cannot extract relevant documents as precise as HiCal. Besides our decision table approach is similar to BM25, which uses no user feedback at all. Interestingly none of the used features did have a big impact on the results.

After evaluating the used features, we now assess another data processing method. High inequality in datasets between relevant and irrelevant docu-

Approach	WSS @85%	WSS @90%	WSS @95%
NB	**59.03**	56.75	47.53
NB TF	58.90	56.10	46.67
NB TF-IDF	58.78	**57.00**	**48.80**
Log	43.14	41.58	**35.14**
Log TF	43.22	**42.29**	33.73
Log TF-IDF	**43.73**	42.27	34.52
RF	52.82	**51.34**	**41.83**
RF TF	**54.30**	47.90	38.06
RF TF-IDF	53.01	49.83	35.68
DT	13.87	09.59	05.36
DT TF	13.87	09.59	05.36
DT TF-IDF	**13.87**	**09.59**	**05.36**
BM25	20.24	11.66	09.68
HiCal	64.75	65.27	59.96

TF = term frequency; TF-IDF = term frequency - inverse document frequency; NB = Naive Bayes; Log = logistic regression; RF = random forest; DT = decision table

Table 4.1: Machine Learning Approaches with different features. To quickly see the best overall result for each approach, we highlighted that result.

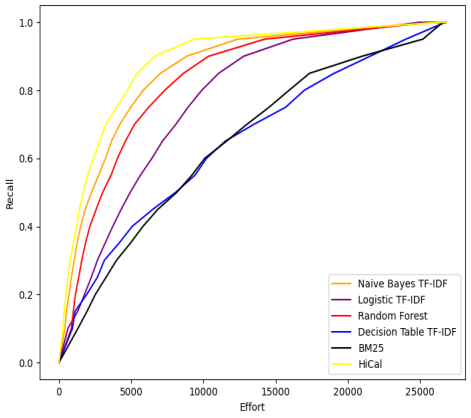

Figure 4.1: Recall-Effort Plot of Machine-learning based approaches with different features.

ments can result in bad machine-learning models because the machine-learning model could just predict the negative class and still achieve high performance measures due to highly unbalanced classes. Due to the high inequality of relevant and irrelevant documents in the here used datasets, we undersampled the class of irrelevant documents. Undersampling means that we do not use every document labeled as irrelevant for training the machine-learning model. We tested this undersampling with different distributions. A distribution of undersampling describes the ratio of relevant documents proportionally to the ratio of irrelevant documents. For example, a distribution of 1.0 means that the amount of relevant and irrelevant documents is the same. Distribution of 2.0 means there are twice as many irrelevant documents as relevant documents. We tested the algorithms with the features that had the best performance.

The results with an undersampling distribution range of [0.5, 1.0, 1.5, 2.0] are shown in Table 4.2. Here our logistic regression approach had the worst performance. It is worse than the logistic regression approach without undersampling. Interestingly for the logistic regression approach is that the undersampling distribution of 2.0 has no better performance than the undersampling distribution of 1.5. Although in the dataset of CEEDER-2018 the ratio of relevant documents is about 5%. The third best approach is the decision table approach. It improved significantly in contrast to the non-undersampling approach and apparently works best if there is the same amount of relevant and irrelevant documents. The second best approach is again the random forest

Approach	U-rate	WSS @85%	WSS @90%	WSS @95%
NB TF-IDF	0.5	57.53	55.31	47.49
NB TF-IDF	1.0	58.55	56.49	44.18
NB TF-IDF	1.5	**58.64**	56.45	47.99
NB TF-IDF	2.0	58.58	**57.22**	**48.26**
Log TF-IDF	0.5	29.63	24.23	17.84
Log TF-IDF	1.0	28.05	26.81	22.23
Log TF-IDF	1.5	**34.64**	**30.28**	22.02
Log TF-IDF	2.0	29.29	28.77	**23.74**
RF	0.5	54.30	53.70	48.24
RF	1.0	55.50	52.74	47.22
RF	1.5	**56.79**	52.71	40.40
RF	2.0	56.61	**54.39**	**48.36**
DT TF-IDF	0.5	36.27	**34.62**	25.15
DT TF-IDF	1.0	**43.99**	32.48	**25.45**
DT TF-IDF	1.5	38.78	23.10	15.45
DT TF-IDF	2.0	33.79	28.57	11.58

U-Rate = undersampling distribution rate;
TF-IDF = term frequency - inverse document frequency;
NB = Naive Bayes; Log = logistic regression;
RF = random forest; DT = decision table

Table 4.2: Machine Learning Approaches with Undersampling. To quickly see the best overall result for each approach, we highlighted that result.

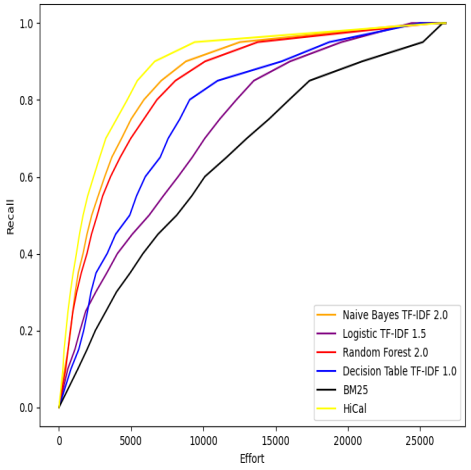

Figure 4.2: Machine Learning Approaches with Undersampling

approach, which slightly improved compared to the random forest approach without undersampling. The best of our machine-learning approaches is the naive Bayes approach, which like random forest improved slightly compared to no undersampling. Again none of our approaches were able to outperform HiCal but they all performed better than BM25. The overall best results of the undersampling approaches are visualized in Figure 4.2. Here the biggest difference between undersampling and no undersampling is seen in the curve of the decision table and logistic regression approach. On the other two approaches undersampling seemed to have no significant impact.

These tests on CEEDER-2018 have shown that the logistic regression approach with tf-idf feature and no undersampling, the naive Bayes approach with tf-idf feature and undersampling distribution of 2.0, the decision table approach with tf-idf feature and undersampling distribution of 1.0 and the random forest approach with boolean term occurrence feature and undersampling distribution of 1.5 had the best performances.

We use these approaches and test them on CEEDER-2019 and the genome TREC data set. Table 4.3 shows the results on the CEEDER-2019 data set. The results show that our logistic regression approach is the worst of our approaches, the third best is the decision table approach, the second best is the random forest approach and our best approach is the naive Bayes approach. The WSS measure indicates that the results are very similar to the ones on

Approach	WSS @85%	WSS @90%	WSS @95%
NB TF-IDF 2.0	**55.57**	52.20	**49.29**
Log TF-IDF	34.10	32.72	26.84
RF 2.0	52.45	**52.91**	48.71
DT TF-IDF 1.0	40.68	38.30	26.80
BM25	9.04	-0.38	-0.87
HiCal	**64.60**	**66.48**	**61.57**

TF-IDF = term frequency - inverse document frequency;
NB = Naive Bayes; Log = logistic regression;
RF = random forest; DT = decision table

Table 4.3: Machine Learning Approaches on CEEDER-2019

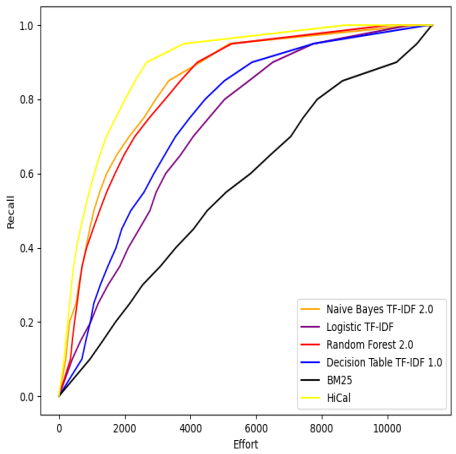

Figure 4.3: Machine Learning Approaches on the JKI 2019 data set.

CEEDER-2018. Also HiCal performed similar on both data sets only BM25 performed worse on CEEDER-2019.

Figure 4.3 shows those results in an effort-recall plot. There one can see that in the beginning, naive Bayes is almost identical to the HiCal approach indicating that it can predict unknown documents well with only few relevance judgments. One can see that on the JKI data sets, the naive Bayes approach is best compared to our approaches, and logistic regression performs worst compared to our approaches even though it also uses a logistic regression model just like HiCal.

The last tests of our machine-learning based approaches were run on the genome TREC data set. The parameters of each approach is the same as on the previous data set. Table 4.4 shows the results of our approaches against HiCal and BM25. Interestingly all approaches had a significantly worse performance than on the previous data sets. This could be because in the genome TREC data set there are about 25% relevant documents and those documents might have very different content, hence the machine-learning models can not predict very well if a document is relevant. Our logistic regression approach performed worst of all our approaches, the decision table approach performed significantly better. Our naive Bayes approach performed not significantly better than the decision table approach as on CEEDER-2018 or CEEDER-2019 and our random forest approach performed just slightly worse than HiCal.

Figure 4.4 visualizes the discussed results on the genome TREC data set. Here one can see that the random forest approach is similar to HiCal after

Approach	WSS @85%	WSS @90%	WSS @95%
NB TF-IDF 2.0	36.23	35.73	32.62
Log TF-IDF	24.17	23.46	19.51
RF 2.0	**43.03**	**41.02**	**37.23**
DT TF-IDF 1.0	35.34	32.57	27.07
BM25	1.76	2.15	3.59
HiCal	**44.95**	**42.38**	**38.32**

TF-IDF = term frequency - inverse document frequency;
NB = Naive Bayes; Log = logistic regression;
RF = random forest; DT = decision table

Table 4.4: Machine Learning Runs on genome TREC data set

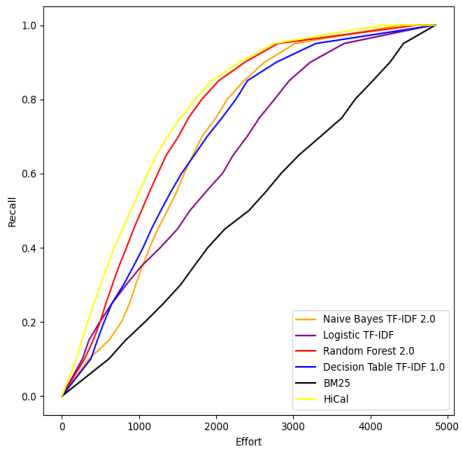

Figure 4.4: Machine Learning Approaches on the genome TREC data set.

evaluating half the documents. Naive Bayes has a worse performance in the beginning compared to the results on the previous data sets.

None of our approaches was able to beat the state-of-the-art HiCal algorithm. The feature selection and data preprocessing did not have the impact we supposed. The naive Bayes and random forest approach came closer to the HiCal results than BM25 and our other approaches.

4.2.2 Evaluation of the Keyquery Approach

In this section, we discuss the results of the keyquery approach. First we tune parameters on CEEDER-2018 and then we test them on CEEDER-2019 and the genome TREC data set. We performed a grid search for our keyquery system with different values for k, l and m. k states that a relevant document must be under the top-k results, m states how many relevant documents have to be in the top results. The parameter m can weaken the first condition if m has a smaller value than k. A small value of m weakens the condition of parameter k, in a way that a query is considered more likely to be a keyquery. The parameter l states that there must be at least l results. A large value for l means that the keyquery must be more general to retrieve many documents. A small value for k means that the keyquery must be more specific regarding that document. The parameters k and l were set to a range of [10, 20, 50, 100] and m was set to a certain percentage of the value of k. This means that for each k, m was set to [10%, 30%, 50%, 70%, 100%] of k's value. For example,

if k equals 10 m is set to [1, 3, 5, 7, 10].

We assume that a combination of a small value for k and a larger value for l should give good results because the keyqueries have to be so specific that the relevant document is in the top-k results, but also general enough that it retrieves many documents that might be related to the relevant document and hence possibly relevant. A small value for m would indicate that a query is more easily a keyquery. More keyqueries might result in better overall results even though they might not be keyqueries as defined. Due to the number of results evaluating all those different parameter settings, we only display a subset comprised of the best and worst results from that parameter study. These results are shown in Table 4.5.

One can see that the parameter m has a significant impact on the performance as can be seen on the first two results with a rising value of m the performance got worse. Although in most cases when l is larger than k the performance improved as seen in the fourth result. The best results for the value of m were reached when it was set to $0.3 \cdot k$. The selection of a smaller value for k than for l also gave marginally better results than other tests, but these differences are not significant. Compared to the machine-learning-based approaches the best results of the keyquery-based approach are better than the logistic regression and decision table approach on CEEDER-2018 and not significantly worse than the random forest approach.

Figure 4.5 visualizes these results. Noticeably the keyquery approaches do not slope as early upwards as the machine-learning-based approaches. This might be because the keyquery approach needs many relevant documents to find keyqueries. Therefore the keyquery approach needs more iterations to finally compute keyqueries. All keyquery-based approaches need to see more documents than HiCal to achieve similar recall of relevant documents. Therefore their performance is not as good as HiCal's. The higher the value of m, the later the approach can detect relevant documents. This is due to the fact that this approach has fewer keyqueries and can therefore not find relevant documents early. If m has a small value, this approach can detect relevant documents early, because the condition that a query is a keyquery is low. Then queries which does not suffice the strong conditions of being a keyquery become a keyquery. Then it is possible that more irrelevant documents are detected as relevant and the performance decreases.

From these tests we take the two best results for following the parameter tests. Each keyquery is given to a search engine which retrieves a ranked list of the remaining documents. The lists for each keyquery must be merged into a final list of the remaining documents and we tried two merging or interleaving strategies. The more trivial interleaving strategy, balanced interleaving, was used in the tests above. In the following tests we tested teamdraft interleaving.

Approach k	l	m	WSS @85%	WSS @90%	WSS @95%
Keyquery 10	10	1	45.19	41.12	34.71
Keyquery 10	10	7	33.85	29.10	23.98
Keyquery 20	20	2	47.70	44.61	38.50
Keyquery 20	50	6	**51.06**	**50.07**	37.69
Keyquery 50	50	34	20.24	11.66	9.68
Keyquery 50	100	25	23.15	20.72	11.91
Keyquery100	100	10	50.97	50.01	**43.24**
BM25			20.24	11.66	9.68
HiCal			64.75	65.27	59.96

Table 4.5: Keyquery Approach with different values for k, l and m; with the best results highlighted

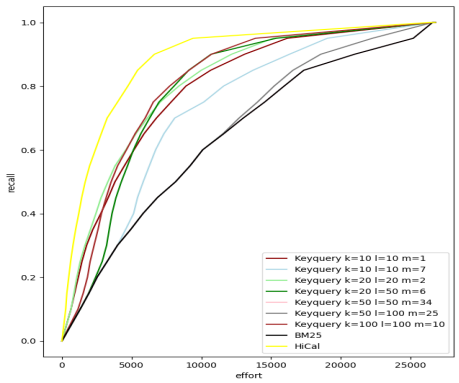

Figure 4.5: Keyquery Approach with different values for k, l and m

Table 4.6 shows the result of the keyquery approach with teamdraft interleaving and with balanced interleaving. The choice of the interleaving strategy did not have a vast impact on the results but it can be observed that the results with teamdraft interleaving performed slightly better. This makes the second result more similar to the random forest approach on CEEDER-2018 which is slightly worse than our naive Bayes approach which performed best. The results can also be seen in Figure 4.6.

Another parameter that we evaluated was the removal of redundant keyqueries. This method deteriorated the results, such that they performed partly worse than BM25. Therefore we did not evaluate further with this method.

All of the above used approaches have one flaw: they need to see many documents to function well. In the beginning, there are too few relevant articles to make keyqueries. Therefore we add pseudo-relevance feedback to the keyquery approach. Pseudo-relevance feedback are relevant labeled documents, labeled from our machine-learning approaches. When a document is predicted as relevant with a score of more than 80%, we consider it relevant. We used the score of 80%, so that the machine-learning approach is almost sure that a document is relevant. Our keyquery approach can use this document to create keyqueries. We used all four machine learning approaches to create pseudo-relevance feedback. Each machine learning approach can provide up to 5 pseudo-relevant documents if its predicted score is over 80%. We hypothesize that this amount of feedback suffices for the keyquery approach to create keyqueries earlier than on the previous approaches.

Table 4.7 shows the results of our keyquery approach with pseudo-relevance

Approach	I	WSS @85%	WSS @90%	WSS @95%
Keyquery (1)	Team	51.43	50.10	36.93
Keyquery (2)	Team	**54.50**	**52.69**	**45.26**
Keyquery (1)	Bal	51.06	50.07	37.69
Keyquery (2)	Bal	50.97	50.01	43.24
BM25		20.24	11.66	9.68
HiCal		64.75	65.27	59.96

I = Interleaving method;
Team = Teamdraft Interleaving; Bal = Balanced Interleaving;
(1) k=20 l=50 m=6; (2) k=100 l=100 m=10

Table 4.6: Keyquery Approach with Teamdraft Interleaving and the best results highlighted

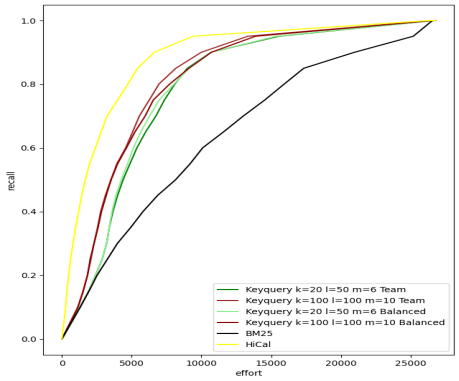

Figure 4.6: Keyquery Approach with Teamdraft and Balanced Interleaving

feedback from our machine-learning approaches against the results of our keyquery-based approach with teamdraft interleaving. One can see that the results with pseudo-relevance feedback have very similar results to the results with teamdraft interleaving. The approaches with pseudo-relevance feedback did not perform better with this feedback. Figure 4.7 shows the results of the keyquery approach with pseudo-relevance feedback against the keyquery approach without it. One can see that the curve of the runs with pseudo-relevance feedback starts earlier to differentiate from the initial BM25 ranking, which is taken when no keyqueries could be extracted. But this strong upwards slope does not last and it flattens so that they are both similar in the end. This may be because too many irrelevant documents were considered as relevant by the machine learning approaches.

After we evaluated the parameters for our keyquery-based approach on CEEDER-2018 we test them on CEEDER-2019 and genome TREC. Since pseudo-relevance feedback does not improve our keyquery approach's performance, we do not test it on CEEDER-2019 and genome TREC. On those data sets, we test our keyquery approach with the parameters as above for k, l and m and teamdraft interleaving against our four machine-learning approaches, BM25 and HiCal.

Table 4.8 shows the results of all tested approaches on CEEDER-2019. One can see that based on WSS score, our keyquery approach was better than BM25, the logistic regression approach and the decision table approach. The naive Bayes and random forest approach performed slightly better, and HiCal outperformed every approach. Figure 4.8 shows that in the beginning all

Approach	P	WSS @85%	WSS @90%	WSS @95%
Keyquery (1)	Pseudo	52.41	48.53	36.59
Keyquery (2)	Pseudo	54.29	52.02	**45.48**
Keyquery (1)		51.43	50.10	36.93
Keyquery (2)		**54.50**	**52.69**	45.26
BM25		20.24	11.66	9.68
HiCal		64.75	65.27	59.96

P = Pseudo-Relevance Feedback;
(1) k=20 l=50 m=6 Teamdraft Interleaving;
(2) k=100 l=100 m=10 Teamdraft Interleaving;

Table 4.7: Keyquery Approach with Pseudo-Relevance Feedback and the best results highlighted

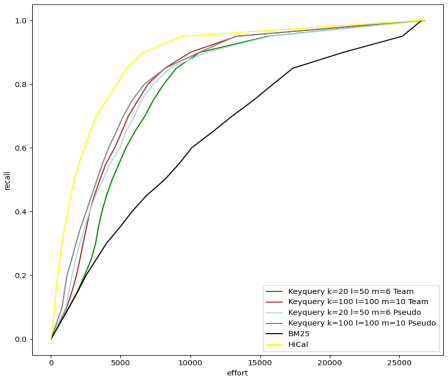

Figure 4.7: Keyquery Approach with and without Pseudo-Relevance Feedback

machine-learning based approaches had a better recall than our keyquery approach, but with more screened documents the recall of our keyquery approach improved significantly.

Our final test was on genome TREC. Here, our keyquery approach performed the worst on all three sets. Table 4.9 shows that both keyquery runs are very similar and that the keyquery run with the parameters $k = 100$, $l = 100$, and $m = 10$ that performed better in previous runs is now worse. On this set, our keyquery approach is only better than BM25. An interesting observation is different performance between the datasets. All approaches, not just ours, performed worse on the genome TREC data set even though the proportion of relevant documents is significantly higher than on CEEDER-2018 and CEEDER-2019. The relevant documents may have only a small intersection of similar information, so approaches working with the content to find relevant documents can not easily retrieve them. Figure 4.9 shows the results of all tested approaches on genome TREC in an effort-recall plot.

4.2.3 Discussion

The evaluation has shown that that the current state-of-the-art algorithm for technology-assisted reviews performs better than our machine-learning-based approaches and our keyquery-based approach. We tested our machine-learning algorithms on three different kinds of features: (1) boolean word occurrence, (2) tf and (3) tf-idf. Although we re-implemented HiCals logistic regression with weka, a java toolkit for machine-learning. Our re-implementation

Approach	U-Rate	WSS @85%	WSS @90%	WSS @95%
Keyquery (1)		51.06	**50.71**	41.32
Keyquery (2)		**53.33**	50.27	**43.71**
NB TF-IDF	2.0	**55.57**	**52.20**	**49.29**
Log TF-IDF		34.10	32.72	26.84
RF	2.0	52.45	52.91	48.71
DT TF-IDF	1.0	40.68	38.30	26.80
BM25		9.04	-0.38	-0.87
HiCal		**64.60**	**66.48**	**61.57**

TF-IDF = term frequency-inverse document frequency;
U-Rate = undersampling distribution rate;
(1) k=20 l=50 m=6 Teamdraft Interleaving;
(2) k=100 l=100 m=10 Teamdraft Interleaving;
NB = Naive Bayes; Log = logistic regression;
RF = random forest; DT = decision table

Table 4.8: Machine-Learning and Keyquery Runs on CEEDER-2019

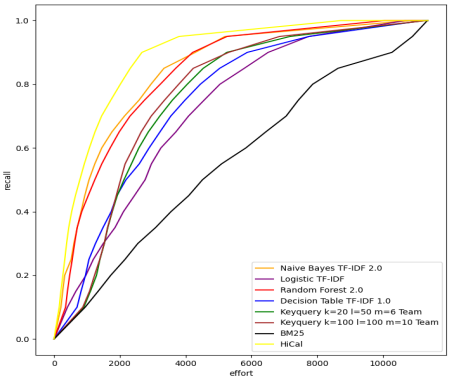

Figure 4.8: Machine-Learning and Keyquery Runs on CEEDER-2019

Approach	U-Rate	WSS @85%	WSS @90%	WSS @95%
Keyquery (1)		**19.20**	**15.55**	**6.72**
Keyquery (2)		17.40	13.61	6.02
NB TF-IDF	2.0	36.23	35.73	32.62
Log TF-IDF		24.17	23.46	19.51
RF	2.0	**43.03**	**41.02**	**37.23**
DT TF-IDF	1.0	35.34	32.57	27.07
BM25		1.76	2.15	3.59
HiCal		**44.95**	**42.38**	**38.32**

TF-IDF = term frequency-inverse document frequency;
U-Rate = undersampling distribution rate;
(1) k=20 l=50 m=6 Teamdraft Interleaving;
(2) k=100 l=100 m=10 Teamdraft Interleaving;
NB = Naive Bayes; Log = logistic regression;
RF = random forest; DT = decision table

Table 4.9: Machine-Learning and Keyquery Runs on genome TREC

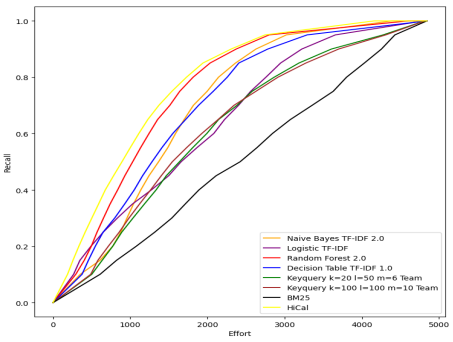

Figure 4.9: Machine-Learning and Keyquery Runs on genome TREC

achieved a significantly worse performance than HiCals algorithm. We expect that this difference is due to different implementation details and possibly more fine-tuned parameters on HiCal. Our naive Bayes approach and random forest approach have a similar slope in the beginning as HiCal but their performance decreases faster. The worse performance may be to the simplicity of naive Bayes and lack of parameter fine-tuning for random forest. Also, different features can have a significant impact on the results as normalization or stemming. Our decision table's performance improves a lot by undersampling especially when the amount of relevant and irrelevant documents are balanced. With other features and different parameter settings, the performance might improve.

A comparison between our kequery-based approach and our machine-learning based approaches has shown that the slope of the machine-learning based approaches are better in the beginning. This is because the keyquery-based approach can create keyqueries only with many relevant documents. Pseudo-relevance feedback are documents that were predicted relevant by our machine-learning approaches. We assumed that pseudo-relevance feedback provided by our machine-learning approaches could help to improve our keyquery-based approach. In fact, the slope was higher than before in the beginning but decreased faster. If we were able to improve our machine-learning-based approaches, they would be able to give better predictions on documents. Then fewer non-relevant documents would be considered as relevant and taken into consideration for building keyqueries.

The most difficult dataset was genome TREC. All approaches performed worse on this data set than on CEEDER-2018 and CEEDER-2019. This may be because relevant documents on genome TREC do not share much content.

Furthermore our keyquery approach does not use the initial query as relevant document, which the machine-learning approaches do. A test on the machine-learning approaches show that the use of this query improves the performance slightly. If the keyquery approach would use this query, it might improve as well.

Chapter 5

Future Work and Conclusion

In this section we outline possible improvements to our machine-learning based approaches and our keyquery-based approach. After that we summarize the main aspects of this work and draw a conclusion.

5.1 Future Work

First we look into methods how to improve the performance of the machine learning approaches. One might create an ensemble of existing approaches like using an ensemble of support vector machines [22] or other ensembles which performed best among the tested approaches [14].

For our machine-learning-based approaches we tested undersampling and several kinds of feature selection like (1) boolean word occurrence, (2) tf and (3) tf-idf to improve them. There is still a vast range of parameters to be tested for our machine-learning based approaches. One could use normalization or a lemmatization. In addition we left all parameters for the machine-learning models at default, the performance can be improved by doing a grid search on all possible parameters for the different models. Due to the fact, that our logistic regression approach is worse than the current state-of-the-art approach, called HiCal, which also uses a logistic regression model, one could also study the source code of the state-of-the-art approach to find the features they are using. Then our logistic regression approach should be as good as HiCal.

After looking into ways of improving our machine-learning-based approaches we will now look into ways of improving our keyquery-based approach. In the evaluation we shortly discussed if the removal of redundant keyqueries improves the performance. Redundant keyqueries are keyqueries that do not provide more information than another keyquery. Evaluation showed that the removal of the redundant keyqueries as implemented now deteriorated the results. If we do further research on the removal of redundant keyqueries it

could improve the performance of the keyquery approach. Furthermore one could use the non redundant keyqueries for a single boolean query. With this boolean query the retrieval system could possibly give a better result than merging the results of each keyquery because the scores on documents of keyqueries can be deficient.

A second idea of improving the keyquery-based approach is to use more and possibly better pseudo-relevance feedback. Pseudo-relevance feedback are documents predicted as relevant by our machine-learning approaches and used by our keyquery approach to generate keyqueries. If our machine-learning-based approaches improved, then our keyquery-based approach could create keyqueries only for relevant documents and hence improve its performance.

Another interesting approach for future work is the usage of context-dependent term weights, which in experiments had better rankings in ad-hoc web search than tf and BM25 [5]. The score of a context-dependent term weight states the importance of a term in its context. One advantage is that these scores can be directly used in a inverted index of a search engine and hence can be used out-of-the-box for our keyquery-based approach and also for our machine-learning-based approach as another kind of feature. We can see that there is more to achieve in later work on technology-assisted review creation with the help of keyqueries.

5.2 Conclusion

In this thesis, we tackled the task of technology-assisted review creating via keyqueries. Due to the importance of systematic reviews and the manual effort involved in creating a systematic review, it is necessary to provide modern technological systems that can reduce the time of screening documents for systematic reviews. There is no modern system that can provide all relevant documents for a research topic at once. Therefore, scientists created a framework that learns from user feedback that documents are relevant and can provide more relevant documents.

Given a corpus of potential relevant documents, users evaluate the top documents from this corpus and provide feedback on the seen documents. A technology-assisting system uses this feedback to rank the corpus and tries to put relevant documents in the top-results. Then the users evaluate these top results again and the technology-assisting system can maybe provide a better ranking with the additional feedback.

We first developed machine-learning approaches as technology-assisting system that ranks the corpus. These machine-learning-based approaches use machine-learning models to predict if a document is relevant. However, our

primary focus lay on so-called keyqueries. Keyqueries are queries that, when given to a search engine, retrieve the identified relevant documents in the top results. The hypothesis is that the additional top results that are not the identified relevant documents are likely to be relevant.

In our experimental evaluation we used three simulated systematic reviews, to compare our machine-learning based and keyquery-based approaches against themselves, against the current state-of-the-art system for technology-assisted reviews, called HiCal and against BM25.

All of our approaches were able to outperform the BM25 ranking. Our naive Bayes approach and random forest approach are coming close to the performance of HiCal. Our logistic regression approach performed significantly worse than HiCal, which is interesting because both use a logistic regression model for predictions. Our decision table performed slightly better than our logistic regression approach. Interestingly, feature selection on our machine-learning approaches did not have a significant impact on their performance.

Our keyquery approach performed worse than HiCal and our naive Bayes approach and performed similar to our random forest approach on the data sets of the Julius-Kühn-Institut, namely CEEDER-2018 and CEEDER-2019. On the last data set, namely genome TREC, the performance of our keyquery-based approach decreased significantly, such that they performed worse than all other approaches except BM25.

Hence we conclude that further research regarding technology-assisted reviews with keyqueries needs to be done. They can be easily interpreted due to the keyquery criteria and the results are easier replicable than machine-learning models. Therefore they are useful for technology-assisted reviews.

Bibliography

[1] A. Alharbi and M. Stevenson. Ranking abstracts to identify relevant evidence for systematic reviews: The university of sheffield's approach to clef ehealth 2017 task 2. 2017. 2

[2] A. M. Cohen, W. R. Hersh, K. Peterson, and P.-Y. Yen. Reducing Workload in Systematic Review Preparation Using Automated Citation Classification. *Journal of the American Medical Informatics Association*, 13(2):206–219, 03 2006. 4.1

[3] G. Cormack and M. Grossman. Technology-assisted review in empirical medicine: Waterloo participation in clef ehealth 2018. In *CLEF (Working Notes)*, 2018. 4.1

[4] G. Cormack and M. R. Grossman. Autonomy and reliability of continuous active learning for technology-assisted review. *ArXiv*, abs/1504.06868, 2015. 2

[5] Z. Dai and J. Callan. Context-aware document term weighting for ad-hoc search. *Proceedings of The Web Conference 2020*, 2020. 5.1

[6] A. Dasdan, P. D. Alberto, S. Kolay, and C. Drome. Automatic retrieval of similar content using search engine query interface. *Proceedings of the 18th ACM conference on Information and knowledge management*, 2009. 2.1

[7] N. Ghelani, G. Cormack, and M. Smucker. Refresh strategies in continuous active learning. In *ProfS/KG4IR/Data:Search@SIGIR*, 2018. 2

[8] B. Golshan, T. Lappas, and E. Terzi. Sofia search: a tool for automating related-work search. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012. 2.1

[9] D. Gough, S. Oliver, and J. Thomas. An introduction to systematic reviews. 2017. 3

[10] M. R. Grossman, G. Cormack, and A. Roegiest. Trec 2015 total recall track overview. In *TREC*, 2015. 2

[11] M. R. Grossman, G. Cormack, and A. Roegiest. Trec 2016 total recall track overview. In *TREC*, 2016. 2

[12] M. Hagen, A. Beyer, T. Gollub, K. Komlossy, and B. Stein. Supporting scholarly search with keyqueries. In *ECIR*, 2016. 1, 2.1, 3.2

[13] K. Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 1. *Inf. Process. Manag.*, 36:779–808, 2000. 4.1

[14] T. Lange, G. Schwarzer, T. Datzmann, and H. Binder. Machine learning for identifying relevant publications in updates of systematic reviews of diagnostic test studies. *medRxiv*, 2020. 2.1, 4.2.1, 5.1

[15] D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR 94*, 1994. 1

[16] L. McKeever, V. Nguyen, S. Peterson, S. Gomez-Perez, and C. Braunschweig. Demystifying the search button. *JPEN. Journal of Parenteral and Enteral Nutrition*, 39:622 – 635, 2015. 2

[17] M. Michelson and K. Reuter. The significant cost of systematic reviews and meta-analyses: A call for greater involvement of machine learning to assess the promise of clinical trials. *Contemporary Clinical Trials Communications*, 16, 2019. 1

[18] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*, 2008. 3.3

[19] H. Scells, G. Zuccon, and B. Koopman. Automatic boolean query refinement for systematic review literature search. *The World Wide Web Conference*, 2019. 2

[20] G. Surita, R. Nogueira, and R. Lotufo. Can questions summarize a corpus? using question generation for characterizing covid-19 research. *ArXiv*, abs/2009.09290, 2020. 2

[21] van de Schoot R. An open source machine learning framework for efficient and transparent systematic reviews. *Nat Mach Intell*, 3:125–133, 2021. 2, 2.1

[22] B. C. Wallace, T. Trikalinos, J. Lau, C. Brodley, and C. Schmid. Semi-automated screening of biomedical citations for systeamticreviews. *BMC Bioinformatics*, 11:55–55, 2009. 5.1

[23] H. Zhang, M. Abualsaud, N. Ghelani, A. Ghosh, M. Smucker, G. Cormack, and M. R. Grossman. Uwaterloomds at the trec 2017 common core track. In *TREC*, 2017. 2

[24] H. Zhang, M. Abualsaud, N. Ghelani, M. Smucker, G. Cormack, and M. R. Grossman. Effective user interaction for high-recall retrieval: Less is more. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018. 2

[25] H. Zhang, G. Cormack, M. R. Grossman, and M. Smucker. Evaluating sentence-level relevance feedback for high-recall information retrieval. *Information Retrieval Journal*, 23:1–26, 2020. 2

[26] H.-T. Zhang. Increasing the efficiency of high-recall information retrieval. 2019. 2

[27] J. Zou and E. Kanoulas. Towards question-based high-recall information retrieval. *ACM Transactions on Information Systems (TOIS)*, 38:1 – 35, 2020. 2, 2.1

[28] J. Zou, D. Li, and E. Kanoulas. Technology assisted reviews: Finding the last few relevant documents by asking yes/no questions to reviewers. *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018. 2