Universiteit Leiden

ICT in Business and the Public Sector

Using relevance feedback and text similarity to reduce review effort in eDiscovery

Name: Timo Kats
Student-no: s2012138

Date: 03/07/2023

1st supervisor: Dr. P.W.H van der Putten
2nd supervisor: Prof. Dr. Ir. J.C. Scholtes

Company supervisor: L. Jean-Louis PhD
Company supervisor: Z. Gerolemou MSc

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The manual review process is the biggest cost driver in electronic discovery (eDiscovery). This is driven by high review effort, which refers to the amount of documents that need to be reviewed manually. Most approaches designed to reduce this review effort implement an active learning based system that leverages user feedback to exclude non-relevant documents faster. In eDiscovery this is also referred to as "Technology Assisted Review" (TAR). Although this method is effective at reducing review effort, it also has a possibility of creating false negatives without the awareness of the user. As a result, this research proposes a different strategy to reducing review effort. More specifically, through iteratively re-ranking the relevance rankings based on user feedback, which is also referred to as relevance feedback. In our proposed method for this, the relevance rankings are produced by a BERT based dense vector search and the relevance feedback is based on cumulatively summing the queried and selected embeddings. Our results show that this method can reduce review effort between 17.85% and 59.04%.

Contents

1 Introduction

Prior to lawsuits and investigations there is a discovery process during which each party is required to produce evidence, in the form of information/documents/e-mails, to the other party [44]. In 2006 this process was extended by the Federal Rules of Civil Procedure (FRCP) to all electronically stored information (ESI), which gave rise to the field of electronic discovery (eDiscovery) [42].

The field of eDiscovery refers to any process where ESI is collected, located and searched with the intent of using it as evidence in lawsuits or investigations [14]. Generally, this ESI is a heterogeneous collection of emails, office documents (e.g. word files, excel spreadsheets, etc.), databases, and other forms of data (e.g. source code or voice mails) [17].

In a typical eDiscovery scenario, a set of documents is collected from ESI based on a list of custodians (i.e. people that have administrative control over a piece of data) or a date range. Thereafter, this set of collected documents is manually reviewed by a legal team in order to find documents that could be used in a lawsuit or investigation.

The costs of eDiscovery has grown over the years. In fact, the costs associated with the eDiscovery process sometimes exceed the amount in controversy during legal disputes [32]. The biggest cost driver here is the manual reviewing process, which has been estimated to account for "90% of all eDiscovery costs" [55]. These high review costs are driven by high review effort [52], which refers to the amount of documents that have to be manually reviewed by a human annotator [15].

Given these figures, different strategies aimed at reducing manual review effort have been proposed in the "Text Retrieval Conference" (TREC) [59]. By and large, these strategies focused on implementing active-learning based methods that speed up the reviewing process through excluding non-responsive documents (i.e. documents that are not relevant to a lawsuit or investigation) faster based on user feedback. In eDiscovery this strategy is generally referred to as Technology Assisted Reviewing (TAR) [27].

1.1 Problem statement

Despite being effective at reducing review effort, the usage of TAR in eDiscovery does have a limitation. Namely, the possibility of false negatives. In eDiscovery this is particularly problematic, since any false negative might "not just be relevant, but crucial" to a lawsuit or investigation [23]. TAR is particularly prone to this issue, since the active-learning based system can create false negatives automatically without the explicit awareness of the user.

Also, TAR is typically applied after a deduplication process [27]. This process is effective at removing exact duplicates, but not near duplicates. As a result, review effort can be spent on near-duplicate documents that are practically similar regarding their textual contents.

Given these shortcomings, the goal of this research is to evaluate a different approach for reducing review effort. For this, we combine two major components: textual similarity and relevance feedback. Here, the textual similarity is used to produce relevance rankings, whilst relevance feedback is used to iteratively improve those relevance rankings based on user feedback. For the textual similarity component, we evaluate similarity methods that are based on Quorum search, TF-IDF and BERT. Next, for the relevance feedback component, we compare text based and vector based feedback strategies.

1.2 Research question

Given the limitations of the current TAR process identified in the previous subsection, we formulate the research questions stated below. The first research question relates to selecting the most suitable text similarity approach for the relevance rankings. Next, the second research question relates implementing that approach using different relevance feedback strategies with the goal of reducing review effort.

- RQ1: Can text similarity be used for relevance feedback?
- RQ2: To what extent can relevance feedback help to reduce review effort?

1.3 Thesis outline

The remainder of this thesis is structured as follows. In section 2 we provide a summary of works related to this research. This consists of works related to the domain of eDiscovery, text similarity methods, and relevance feedback. Next, we describe the methodology and experimental setup of this research in section 3 and section 4. Here, we first discuss the text similarity methods used for the relevance rankings. Next, we discuss the relevance feedback strategies used to implement the identified text similarity method. In section 5 we share the results of these experiments. Thereafter, in section 6, we interpret these results and discuss the limitations of the experiment. Based on these limitations we share our suggestions for future work. Finally, in the conclusion section we answer the research questions stated earlier.

2 Background

In this section we provide an overview of the literature related to this research. First, we discuss the eDiscovery process and how it's implemented in different legal systems. Secondly, we discuss literature related to the text similarity methods used in our experiment. Next, we build on that through discussing literature related to the usage of those text similarity methods in eDiscovery. Thereafter, we review other methods that reduce review effort and the role of machine learning in this. Finally, we discuss different relevance feedback strategies related to our research.

2.1 eDiscovery in different legal systems

The discovery process is the exchange of information between two parties prior to lawsuits and investigations. The field of eDiscovery refers to this process, but with the inclusion of electronically stored information [42]. The implementation of eDiscovery differs per legal system [67]. This subsection provides an overview of these legal systems and their differences regarding the role of eDiscovery.

2.1.1 Adversarial legal systems

The adversarial system is specific to common law, which is generally practiced in Anglo-centric countries like the United States, Canada and the United Kingdom. Here, the court (often with a jury) acts as a referee between the two parties in a legal dispute. Therefore, the parties are responsible for preparing and presenting their own case in court [56].

As a result, the parties in a legal dispute must collect evidence from each other [5]. The extent to which evidence is collected depends on the country. For example, in the United States the extent to which a party is allowed to ask for evidence from another party extends to anything relevant to the case which is non-privileged [42]. So for example, attorney-client privileged information is excluded from this discovery process. This legal format results in a recall oriented eDiscovery process where large sets of evidence are considered in court [30]. As a result, an approach to reducing review effort should be recall oriented when applied in an adversarial legal system. Considering relevance feedback has been categorized as "High-Recall-Information-Retrieval" (HRIR) [68], our approach is favorable in this legal system.

2.1.2 Inquisitional legal systems

The inquisitional legal system is specific to civil law, which is practiced in many (non-Anglo) countries. For example, Germany, France and The Netherlands. Here, in contrast to the adversarial legal system, the judge acts as an investigator instead of a referee [56].

As a result, this eDiscovery process tends to be less recall oriented than in the adversarial legal system [30]. Hence, the added value of a high recall approach (like relevance feedback) is lessened in this legal system.

2.2 Text similarity methods and techniques

The text similarity methods used in this research stem from different approaches to computing textual similarity. More specifically, our TF-IDF and Quorum based similarity approaches have a background in term based similarity methods, whereas our BERT based approach is a context based text similarity method. As a result, this subsection provides an overview of the related works in these categories.

2.2.1 Term based similarity methods

Term based similarity methods compute text similarity through taking the common features between two pieces of text into account [12]. As a result, these methods don't incorporate contextual language properties like homonymy or synonymy in their similarity computations.

A simple (and commonly used [34]) implementation of term based similarity is Jaccard similarity [33], which computes text similarity based on how many features two texts have in common divided by the total number of features across both texts. The implementation of this similarity method is based on set algebra (using the intersection and the union operators). The formula for this is given below. Here, A and B refer to the set of unique words for both documents.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B|} \tag{1}$$

A disadvantage of Jaccard is how it's impacted by the size of the data. Computing the intersection between two texts (without optimization) amounts to quadratic ($O(n^2)$) time complexity [19]. Moreover, properties like word frequency can't be included when simplifying texts with set algebra. As a result, we don't use this approach in our experiments for basic term based text similarity. Instead, we use Quorum search [6].

Quorum search is used to return documents that contain a specific number of terms (see section 3.1.1). It complements Jaccard through sorting the words based on frequency. As a result, their position in the sorted list contains more information than only occurrence. Moreover, it uses an index to return documents that contain specific terms, which is a datastructure that allows for faster searching through mapping words to locations (e.g. documents) [40]. As a result, this method has linear time complexity ($O(n)$) [6] and therefore scales better than Jaccard similarity.

However, a shortcoming of Quorum search is that frequent terms within a document are unlikely to be distinctive or important [10]. To deal with this shortcoming, there are two approaches. The first approach is based on manually filtering out frequent words based on a pre-defined list of words that are known to be common in a given language. These words are often referred to as "stop words". Removing stop words when searching for textually similar documents tends to have a beneficial effect [10]. Hence, we filter out stopwords in our experiment.

The second approach is based on diminishing the value of words that are common in a corpus [10]. An advantage of this method compared to filtering stopwords is that it's more dynamic. Meaning, certain words that are common within a specific context (e.g. the word "patient" in medical data) might not be included in pre-defined lists of stopwords.

This approach is implemented in our TF-IDF based similarity method (see section 3.1.3) through the *inverse document frequency*. Here, terms are given a measure of uniqueness by dividing the amount of documents in total by the amount of documents that have a specific term. The formula for this metric is given below. Here, D refers to the number of documents in the dataset whereas d refers to the number of documents that contain term t. As a result, terms that appear in many documents (and are therefore less unique) are given a diminished value [53].

$$idf(t, D) = log \frac{|D|}{1 + |\{d \in D : t \in d\}|} \tag{2}$$

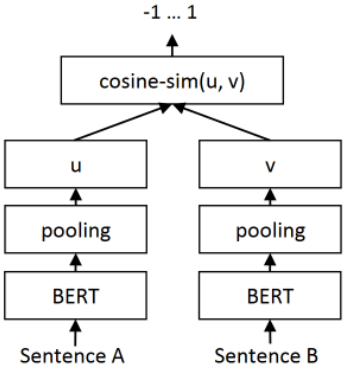


Figure 1: Simplified architecture of SBERT [48]

2.2.2 Context based similarity methods

In this research we also use context based similarity methods. In contrast to the term based similarity methods mentioned previously, these methods do incorporate a form of semantic meaning. This is based on the "distributional hypothesis" [25], which is built on the idea that "words that occur in the same contexts tend to have similar meanings" [12]. Given this hypothesis, this research uses pre-trained word embeddings that provide vector-based representations of texts. This enables us to compute the textual similarity based on the similarity between the vectors (e.g. with Cosine similarity) [12].

In this category of pre-trained word embeddings we are specifically focused on BERT (short for "Bidirectional Encoder Representations from Transformers") [21], which forms the basis of one of the similarity methods that this research explores (see section 3.1.4). BERT is based on transformers, which is a deep learning model based on a neural network architecture [63]. This architecture is similar to recurrent neural networks (RNNs) in the sense that both can be used to detect patterns in a sequence of data [54].

However, in contrast to RNNs, transformers can process complete sequences without needing multiple iterations. As a consequence, this architecture allows BERT to capture the semantic meaning of a word in a sequence based on the terms that surround it. This is accomplished during the pre-training and fine-tuning steps of the BERT framework, which consists of two tasks [21]. First, the "masked language model", which is to hide (i.e. mask) words in a sentence and predict their value. Second, "next sentence prediction", which is to predict whether two sentences have a logical/sequential connection.

The fine-tuning step of BERT can be done on domain specific data, which makes it possible to create custom versions of BERT for specific tasks. Some BERT versions that are related to the topic of this research are LEGAL-BERT [11] (which was trained on legal texts like court cases, contracts, legislation, etc.) and Goldilocks (which is optimized for technology assisted reviewing in eDiscovery [66]). However, despite being related to the topic of this research, these BERT models are not optimized for text similarity tasks. Hence, we don't use these BERT models in our research.

Sentence-BERT (SBERT) on the other hand is a BERT model specifically made for measuring text similarity using the cosine distance metric [48]. The key advantage SBERT has over the other BERT models for text similarity tasks is its reduction in computational overhead. The researchers found that for finding the most similar pairs in a collection of 10.000 sentences, BERT would take ≈65 hours whereas SBERT would take 5 seconds. Moreover, it enables a similarity

search between larger bodies of text (like sentences and paragraphs) through mean-pooling the word embeddings.

A simplified overview of this architecture can be found in Figure 1. Here, the (word level) BERT embeddings are first mean-pooled to create paragraph/sentence level embeddings. Thereafter, these embeddings are compared through their cosine similarity. Note, for the mean-pooling process, text that exceeds 384 words in length is truncated [48]. As a result, this approach can't be implemented on a level of text granularity that exceeds this amount.

Given these advantages, we will use SBERT in our experiments to compute text similarity. Note, a potential alternative to SBERT could be tBERT [43], which is another BERT model specifically made for text similarity tasks. However, this BERT model is also topic-informed, which makes it more fit for domain specific cases. Hence we will not use tBERT in this research.

2.3 Usage of text similarity methods in eDiscovery

In eDiscovery – besides custodian lists, keywords or date ranges – documents can be used as queries [18]. This is referred to as "Query by Document" (QBD) and is the implementation of the text similarity methods in this research. QBD has some advantages over traditional queries. First, it nullifies the effort that comes with constructing a traditional query. Second, it can return documents that traditional queries can miss [18]. Note, the "richness" of a document (i.e. the prevalence of relevant terms in a queried document) [65] has a positive impact on the effectiveness of this technique.

Next, the text similarity methods mentioned in the previous section can also be used for near-duplicate detection in eDiscovery. This is different from our research, which focuses on finding relevant documents instead of near duplicate documents. Still, near duplicate detection is an important field in eDiscovery. In fact, the number of similar documents in eDiscovery corpora typically ranges between 25%-50% [58]. As a result, near duplicate detection can significantly reduce the size of these corpora through filtering out duplicate documents.

2.4 Methods for reducing eDiscovery review effort

This section provides an overview of other strategies that aim to reduce review effort currently used in eDiscovery.

2.4.1 Technology assisted reviewing

In eDiscovery, review effort is often reduced using some form of automation. Typically, this process is not fully automated, since the usage of automated and manual reviewing is generally viewed as a trade-off where humans are more accurate and computers are more efficient [41]. As a result, it's common to establish a synergy between automated and manual reviewing. In eDiscovery, this is often referred to as "technology assisted reviewing" (TAR), predictive coding or predictive ranking [60]. In this research we refer to this process as TAR.

In TAR, the computer takes the input from the user (who labels a document as relevant or non-relevant) and uses it to automatically label other documents [27]. Because TAR leverages user input, the system tends to become better at suggesting/labeling documents as the process progresses [60].

Currently, there are three major protocols for TAR [27]. First, the CAL (continuous active learning) protocol [28], which is generally seen as the most effective at reducing review effort [16]. In summary, CAL uses the input from the initial seed set and the input from the continuous reviewing from users to select the "next most likely relevant" document for review until no more relevant documents can be found. A variation on this process can be found in MINECORE,

Protocol	Seed set	Stopping criterion
CAL	Judgmental sampling	Enough documents found (e.g. measured in drop-off precision)
SAL	Judgmental sampling	Classifier good enough (e.g. measured in precision, recall or F1)
SPL	Random sampling	Classifier good enough (e.g. measured in precision, recall or F1)

Table 1: Different TAR protocols (sorted from least to most review effort)

which stands for "minimizing the expected costs of review" [41]. Here, the user is presented documents for which the classifier is least certain.

Second, in the SPL (simple passive learning) protocol [28], the (human) reviewer pre-selects documents as training examples. These documents are used to train the model. Once adequate training has been achieved, this model is used to label every document. Generally, this labeling is re-reviewed manually. Finally, in the SAL (simple active learning) protocol [28], documents are selected and labeled by a (human) reviewer. This labeling is then used to train the model until it's adequately trained. Generally, the model pre-selects documents for labeling that it's least certain about. Both SPL and SAL are referred to as TAR 1.0, and have been found less effective at reducing review effort than CAL [60]. Note, a tabular summary of these TAR protocols (sorted by review effort) can be found in Table 1.

Still, regardless of the TAR protocol, review effort is reduced through excluding documents based on user feedback. As a result, the creation of false negatives without the awareness of the user remains an issue. In eDiscovery this is particularly problematic, since any false negative might be crucial in an investigation or lawsuit [23]. This problem is augmented in an adversarial legal system, since this discovery process is more recall oriented [30]. As a result, our approach to reducing review effort uses user input to *re-rank* documents instead of *exclude* them.

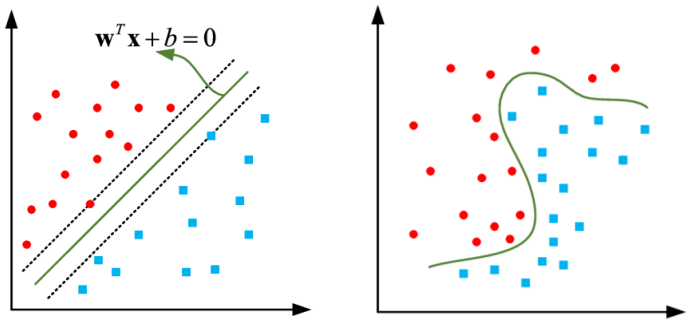
2.4.2 Clustering search results

After collection, instead of reviewing search results one at a time, search results can also be reviewed in clusters. These clusters are based on the similarities between the documents and can be reviewed through a representative document (also referred to as a centroid). As a result, multiple documents can be (dis)approved in one action, which reduces manual review effort [22].

IBM [34] used this technique to group/cluster e-mails together in their "eDiscovery Analyzer". This was implemented in the review stage of the eDiscovery process by showing semantically similar search results in groups, making it possible to (dis)approve multiple search results at once. According to IBM, the usage of text similarity in this setting increased the precision of the review process from 0.62 to 0.91. Thus, the text similarity methods identified in this research could be implemented in a similar fashion.

2.4.3 Topic Modeling

Topic modeling is a collection of methods that reduce large collections of text into a smaller subspace of topics/clusters [13]. An example of such a method is non-negative matrix factorization (NMF). This method has been used in combination with TF-IDF embeddings to compute the dissimilarity between the relevant documents in the seed set, and the remaining documents in the dataset [45]. The philosophy behind this method is that documents that are very dissimilar to the relevant documents in the seed set are likely to be non-relevant. As a result, these documents can already be labeled as such. According to the researchers this strategy reduces the manual review effort between 10.6% and 96.9%. Because this research used the same dataset as our research (the RCV-1 v2 news corpus [3]), we use these results to contextualize the reduction of review effort from our proposed method.



(a) Linear-SVM (b) Kernel-SVM

Figure 2: Illustration of different types of SVMs from [29]

2.4.4 Supervised machine learning

The implementations of the previously mentioned TAR protocols are generally based on supervised machine learning, often through support vector machines (SVMs) [45]. Originally introduced by Vladimir Vapnik in 1995 [62], SVMs are a form of supervised machine learning that classify data points in two (or more) classes through fitting a hyperplane between them. This hyperplane is fitted based on a "convex optimization problem", which strives to maximize the distances between the hyperplane and the data points it separates. The shape of this hyperplane can be of two categories: linear-SVMs (straight hyperplane) or kernel-SVMs (flexible hyperplane). An illustration of the differences between these two types of SVMs can be found in Figure 2. For both variants, the SVM requires labeled data to be trained on. In TAR, the manual labeling of data points is typically used for this [69] (e.g. through the pre-selected data points in the SPL protocol)

In eDiscovery the data points are often based on word embeddings. For example, a commonly used (SVM based) method to classify text in eDiscovery is a linear-SVM based on TF-IDF embeddings [64]. In fact, a comparison of popular machine learning algorithms in eDiscovery found that this approach outperforms other commonly used approaches, like bag-of-words based linear-SVMs or TF-IDF based logistic regression [64]. Hence, in order to contextualize our results in the broader domain of eDiscovery, we also use this method as a baseline approach.

2.5 Relevance feedback

Relevance rankings in eDiscovery refers to the order in which documents are returned [57]. In this research we are particularly interested in relevance feedback, which refers to changing the relevance ranking through user feedback, generally in multiple iterations [40].

In eDiscovery, the usage of relevance feedback has been found particularly useful for "High-Recall-Information-Retrieval" (HRIR) [68]. This is due to the fact that in relevance feedback documents only get *re-ranked* instead of *excluded* (as is the case in TAR). As a result, this approach to reducing review effort is particularly useful in adversarial legal systems [30].

Note, a variation of relevance feedback frequently used in research (since it's not always viable to give manual feedback in experiments) is *pseudo-relevance feedback*. In pseudo relevance feedback, the feedback is based on automatic analysis. For example, through assuming that only the top n documents are relevant, or through assuming that only search results with a similar annotation/label are relevant [40].

As for implementing relevance feedback, there are two major categories of relevance feedback strategies. [35]. First, text based relevance feedback strategies. Second, vector based relevance feedback strategies. Both of these strategies are implemented in our experiments.

2.5.1 Text based strategies

A commonly used text based relevance feedback strategy is "keyword expansion". Here, keywords from the selected search results are appended to the query. There are two main variations of this method [9]. The first variation is to select terms that co-occur with terms from the original query. The philosophy behind this approach is that terms that co-occur with terms from the original query are more likely to be relevant to the search. The formula to calculate this feature is shown below. Note, in this formula $C(t_i, e|D)$ refers to the frequency a query term t_i co-occurs with expansion term e. By default, a co-occurence is defined through a range of 12 words. Also, the formula corrects co-occurence with (over)frequent terms through dividing the co-occurrences with a specific expansion term by the total frequency (tf) of that expansion term.

$$f(e) = log\frac{1}{n}\Sigma_{i=1}^{n}\frac{\Sigma_{d\in F}\ C(t_i, e|D)}{\Sigma_t\ \Sigma_{d\in F}\ tf(t, D)} \tag{3}$$

Another keyword-expansion variation for relevance feedback is based on the *inverse document frequency* mentioned earlier in this section. Here, the underlying assumption is that more unique words are more valuable to the search. In this thesis we will use this strategy for selecting keywords to expand our queries with.

2.5.2 Vector based strategies

Assuming vector representations of the query and search results are available, user feedback can also be applied to the queried vector directly. This is referred to as vector based pseudo relevance feedback. In general, there are two commonly used methods for this [35]. First, the queried vector can be averaged with the positive search results. Second, the queried vector can be summed with the selected search results. Both of these strategies are implemented in our research.

A commonly variation of averaging query vectors is based on Rocchio's method for relevance feedback [35]. Originally invented by Joseph John Rocchio in 1966, the high-level idea of this method is to move the query vector towards the selected vectors through assigning different weights to selected and queried vectors [50]. The version of Rocchio implemented by most researchers today [8] [4] differs slightly from the original method, since it omits the negative feedback (i.e. non-selected documents) from the formula. As a result, this version of Rocchio can be seen as a weighted average between the (original) queried embedding and the (averaged) selected embeddings.

The weight of the queried embedding (α) and the weight of the averaged selected embeddings (β) can be set by a user. Still, the default/consensus values most research adheres to is $\alpha = 0.5$ and $\beta = 0.5$ [4]. Hence, we use Rocchio with those parameter values as a baseline method in our experiment.

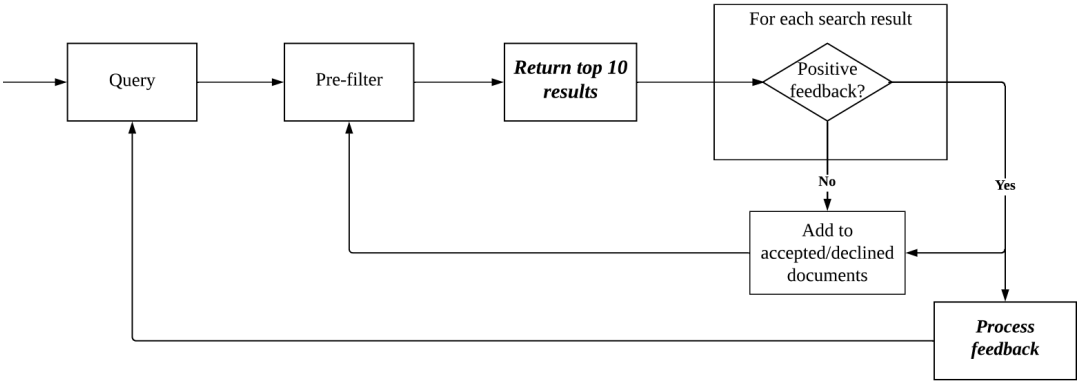


Figure 3: Flowchart of the method

3 Methodology

In this research we evaluate different relevance feedback strategies and text similarity methods with the objective of reducing review effort. As shown in Figure 3, the method for accomplishing this is based on iteratively presenting the user with a set of (10) results to accept or decline. Thereafter, the accepted documents are used to improve the query (and consequently results) for the next iteration.

For returning the results (i.e relevance ranking) we experiment with different text similarity methods, levels of textual granularity, and ranking methods. Moreover, we compare the performance of these text similarity methods with a supervised approach to contextualize their performance. These methods are explained in the first and second subsection. Next, for processing the feedback given after each iteration, we experiment with different feedback strategies. These feedback strategies are explained in the third subsection.

Note, a common denominator of all our experiments is the usage of a search engine. In our case, this search engine is based on an inverted index at the document level (using unique document identifiers). This is also referred to as a "document index".

3.1 Text similarity methods

In this research we have two baseline methods. The first baseline method is a basic text similarity method (Quorum operator). The second baseline method is a supervised machine learning approach commonly used in eDiscovery for similar classification tasks (TF-IDF based linear-SVM). The purpose of this method is to contextualize the results of our text similarity methods within the broader domain of eDiscovery. Next, our second and third similarity methods are based on TF-IDF and dense vector search.

3.1.1 Quorum search

The Quorum operator is used to search for documents that contain a specific number (M) of terms [6]. For example, given the search terms {Foo, Bar} where $M = 1$, a document should at least contain one of these terms for the quorum operator to return true. This search operation is conducted on a document's N most common terms.

As a result, the Quorum operator requires two parameters. First, N changes the amount of words considered for comparison (expressed as a percentage of the *queried* document's size).

Algorithm 1: Quorum search

```
input  : query, documentIndex
output: Number of matching terms between queried article and target articles.
matches ← emptyMatches()
for term in query do
    for documentId in documentIndex[term] do
        matches[documentId] += 1            // Docs that have term from query.
return matches
```

Increasing this parameter will increase the recall of the quorum operator. Second, M changes the amount of terms that need to match between a query and a document (expressed as a percentage of N). Increasing this parameter will increase the precision of the quorum operator. Given the meaning of these parameters, $M \leq N$. Else, there won't be enough terms in the documents to achieve the number of required matches.

Note, since the Quorum operator isn't available in any public libraries, we implemented our own version. The pseudo-code for this implementation can be found in Algorithm 1. Here, the document index and query are already pre-processed based on the documents' N most frequent terms. Also, the threshold M will be applied to the returned values after this function. Finally, the results are sorted based on the matches divided by document length.

3.1.2 SVM based approach

Our second baseline method is a linear-SVM based on TF-IDF embeddings. Note, in contrast to the other methods in this section, this is a supervised approach. Meaning, it's first trained on annotated data. As a result, this approach differs from our other approaches and is exclusively meant to contextualize the results of our other similarity methods.

First, TF-IDF [49] vectorizes text based on multiplying the frequency of a word in a document with the *inverse* frequency of that word in the entire data set (formula for this is given below). Hence, TF-IDF looks at the "uniqueness" of a word instead of just the frequency. The result of TF-IDF is a sparse vector where all the words that exist in a document are given their TF-IDF value (words that don't appear in a specific document have zero values). The formula for computing the TF-IDF value for a term is given below.

$$TF = \frac{\text{term frequency in document}}{\text{total words in document}}$$

$$IDF(t) = \log_2\left(\frac{\text{total documents in data set}}{\text{documents with term}}\right)$$

$$\text{TF-IDF score for term t in data set} = \text{TF} \cdot \text{IDF(t)}$$

Next, these embeddings are used to train a Linear-SVM. Meaning, the data-points that the Linear-SVM aims to separate are based on the TF-IDF embeddings of the texts in the dataset. For this, we use a 60/40 train/test split. The classes of these data-points *and* the data used for this method will be explained in the experimental setup.

3.1.3 TF-IDF based approach

Our second text similarity method is based on TF-IDF, which makes it a statistical approach to finding similar texts. As exemplified in the previous subsection, TF-IDF builds on Quorum search through multiplying the frequency of a term in a document with a measure of uniqueness based on the entire dataset. Hence, computing the similarity between two texts based on TF-IDF captures more information than Quorum search.

In order to use TF-IDF to find similar documents, we use MoreLikeThis (MLT). In summary, MLT queries the terms from a document (that have the highest TF-IDF values) individually using the document index. The documents returned by these queries are ranked based on combining their MLT scores, which is defined as the sum of the TF-IDF values of the matching terms between the queried document and the returned document [24].

Note, we are aware that a commonly used implementation of TF-IDF to identify textually similar texts is based on computing the cosine distance between the TF-IDF vectors. However, computing the Cosine distance between n vectors results in quadratic time and space complexity ($O(n^2)$). Hence, this method is not feasible in a real-world eDiscovery scenario. As a result, we use MLT instead.

3.1.4 Dense vector search

Our third similarity method is based on Sentence-BERT (SBERT) embeddings. These embeddings can be used to find similar texts using "dense vector search" (DVS). The distance metric used for this is cosine similarity, which computes the similarity between two vectors (i.e. embeddings) based on the cosine value of the angle between the two vectors [46]. This angle is computed through dividing the dot-product (which is the sum of products from both vectors) by the length of the vectors. This means that the potential values of this similarity measure range from -1 (completely opposite) to 1 (completely similar). The formula for this is given below.

$$cosine\ similarity = \cos(\theta) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \tag{4}$$

In order to find embeddings with a high cosine similarity (or low cosine distance) in a large dataset efficiently we use Hierarchical Navigable Small Worlds (HNSW) based vector search [39]. HNSW is an algorithm that finds the k most similar documents to a query with logarithmic time and space complexity (O(log(N)). It accomplishes this based on Navigable Small World (NSW) graphs.

NSW graphs are network graphs that are built to have both long-range links and short-range links. As a result, each vertex (which in our research represents an SBERT embedding) is connected to a number of other vertices. In NSW these connected vertices are referred to as a *friend list*. Using this friend list, a nearest neighbor search is conducted through a *greedy* search process. Meaning, each iteration a new vertex is selected from the current friend list based on being closest to the queried vertex. When no vertex closer to the query than our currently selected vertex is available in the friend list, the stopping criterion is reached.

HNSW builds on this concept through adding a hierarchy. Meaning, the graph is separated in layers. In the top layers vertices are connected through long distances. In the lower layers the distances between the vertices are shorter. As a result, starting in the top layer, the algorithm can easily traverse longer distances to find vertices closer to the query. When reaching a local minimum, the algorithm traverses to a layer below to continue the search.

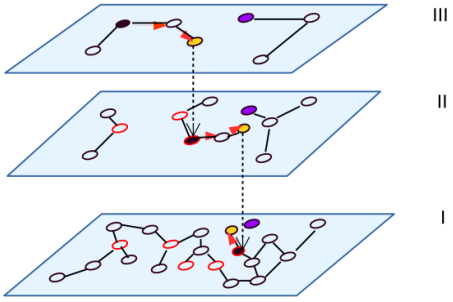


Figure 4: Illustration of the HNSW algorithm from [36]

Consequently, this algorithm decreases the amount of steps needed to find the closest neighbor. Because, in the first set of iterations the stepsize is larger and local minimums are found faster. An illustration of the HNSW algorithm can be found in Figure 4.

3.2 Text granularity and ranking methods

Because relevant information can be exclusive to a specific part of a document [1], we conduct experiments on two levels of text granularity: document and paragraph. Here, the paragraph level means that we query and retrieve paragraphs instead of documents. However, for the relevance ranking we only consider documents. As a result, the experiments on the paragraph level require a document ranking to be derived from the returned paragraphs. For this, we define two different paragraph based document rankings.

The first ranking method is based on taking the highest ranked paragraph of a document in the ranking as the overall document ranking. For example, say we return 6 paragraphs from 3 unique documents in the following order: $\{d_1, d_2, d_2, d_3, d_3, d_3\}$ where d_i refers to a paragraph from document i. Then, our document ranking will be as follows: $\{1, 2, 3\}$. Note how the ranking from the first paragraph of a document determines its position in the document ranking.

The second ranking method is based on counting the amount of paragraphs per document in the ranking, and using that count to rank the documents. For example, say we return the same 6 paragraphs in the following order: $\{d_1, d_2, d_2, d_3, d_3, d_3\}$ where d_i refers to a paragraph from document i. Then, our document ranking will be as follows: $\{3, 2, 1\}$. Note how in contrast to the previous ranking method, the amount of paragraphs per document determines the ranking.

Given the fact that the number of paragraphs per document differs, we expect the approach based on the highest ranked paragraph to outperform the count based approach. Because, in a count based approach, documents with *fewer* paragraphs are biased to have fewer matching paragraphs, regardless of their individual relevance to the query.

3.3 Relevance feedback

Relevance feedback is the process of iteratively improving the relevance rankings based on user feedback. In this research, the feedback is given based on automatic local analysis instead of real human input. As a result, this research uses pseudo relevance feedback. The rules for our pseudo-relevance feedback are explained in the next section. In this section, we explain the different relevance feedback methods researched in this thesis.

3.3.1 Baseline methods

In this research we have three baseline methods for the relevance feedback experiments. The first baseline method is to re-use the original queried embedding. Meaning, if a user accepts/declines documents, the ranking of documents remains constant. This is referred to as "no feedback" or "original".

Next, the second baseline method is based on keyword expansion. In keyword expansion, each iteration a number of keywords from the documents with *positive* pseudo relevance feedback are selected and appended to the original query using an OR operator. As a result, the selected keywords serve as a pre-filter for the original query where the documents should contain at least 1 of the collected keywords to be considered.

The selection of the keywords from the documents is based on TF-IDF. More specifically, after selecting a number of documents through relevance feedback, the words from the selected documents are sorted based on their *inverse document frequency*. Each iteration of feedback, the top 10 of this set of words is appended to the keyword filter.

Finally, the third baseline method is based on Rocchio [50], which is a commonly used vector based approach for relevance feedback. In Rocchio, the queried embedding is a weighted average of the original embedding and (the average of) the selected embeddings. The parameters that control this weighted average are α (which impacts the weight of the original embedding) and β (which impacts the weight of the selected embeddings). The formula for this method is shown below. In this formula, $E(\dots)$ refers to the embedding of a document and $Avg(E(p_1)\dots E(p_k))$ refers to the average embedding of the selected documents. Note, the most commonly used setting for the parameters is $\alpha = 0.5$ and $\beta = 0.5$ [4]. Hence, we use these parameter values in our experiment.

$$E_{Qnew} = \alpha * E(Q_{Original}) + \beta * Avg(E(p_1)\dots E(p_k)) \tag{5}$$

3.3.2 Vector based relevance feedback

Since the queried and collected texts have vectors (e.g. BERT or TF-IDF), the relevance feedback methods are based on vector operations. These vector operations are based on summing and averaging the vectors. For this, we experiment with both cumulative (i.e. include the queried vector in the average/sum) and non-cumulative feedback (i.e. exclude the queried vector in the average/sum).

3.3.3 Feedback amplification

For text similarity methods implemented on the paragraph level, relevance feedback can be amplified to the document level. Meaning, if a given paragraph receives positive relevance feedback, then that feedback can be extended to other paragraphs that have the same parent document. In our research this will be referred to as "amplified feedback" (or "amp" in tabular formats). Note, feedback amplification is not applicable to any of our baseline methods.

4 Experimental Setup

The first experiment focuses on comparing the performances of different text similarity methods. The second experiment focuses on implementing the best performing similarity method using different relevance feedback strategies. This section provides an overview of these experiments and some common denominators like performance evaluation, data (pre-processing) and configuration.

4.1 Performance evaluation

We evaluate the different similarity methods based on recall, precision and F1 scores (see formula's below). For these metrics, the definition of a "true positive" is a returned document that is of the same set as the queried document. This positive set is based on the annotations of the dataset. For example, if we query a document annotated as "sports", then the returned document should also be annotated as "sports" to be considered a true positive.

The main performance metrics are defined as follows. First, recall refers to the fraction of relevant items retrieved [40]. Given our definition of a true positive mentioned earlier, this metric resembles the fraction of documents from the positive set that have been returned (as similar) after querying a document.

$$Recall = \frac{\#(relevant\,items\,retrieved)}{\#(relevant\,items)} \tag{6}$$

Next, precision refers to the fraction of retrieved documents that are relevant [40]. Given our definition of a true positive, this metric resembles the fraction of returned documents that are from the positive set.

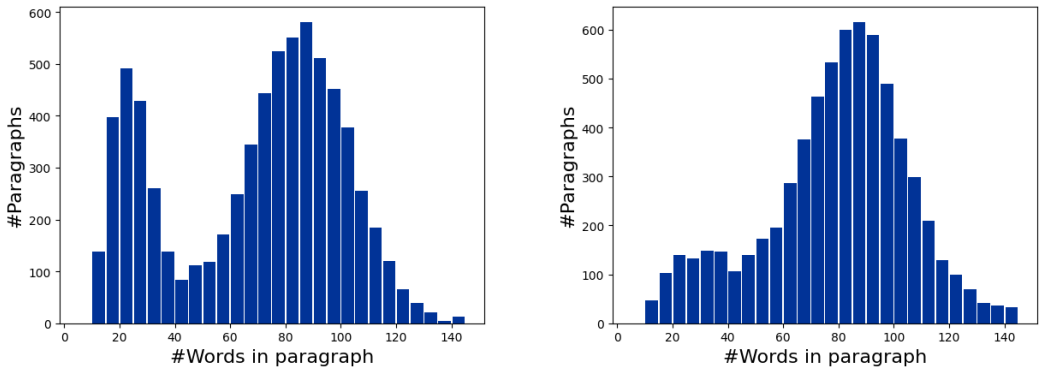
$$Precision = \frac{\#(relevant\,items\,retrieved)}{\#(retrieved\,items)} \tag{7}$$

Besides the precision and recall scores we also evaluate our methods with the F1-score. This performance metric is the weighted harmonic mean of the precision and recall values mentioned earlier.

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

Finally, to combine the results of different experiments into one concluding metric, we take the macro average. This metric is based on averaging the precision/recall/F1 score of different classes (i.e. topics) in our experiments. This metric will be used to compare the performances of different similarity methods and relevance feedback strategies.

$$Macro\ Average(X) = \frac{\sum_{i=1}^{n} X_i}{n} \tag{9}$$



(a) Test set

(b) Ambiguous set

Figure 5: Distributions of the paragraph lengths

4.2 Data and pre-processing

Our research uses the RCV-1 v2 [3] dataset in all the experiments. This dataset was made public in 2005 by Reuters News and consists of 806784 news articles. A complete overview of this dataset can be found in Table 35. Due to hardware constraints, we did not use the complete dataset in our experiment. Instead, we randomly sample 300 articles per topic from a set of 15 topics.

The sampled topics in our research are as follows. Firstly, the train set consists of the topics {Religion, Health, Travel and Tourism, Advertising and Promotion, Unemployment}. Second, the set of validation topics is {Fashion, Metals Trading, Crime Law Enforcement, Arts Culture Entertainment, Weather}. Thirdly, our test topics are {Strategy/Plans, Regulation/Policy, War Civil War, Sports, Elections}.

For all similarity methods, the final result is based on the performance on the test set. The usage of the train and validation sets differs per method and will be discussed in this section. Note, besides the regular test set, the methods are also evaluated on an additional set of topics that stem from the same parent topics. This set will be referred to as the "ambiguous" set and consists of the topics {EC Internal Market, EC Corporate Policy, Forex Markets, Energy Markets}.

Note, because the RCV-1 v2 dataset has multiple topics per article, the proper size of a topic in a random sample can (slightly) exceed the sample size. As a result, a tabular overview of the exact sample size per topic can be found in Table 34 (see Appendix).

As for pre-processing, for Quorum and TF-IDF we filter out stopwords, numbers, and convert the text to lowercase. The stop word list used for this is publicly available on GitHub[1]. For the SBERT and SVM based experiments, we only remove numbers and special characters.

Finally, for the experiments on the paragraph level, we first split the text into sentences using the `<p>...</p>` tags in the XML files from RCV-1 v2 dataset [3]. Next, a paragraph is created through concatenating every 3 adjacent sentences of a document (and the remainder). For SBERT the amount of words in a paragraph can't exceed 384. Hence, the paragraphs in the topic sets used in the SBERT experiments (test and ambiguous) shouldn't exceed that limit. As shown in Figure 5, our experiment adheres to this requirement.

[1] https://github.com/stopwords-iso/stopwords-en

4.3 SVM based approach

In contrast to the similarity methods, our second baseline method (linear-SVM) is supervised. Meaning, in order to use it as a classifier it first needs to be trained on labeled data. To preserve the direct comparison between this baseline method and the similarity methods, the test data is the exact same set of documents as the similarity methods.

As a result, the train data is an additional random sample of documents from the same topics (for both the test and ambiguous set). Since we use a 60/40 train/test split, the size of this random sample is 450 articles. Note, the training samples are checked to not have any overlap with the test samples.

For this baseline method, there is no parameter optimization or ranking. Hence, we run this experiment directly on the test and ambiguous set. Also, – since the purpose of this baseline is to emulate a common approach in eDiscovery – this experiment is only conducted on the document level.

4.4 Quorum search

For the experiments based on Quorum search we set two parameters. First, *N*. Which impacts the amount of words considered for comparison (expressed as a percentage of the queried document). Second, M. Which impacts the amount of words that need to match between two documents to be defined similar (expressed as a percentage of N).

In order to find the optimal combination between N and M we do a grid search of these parameters. For N, this grid search goes from 20% to 100% with a stepsize of 20%. For M, this grid search goes from 5% to 50% with a stepsize of 5%. As a result, for each value of N we iterate through 10 different values of M.

Besides these parameters, there are no parameters that can be changed. So, we conduct this grid search of M and N on both the train and validation sets. The optimal combination on each of these sets is based on the highest macro F1 score. Finally, we take the average of M and N from these optimal combinations as the parameter values for the remaining experiments.

Next, in the Quorum experiments the returned documents/paragraphs are ranked. Hence, we iterate through the returned results using different cutoff rates. The cutoff rates in our experiment are the top 10, 20, 50, 100, 200, 300 and 500 results. For each of these cutoff rates we compute the precision, recall and F1 Score.

Finally, for the paragraph level experiments we test both paragraph based document rankings mentioned in the previous section. Also, to get a better understanding of the paragraphs used in our research, we run the experiment for both querying the first paragraph and a random paragraph.

4.5 TF-IDF based approach

For these experiments we can set two parameters. First, the *minimum document frequency*. This refers to the minimum amount of documents (in the dataset) a word needs to occur in in order to be considered. Second, the *maximum document frequency*. This refers to the maximum amount of documents (in the dataset) a word needs to occur in in order to be considered.

Similar to the Quorum experiments, we set these parameters on the train and validation set. However, due to limited computing resources, we didn't do a full grid search. Instead, we used a manual search to find the optimal parameter values. Finally, the scores on the test set and the ambiguous set will be considered as the definitive performance of this approach.

Model name	Size	#Dimensions	Speed (sentences/sec)
all-mpnet-base-v2	420 MB	768	2800
all-MiniLM-L12-v2	120 MB	384	7500
all-MiniLM-L6-v2	80 MB	384	14200

Table 2: Selected BERT models' characteristics according to [47]

Since we use MLT, the returned documents/paragraphs are ranked. Hence, we iterate through the returned results using different cutoff rates. The cutoff rates in our experiment are the top 10, 20, 50, 100, 200, 300 and 500 results. For each of these cutoff rates we compute the precision, recall and F1 Score.

Finally, the experiments are conducted on both the document and the paragraph level. For the paragraph level experiments we test both paragraph based document rankings mentioned in the previous section. Also, to get a better understanding of the paragraphs used in our research, we run the experiment for both querying the first paragraph and a random paragraph.

4.6 Dense vector search

In our experimental setup for DVS we don't have any parameters to set. Hence these experiments are conducted directly on our test set and our ambiguous set. Next, we experiment with three different pre-trained SBERT models. These are selected based on being all-round (i.e. general purpose) models selected by the SBERT documentation [47].

The first pre-trained model used in this experiment is *all-mpnet-base v2*. This is the largest (and according to the documentation best performing) SBERT model. Its embeddings are of fixed size and have 768 dimensions. The second pre-trained model used in this experiment is *all-MiniLM-L12-v2*. This model is smaller than the previously mentioned pre-trained BERT model and its embeddings only have 384 dimensions. Finally, the third BERT model used in this experiment is *all-MiniLM-L6-v2*, which is the smallest model used in this experiment. Similar to the previous model it also has 384 dimensions. An overview of the characteristics (and performance) of the selected pre-trained SBERT models according to the documentation [47] can be found in Table 2.

Due to a maximum input text size of 384 words in SBERT [48], the DVS experiments are conducted on the paragraph level only. Hence we test both paragraph based document rankings mentioned in the previous section. Also, to get a better understanding of the paragraphs used in our research, we run the experiment for both querying the first paragraph and a random paragraph. Moreover, to verify the numbers shown in Table 2 derived from the documentation, we also record the time taken and RAM consumed by the different SBERT models in our experiment.

Finally, in this experiment the paragraphs returned as similar are ranked. Hence, we iterate through the returned results using different cutoff rates. The cutoff rates in our experiment are the top 10, 20, 50, 100, 200, 300 and 500 results. For each of these cutoff rates we compute the precision, recall and F1 score.

4.7 Relevance feedback

For our relevance feedback experiments we implement a form of pseudo relevance feedback. Here, the feedback is based on the same definition of a true positive as mentioned earlier. Meaning, the topic of a returned document should contain the queried topic to get positive feedback.

Algorithm 2: Pseudo relevance feedback experiment

```
   input  : paragraph, maxIterations
   output: recall at different iterations
 1 iteration = 0
 2 while iteration ≤ maxIterations do
 3 |   filter = acceptedDocuments + declinedDocuments
 4 |   results = query(paragraph, filter)                    // Returns top 10 results.
 5 |   for result in results do
 6 |   |   if feedback for result is positive then
 7 |   |   |   paragraph = processFeedback(result)
 8 |   |   |   acceptedDocuments += result
 9 |   |   else
10 |   |   |   declinedDocuments += result
11 |   iteration + = 1
12 return recall at different iterations
```

An overview of the layout of the experiment can be found in Algorithm 2. Note how each iteration the already collected/declined documents are filtered from the search. Also, note how the query is updated each iteration based on the pseudo relevance feedback. In our implementation of this experiment, we collect 10 documents/paragraphs each iteration, and we iterate to a maximum of 75 iterations (since we have just under 7500 paragraphs in our test set).

For performance evaluation, we record the number of iterations needed to achieve a certain recall. There are two formats for this. First, we record the recall achieved every 5 iterations. This format shows the trend of this performance metric. Second, we record the average (and standard deviation) of iterations needed to achieve exactly 80% recall. This format is used to make an exact comparison between the methods. The value of 80% recall is chosen due to it's common usage as a minimum threshold in eDiscovery scenarios [51]. Finally, for every pseudo relevance feedback strategy we record the time taken per iteration as a separate performance metric.

4.8 Configuration

The experiments are conducted on a local device. The CPU of this device is an Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz and it has 16GB of RAM memory. As for software, the TF-IDF and DVS based text similarity experiments were conducted using Solr [38], which is built on top of Lucene [24]. The SVM based experiments are conducted using the Sklearn library [7]. Here, apart form the token pattern (which is set to `r"(?u)\b\w+\b"`) and the normalization (which is set to "None"), all default parameter values apply. Finally, the source code used in this experiment is publicly available on GitHub[2].

[2]https://github.com/TimoKats/MasterThesis

5 Results

This section is an overview of the results from our experiments. In the first subsection, we discuss the results of the individual text similarity methods. In the second subsection, we discuss the results of the different relevance feedback strategies. Note, for all the results presented in this section a complete and tabular overview is available in the appendix.

5.1 Text similarity methods

In this subsection we share the results of the text similarity methods. First, we share the results for Quorum search. Next, we share the results for the TF-IDF based approach. Thereafter, we share the results for the DVS experiments. Finally, we compare these individual results and contextualize them with the results of our second baseline method.

5.1.1 Quorum search

For the document level, the grid search on the train and validation sets resulted in the parameter values of N=100% and M=7.5%. Next, for the paragraph level, the grid search on the train and validation sets resulted in the parameter values of N=40% and M=10%. Hence, the results on the test and ambiguous sets are based on these parameter values.

Figure 6 shows the recall and precision scores for Quorum search on the document level and paragraph level. These results show that for Quorum search, the document level works better than both paragraph based approaches. Also, the paragraph based document ranking where the *first* paragraph determines the ranking of a document outperforms the document ranking based on *counting* the paragraphs. Finally, for both paragraph based experiments it's apparent that querying the first paragraph gives slightly better results than querying a random paragraph.

In summary, these results show that the most suitable configuration for Quorum search (in our experiment) is implemented using N=100% and M=7.5% on the document level.

5.1.2 TF-IDF based approach

For both the paragraph level and the document level, the manual search on the train and validation sets resulted in the parameter values minDf=0 and maxDf=0.8. Hence, the results on the test and ambiguous sets will be based on these parameter values.

Next, the results in Figure 7 show that for this approach the document level is the most suitable level of text granularity, since it outperforms both paragraph approaches. Also, the paragraph based document ranking where the first paragraph determines the ranking of a document outperforms the document ranking based on counting the paragraphs. Finally, for both paragraph based experiments it's apparent that querying the first paragraph gives better results than querying a random paragraph. In summary, these results show that the most suitable configuration for our TF-IDF based approach is implemented using minDf=0 and maxDf=0.8 on the document level.

5.1.3 Dense vector search

The results based on querying the first paragraph can be found in Figure 10. Next, the results based on querying a random paragraph can be found in Figure 11. In both sets of results, it's apparent that the largest pre-trained SBERT model (all-mpnet-base v2) outperforms the other pre-trained SBERT models for both paragraph based document rankings. Next, similar to the other text similarity methods, the document ranking based on the first paragraph outperforms the document ranking based on counting the paragraphs.

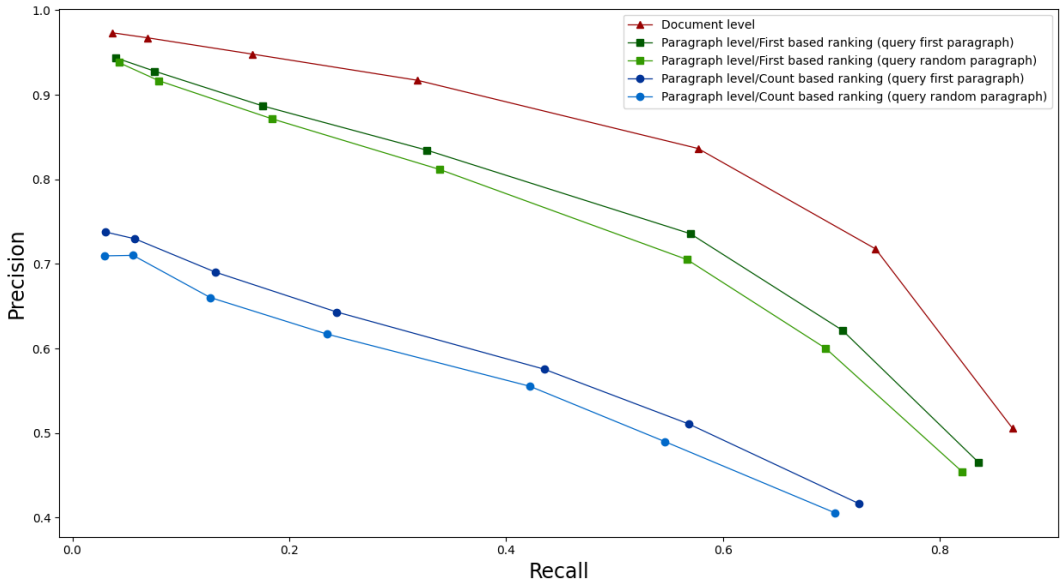


Figure 6: Results for Quorum search on the test set

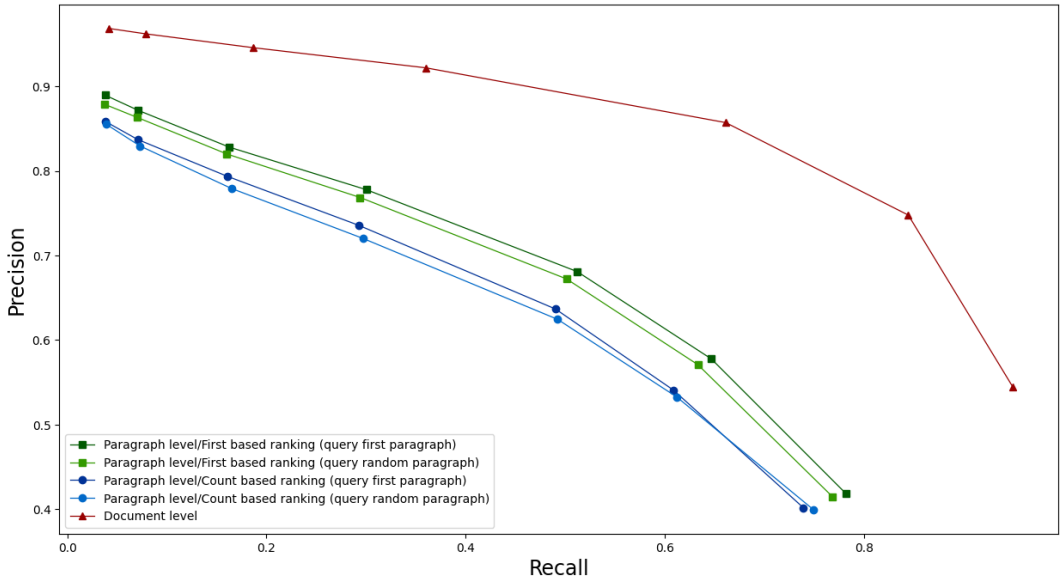


Figure 7: Results for the TF-IDF based approach on the test set

Cutoff@	Quorum (test)	TF-IDF (test)	DVS (test)	Quorum (ambiguous)	TF-IDF (ambiguous)	DVS (ambiguous)
10	0.071	0.079	0.072	0.061	0.072	0.068
20	0.130	0.145	0.132	0.107	0.128	0.123
50	0.282	0.311	0.290	0.222	0.265	0.264
100	0.472	0.517	0.494	0.354	0.421	0.439
200	0.683	0.745	0.755	0.494	0.580	0.643
300	***0.729***	***0.793***	***0.839***	0.547	***0.629***	***0.708***
500	0.638	0.691	0.713	***0.556***	0.612	0.661

Table 3: Macro F1 scores of the best performing configuration for each similarity method

As for the difference between querying a random or the first paragraph of a document, this is shown in Figure 12 for our best performing pre-trained SBERT model and ranking method. Here, in accordance with the results from our Quorum and TF-IDF based approaches, querying the first paragraph gives slightly better results than querying a random paragraph.

Next, the results for RAM usage are shown in Figure 8 and the results for time taken per embedding are shown in Figure 9. In these figures, the x-axis refers to the individual paragraphs (i.e. each x-tick is a paragraph) and the y-axis refers to the time taken/memory consumed to create an embedding for that paragraph. Here, despite the large variance of results shown in Figure 8, both these Figures confirm the pattern described in the documentation from SBERT (shown in Table 2), which is that the larger models are more time and memory consumptive.

5.1.4 Combined results

For comparison purposes, the precision-recall graph for each method's identified configuration is shown in Figure 13 for the test set and Figure 14 for the ambiguous set. Moreover, the macro F1 scores associated with these precision-recall graphs are shown in Table 33. To reiterate, the identified TF-IDF and Quorum based similarity methods are implemented on the document level whereas the DVS based approach is implemented on the paragraph level (using *all-mpnet-base v2*).

On test, DVS outperforms our TF-IDF and Quorum based approaches. Next, on the ambiguous set the results again show that DVS outperforms our other approaches. However, the margin between the individual text similarity approaches is larger than on the test set. Also, the performance of the identified similarity methods is collectively worse on the ambiguous set.

Finally, the results of our second baseline approach are shown in Table 4. When comparing the performance of all text similarity methods with these results on the test set, the results are comparable with our identified DVS method (MF1 of 0.839 for DVS and MF1 of 0.827 for linear-SVM). However, on the ambiguous set the linear-SVM outperforms all similarity methods (MF1 of 0.708 for DVS and MF1 of 0.772 for linear-SVM).

Dataset	Macro Precision	Macro Recall	Macro F1 Score
Test	0.922	0.753	0.827
Ambiguous	0.775	0.791	0.772

Table 4: Results using TF-IDF with a linear-SVM

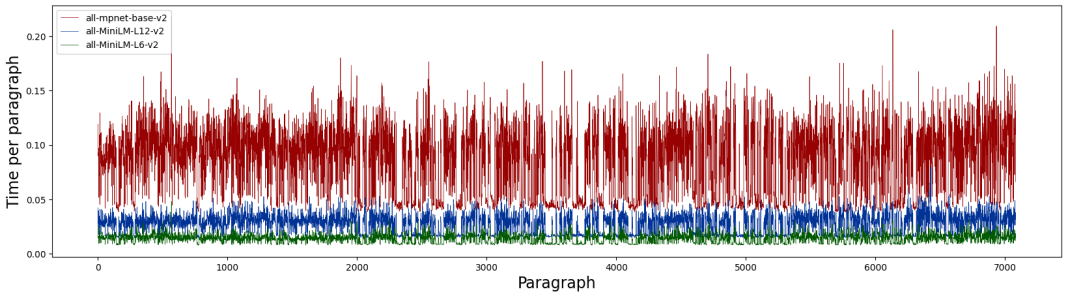


Figure 8: Time consumed to create SBERT embeddings per paragraph

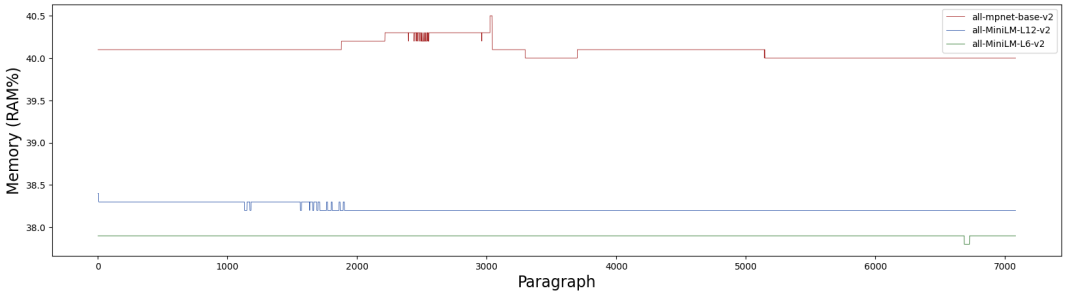


Figure 9: RAM% consumed to create SBERT embeddings per paragraph

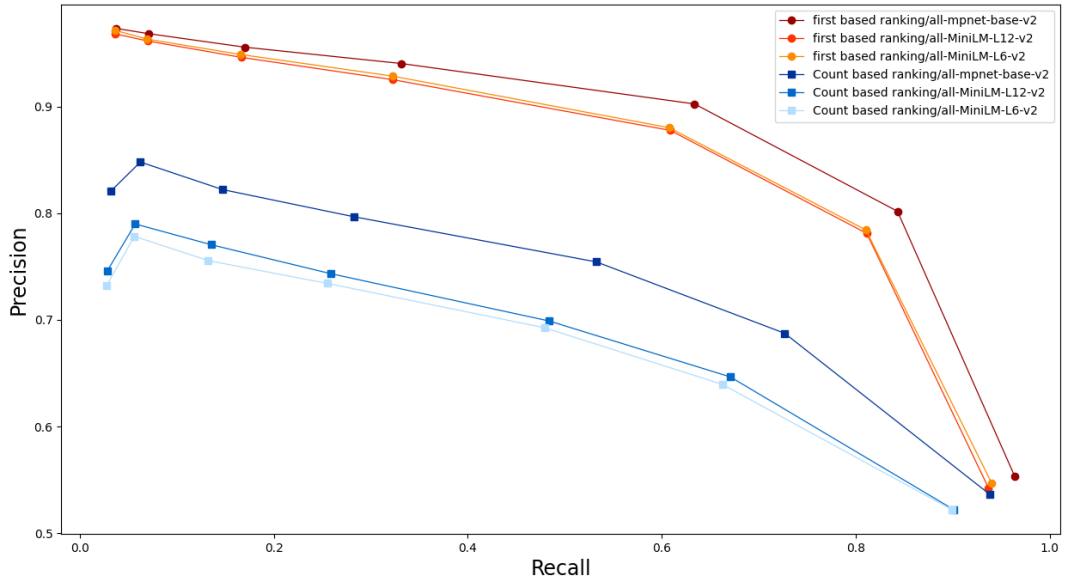


Figure 10: DVS results for querying the first paragraph

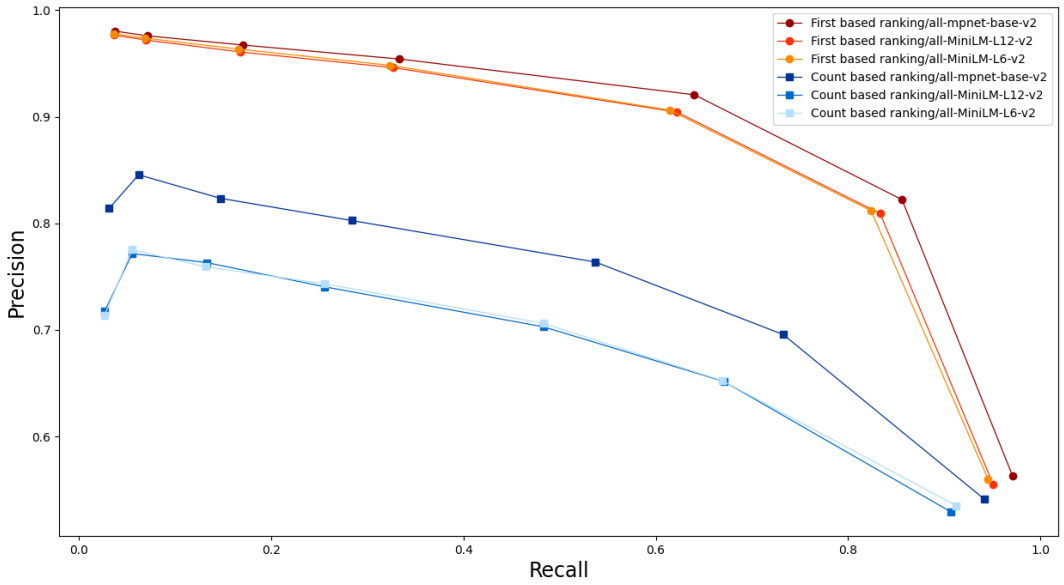


Figure 11: DVS results for querying a random paragraph

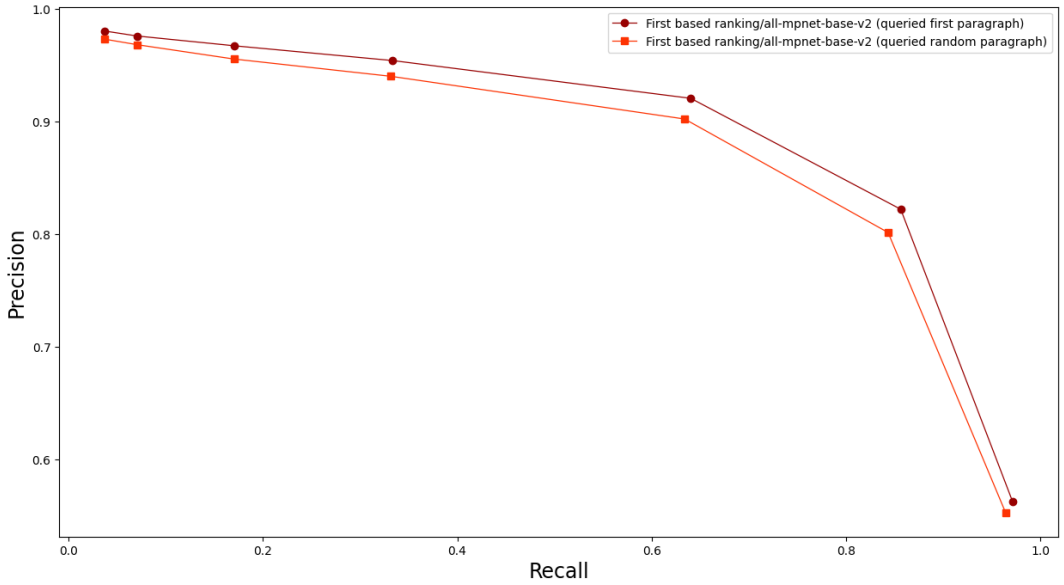


Figure 12: Difference querying a random paragraph and the first paragraph with DVS

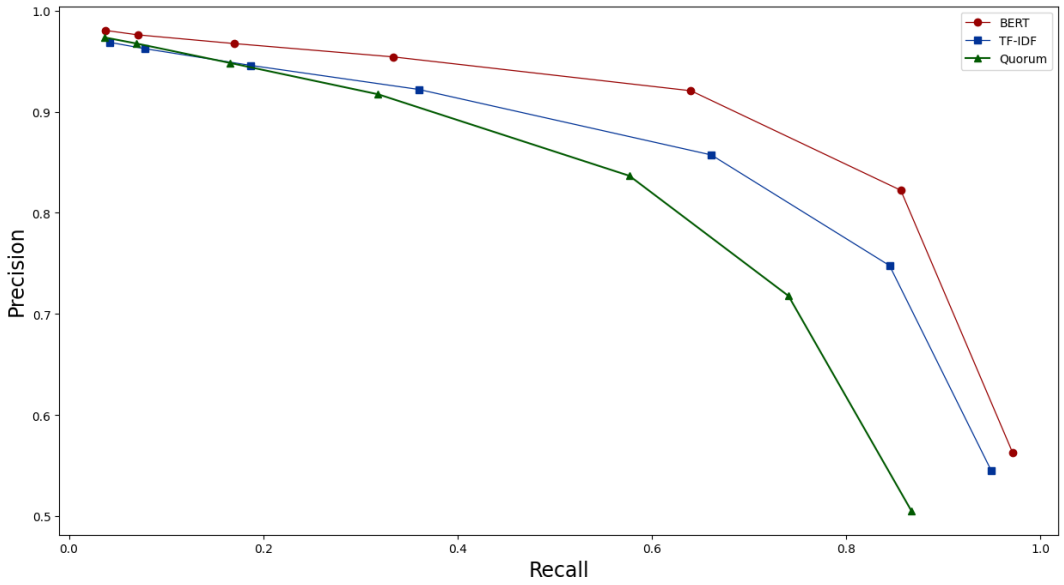


Figure 13: Identified methods' performance on test topics

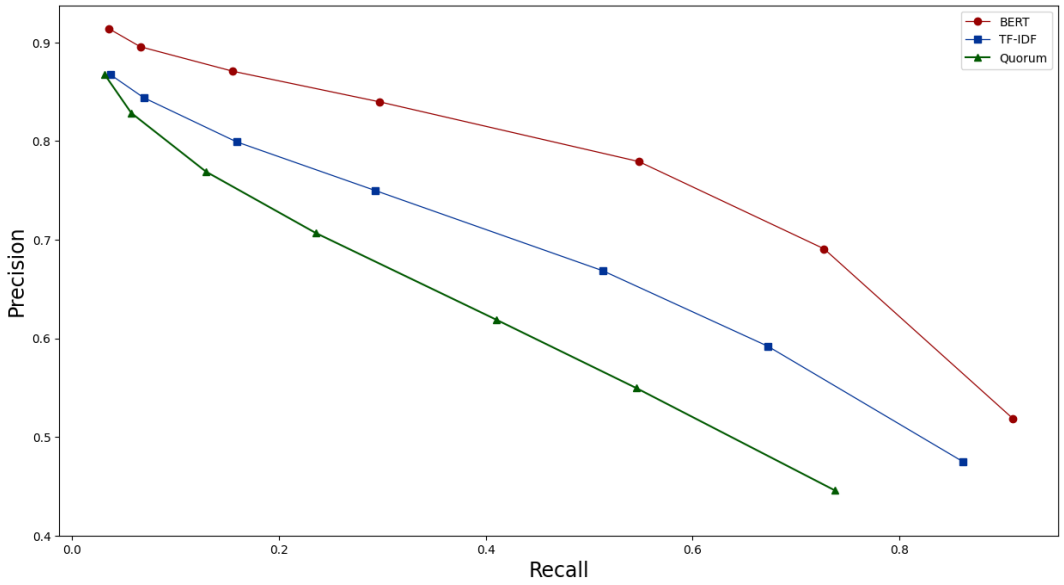


Figure 14: Identified methods' performance on ambiguous topics

Feedback strategy	Mean (Test)	Std Dev (Test)	Mean (Ambiguous)	Std Dev (Ambiguous)
No feedback	33.364	8.836	46.769	10.040
Keyword expansion	32.240	8.025	44.231	9.941
Rocchio ($\alpha = 0.5$, $\beta = 0.5$)	29.502	3.793	37.975	5.110
Average	29.433	1.511	32.648	2.971
Average (amp)	30.167	1.026	32.990	1.797
Sum	***28.285***	0.746	***29.407***	2.126
Sum (amp)	28.462	***0.499***	30.538	***1.310***
Average	31.542	1.136	38.683	4.409
Average (amp)	32.814	1.077	38.412	2.638
Sum	32.802	1.118	38.804	5.278
Sum (amp)	32.201	0.872	38.201	3.774

Table 5: Iterations needed to achieve 80% recall (baselines are in the top cells, cumulative approaches are in the middle cells, non-cumulative approaches are in the bottom cells).

5.2 Relevance feedback

The results for the relevance feedback are based on the identified (i.e. best performing) similarity method from the previous experiments. To reiterate, this method is based on DVS (where the pre-trained model is *all-mpnet-base v2* and the document ranking is based on the first paragraph).

Using this similarity method, the relevance feedback results for the cumulative strategies are shown in Figure 15 and Figure 16. Next, the results for the non-cumulative strategies are shown in Figure 17 and Figure 18. In all of these figures, the y-axis represents the recall and the x-axis represents the iterations needed to achieve that recall. Note, every iteration translates to a review effort of 10 paragraphs. Similar to the previous experiments, the results are available on the test set and the ambiguous set. Finally, the average amount of iterations (and standard deviation) needed to achieve 80% recall for all methods and datasets is shown in Table 5.

First, the results on the test set. Here, it's apparent that the feedback methods based on vector operations require fewer iterations and have a lower standard deviation that the baseline methods. The best performing feedback method is summing the vectors. Next, the results on the ambiguous set. Here, all similarity methods require more iterations to reach 80% recall and have a higher standard deviation than they have on the test set. Still, feedback methods based on summing the vectors (cumulatively) gives the best results.

As for comparing cumulative and non-cumulative relevance feedback methods, here it's apparent that cumulative feedback methods outperform non-cumulative feedback methods for all vector based relevance strategies. Moreover, the difference between averaging and summing vectors is smaller when using non-cumulative strategies.

A noteworthy finding when comparing the results from the test set and the ambiguous set is the gap between the minimal baseline method (of no feedback) and the optimal feedback method (of cumulatively summing the vectors). On the test set, the reduction of review effort (measured in the number of iterations to achieve 80% recall) is 17.85%. On the ambiguous set however, this reduction of review effort equals 59.04%. Another noteworthy finding is that amplifying the feedback to sibling paragraphs reduces the standard deviation, but not the amount of iterations needed to achieve 80% recall.

Finally, for all methods we measured average time taken per iteration. The results for this are shown in Table 6. These results show that relevance feedback strategies based on vector operations (average and sum) add little latency to the experiment compared to no feedback. Still, amplifying feedback to sibling paragraphs does add some latency to the experiment for both averaging and summing vectors. However, keyword expansion adds the most latency to the experiments.

Strategy	**Average time per iteration (in seconds)**
No feedback	0.02298
Average	0.02386
Sum	0.02417
Rocchio	0.02901
Sum (amp)	0.04982
Average (amp)	0.05097
Keyword expansion	0.12093

Table 6: Average execution times for different relevant feedback strategies (sorted)

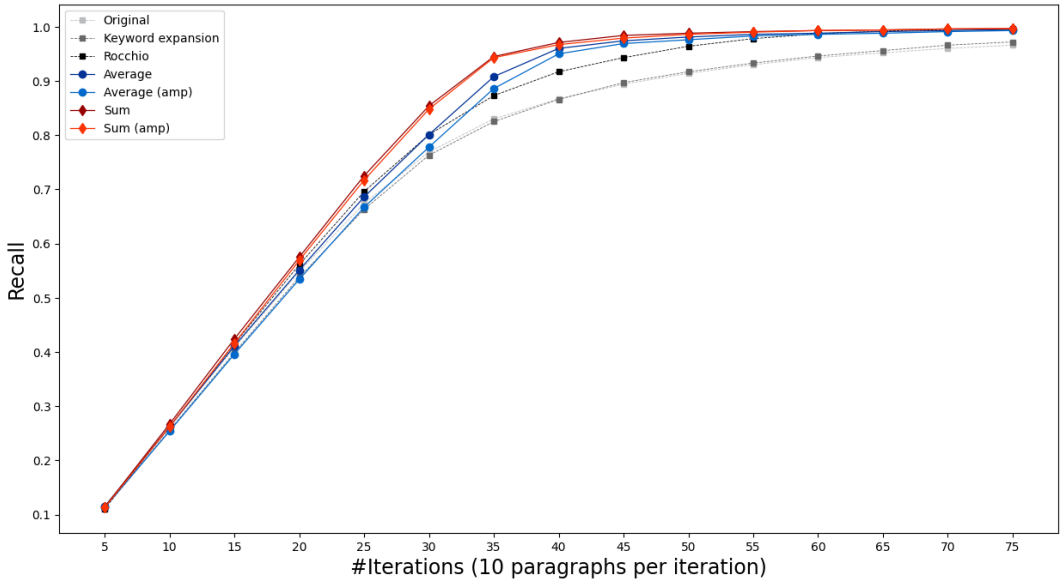


Figure 15: Cumulative methods' performance on test topics

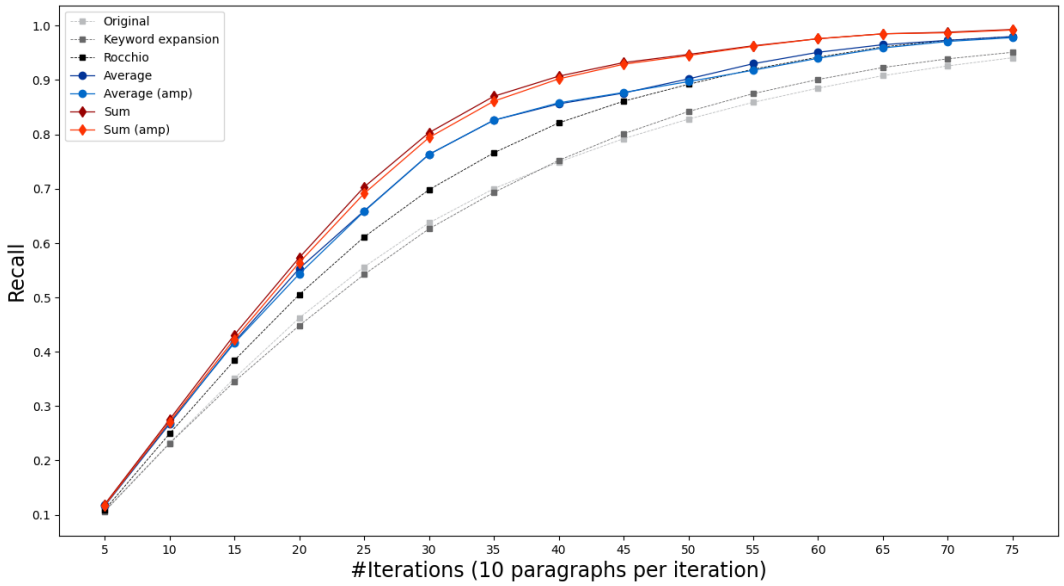


Figure 16: Cumulative methods' performance on ambiguous topics

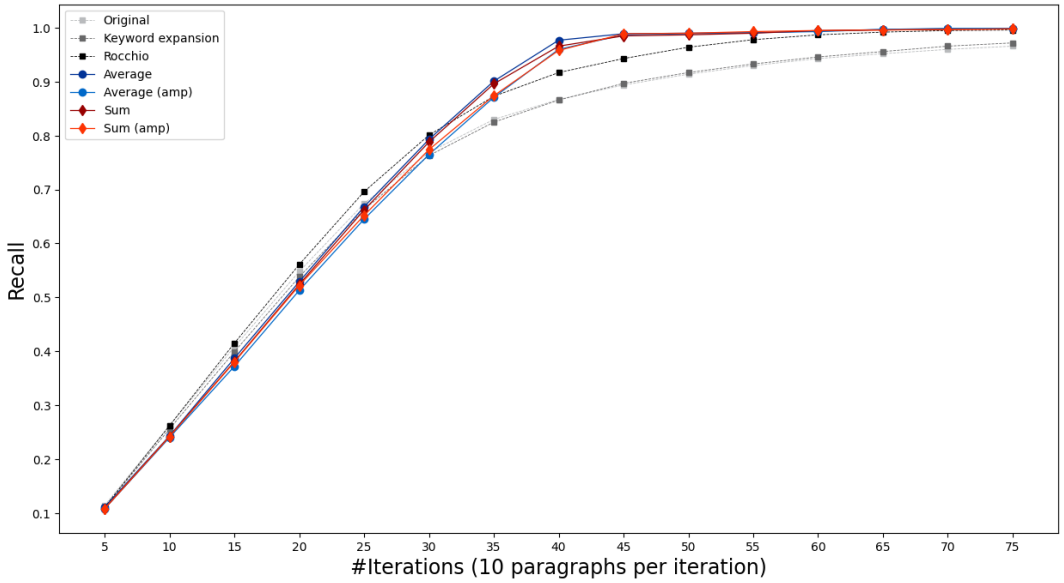


Figure 17: Non-cumulative methods' performance on test topics

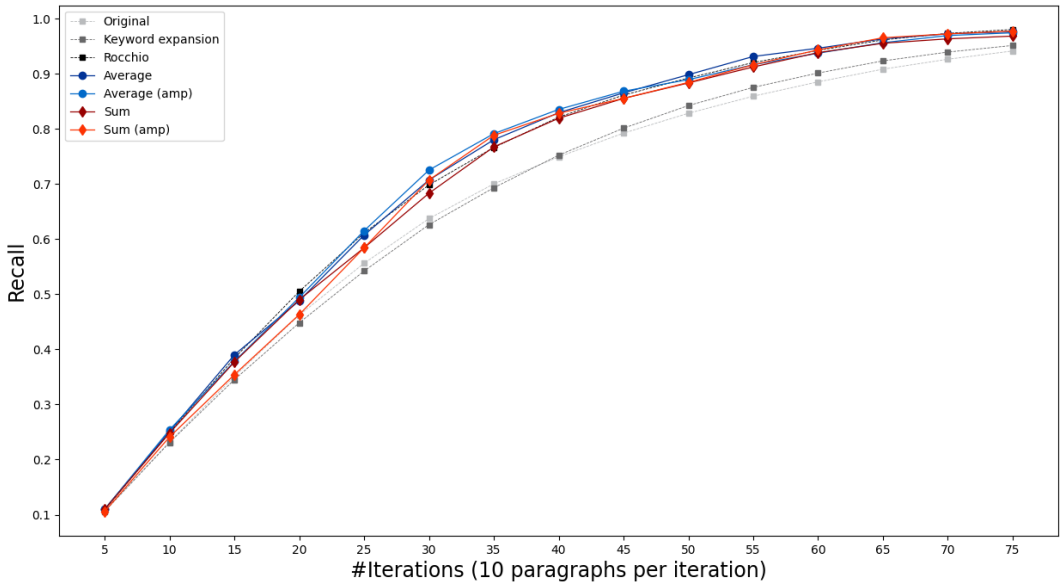


Figure 18: Non-cumulative methods' performance on ambiguous topics

6 Discussion

This section is a discussion of the results. First, we interpret the results for our two main experiments. Second, we discuss the limitations of our research and the resulting suggestions for future work.

6.1 Interpretations

The results presented in the previous section are interpreted in this subsection. First, the results for the text similarity methods. Next, the results for the pseudo relevance feedback experiments.

6.1.1 Text similarity methods

For all of text similarity methods, some common denominators emerge from the results. First, querying the first paragraph gives better results that querying a random paragraph. This coincides with findings from our literature study, where research found that the effectiveness of "query by document" approaches depends on the prevalence of relevant terms in the queried text [65]. Combining this finding with our results, it's probable that the first paragraphs in the RCV-1 v2 dataset outperform random paragraphs because they have a higher prevalence of relevant terms. This property could be exclusive to the news articles used in our research, and therefore might not translate to the data typically used in real-world eDiscovery scenarios.

Another commonality between the methods is related to the paragraph based document rankings. Here, we see that for all methods ranking documents based on the first paragraph in the ranking gives better results than querying documents based on counting their paragraphs in the ranking. Potentially, this could be related to differences in the number of paragraphs per document. Because, if a document only has one paragraph then it's always bound to be at the bottom of a count-based ranking. Regardless of how related/similar that document is.

As for comparing the different text similarity methods, both sets of experiments show that DVS outperforms the other methods. However, the difference in performance between the different methods is more profound on the ambiguous set. This implies that for higher levels of data ambiguity, the DVS based approach has more added value.

Next, the comparison of our similarity methods and our second baseline method. Here, the results show that a commonly used supervised method (in our research exemplified through a TF-IDF based linear-SVM) performs similar to an unsupervised text similarity based method on our test set. For higher levels of data ambiguity however, the linear-SVM outperforms all similarity methods. Still, regarding the fact that the similarity methods are unsupervised (and therefore don't require labeled data to be trained), these results are still encouraging.

Finally, it must be noted that the performance improvement in SBERT does come at a computational cost. As Figure 8 and Figure 9 show, the memory usage and time consumption increases for better performing SBERT models.

6.1.2 Pseudo relevance feedback

For the second experiment, the best performing text similarity method (DVS, with a first-based document ranking and *all-mpnet-base v2* as pre-trained model) was implemented for relevance feedback. For both the test set and the ambiguous set, the experiments show that relevance feedback methods based on cumulatively summing the vectors reduce review effort the most. Interestingly, this improvement does not seem to come at a computational cost. Since Table 6 shows that the execution time of these methods (without feedback amplification to sibling paragraphs) is fairly similar to our minimal baseline method (of no feedback). Note, when

optimizing for a low standard deviation, amplifying feedback to sibling paragraphs is beneficial, which does add latency to the experiment.

As for the reduction of review effort, we compare the iterations needed to achieve 80% recall with our minimal baseline (of no feedback). Using this method, the reduction of review effort is 17.85% on the test set and 59.04% on the ambiguous set. Related work that used an SVM based approach for the same purpose found that their strategy reduced review effort between 10.6% to 96.9% using the same dataset [45]. This implies that the method used in our research could be more predictable in terms of reducing review effort.

Moreover, their suggested method takes 71 seconds per iteration (of 1000 documents), which translates to 0.071 seconds per document. This is far more than our best performing method, which takes ≈ 0.02 seconds per iteration (of 10 documents). Obviously, innovation in hardware could play a role in amplifying this difference, since their experiments were conducted 5 years ago. Also, their experiment didn't use a subset of RCV-1 v2 but the entire dataset. Still, the hardware used in their experiment (a shared 6 Intel Xeon E5-2630v3 CPUs @ 2.40GHz with 150GB of RAM) is powerful than ours (an Intel Core i7-10610U CPU @ 1.80GHz with 16GB of RAM) regarding RAM and CPU clock speeds. As a result, it's still probable that our method is generally faster than an SVM based approach.

6.1.3 Relevance

First, with regards to implementation, our results show that review effort can be decreased using text similarity based relevance feedback methods. An important side note in this finding is that relevance feedback accomplishes this through only re-ranking documents. As a result, this strategy does not produce false negatives automatically without the awareness of the user (in contrast to an SVM based approach). This is particularly important for eDiscovery in an adversarial legal system. Because, in this legal system the discovery process is typically *recall* oriented. Hence, a reduction (or even nullification) of false negatives is favorable.

Second, with regards to scientific novelty and contributions, we should state that the concept of relevance feedback in eDiscovery has been studied before [68]. However, certain parts within our implementation are (to our knowledge) novel and therefore contribute to science.

First, the evaluation of different paragraph based document rankings contribute to the domain of paragraph based document-to-document retrieval. Given the rise of large language models (that are generally limited to the paragraph level [37]) and the fact that relevant information can be exclusive to a specific part of a document [1], these findings are applicable beyond the technologies/embeddings used in this research.

Second, our results show that the usage of sibling paragraphs in relevance feedback can reduce the standard deviation of review effort. To our knowledge, this technique and finding are novel. Moreover, a more "stable" reduction of review effort could be favorable in real-world eDiscovery scenarios. Hence, this finding is not just novel, but also applicable.

6.2 Limitations

First, data in eDiscovery tends to be a heterogeneous mix of emails, documents, direct messages, etc [58]. But, the data used in this research is fairly homogeneous, since it only consists of news articles. Hence, there's a slight mismatch between the type of data used in this experiment and the data used in real-world eDiscovery scenarios. Thus, the lack of data heterogeneity in this experiment is a limitation of this research.

Second, due to hardware constraints, this research used a random sample of 300 articles per topic for the experiments. This does not resemble the size of datasets typically seen in

eDiscovery, which tend to be much larger [58]. Hence, the sample size of our used datasets is a limitation of this research.

Moreover, data in eDiscovery tends to come from the legal domain [17]. This contrasts with the data used in this research, which consists of news articles. As a result, certain properties of our method identified in this research might not translate to real-world eDiscovery applications. For example, the fact that querying the first paragraph outperforms querying random paragraphs. Still, given the fact that the RCV-1 v2 dataset is commonly used as a benchmark dataset in eDiscovery [20], the results are still an adequate indication of our method's performance.

6.3 Future work

Our first suggestion for future work is related to the level of data heterogeneity in the experiments. As mentioned in the previous subsection, data in eDiscovery tends to be heterogeneous whereas the data used in this experiment is fairly homogeneous. As a result, future work could research to what extent heterogeneous data impacts the ability of different text similarity methods to return similar documents.

Next, this research uses Solr's "MoreLikeThis" functionality to increase the speed of our TF-IDF based similarity experiments. Meaning, we didn't use any vector space scoring to compute the similarities between the TF-IDF vectors. The reasoning behind this is that computing the (cosine) similarities between n TF-IDF vectors results in quadratic ($O(n^2)$) time and space complexity, which is simply not viable in a real-world application.

However, recent innovations have made it possible to compute the pairwise similarities between (sparse) vectors much faster. An example of this is the ChunkDot Python library [2], which splits the TF-IDF matrix into chunks and computes the similarities in parallel. Future work could use this innovation to experiment with TF-IDF based cosine similarity as an additional text similarity method.

Thirdly, a category of text similarity methods not included in our experiments is neural network based text similarity methods. The reasoning behind the exclusion of this category of text similarity methods is related to its computational load, which is simply too large given our computational resources. However, if future work does have sufficient computational resources, this category of text similarity methods could be researched more and compared with the methods used in this thesis.

Our next suggestion for future work is related to the recent innovations in the domain of large language models (LLMs). In this research the relevance feedback experiments were based on SBERT embeddings. However, SBERT embeddings are not the only embeddings that can be used for this purpose. In fact, recent innovations from OpenAI has created text embeddings (e.g. *text-embedding-ada-002*) that can be used in a similar fashion [26].

Unfortunately, these embeddings have too many dimensions (1536) for our experimental setup, which allows a maximum of 1024 dimensions. As a result, these embeddings are not included in our experiment. Thus, future work could – given a different experimental setup – include these embeddings and compare their performance with the BERT based approach used in this research.

Finally, LLMs can also be implemented differently from our method. Namely, through a generative Artificial Intelligence (AI) based approach to reducing review effort. For this research, such an approach is considered out of scope. Still, there are developments in this domain that can be considered for further research. First, the usage of generative AI could be a complement to our method when implemented as a co-pilot/assistant for the reviewer. For example through assisting in formulating queries or summarizing search results [31].

Next, generative models (like ChatGPT) can also be used to review documents automatically by classifying documents as relevant/irrelevant to a (provided) topic description. Despite a possibility of hallucinatory reviews (i.e. answers that are factually incorrect or unrelated), this implementation has been shown to have potential [61].

7 Conclusion

This research aimed to evaluate the impact of changing the (text similarity based) relevance rankings based on relevance feedback. For this, the first research question was formulated as follows: can text similarity be used for relevance feedback? Our results show that the most suitable text similarity method performs comparable to a commonly used supervised method (in the form of a linear-SVM) and outperforms our baseline text similarity method. As a result, we conclude that text similarity methods can be used for this purpose.

Next, the second research question was formulated as follows: to what extent can relevance feedback help to reduce review effort? Here, our results show that the relevance feedback method identified in this research reduces review effort between 17.85% and 59.04%.

However, the homogeneity of the data used in this research does not accurately resemble the heterogeneous nature of data typically seen in eDiscovery scenarios. This is something future work could improve upon. Also, the sample size also does not resemble real-world scenarios in eDiscovery. Recent innovations like chunking or improved hardware could address this shortcoming.

Regardless, given the recall oriented nature of eDiscovery in (especially) adversarial legal systems, the results for the relevance feedback experiments are very encouraging. Since, in contrast to an active learning based strategy (which typically used for this purpose), this approach reduces review effort through only re-ranking documents. As a result, there are no false negatives created without the awareness of the user. This feature is important for achieving optimal recall and it makes this approach very suitable for real-world eDiscovery applications.

References

[1] Kolawole John Adebayo. *Multimodal Legal Information Retrieval*. PhD thesis, University of Luxembourg, Luxembourg, 2018.

[2] Rodrigo Agundez. ChunckDot Python library, May 2023. `https://pypi.org/project/chunkdot/`.

[3] Cyril Amini, Massih-Reza & Goutte. Reuters RCV1 RCV2 Multilingual, Multiview Text Categorization Test collection. UCI Machine Learning Repository, 2013.

[4] Avi Arampatzis, Georgios Peikos, and Symeon Symeonidis. Pseudo relevance feedback optimization. *Information Retrieval Journal*, pages 269–297, 2021.

[5] John Bace. Cost of e-discovery threatens to skew justice system. *Gartner RAS Core Research Note G*, 2007.

[6] Siebe Bloembergen and Johannes C Scholtes. System and method for near and exact de-duplication of documents, Sept 2012.

[7] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[8] Tianrun Cai, Zeling He, Chuan Hong, Yichi Zhang, Yuk-Lam Ho, Jacqueline Honerlaw, Alon Geva, Vidul Ayakulangara Panickan, Amanda King, David R Gagnon, et al. Scalable relevance ranking algorithm via semantic similarity assessment improves efficiency of medical chart review. *Journal of Biomedical Informatics*, page 104109, 2022.

[9] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 243–250, 2008.

[10] Christine P Chai. Comparison of text preprocessing methods. *Natural Language Engineering*, pages 509–553, 2023.

[11] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The Muppets straight out of Law School. *arXiv preprint arXiv:2010.02559*, 2020.

[12] Dhivya Chandrasekaran and Vijay Mago. Evolution of semantic similarity—a survey. *ACM Computing Surveys*, pages 1–37, mar 2022.

[13] Uttam Chauhan and Apurva Shah. Topic modeling using Latent Dirichlet Allocation: A survey. *ACM Computing Surveys (CSUR)*, pages 1–35, 2021.

[14] Jack Conrad. E-Discovery Revisited: A Broader Perspective for IR Researchers. In *"DESI Workshop"*, 2007.

[15] Gordon V Cormack and Maura R Grossman. Autonomy and reliability of continuous active learning for technology-assisted review. *arXiv preprint arXiv:1504.06868*, 2015.

[16] Gordon V Cormack and Maura R Grossman. Scalability of continuous active learning for reliable high-recall text classification. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 1039–1048, 2016.

[17] Corporate Counsel. The American Bar Association (ABA), section of litigation, committee on Corporate Counsel. *URL http://www. abanet. org/litigation/committees/corporate*, 2006.

[18] Robert Dale. Law and word order: NLP in legal tech. *Natural Language Engineering*, pages 211–217, 2019.

[19] Fan Deng, Stefan Siersdorfer, and Sergej Zerr. Efficient jaccard-based diversity analysis of large document collections. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1402–1411, 2012.

[20] Vinay Deolalikar. How valuable is your data? A quantitative approach using data mining. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1248–1253. IEEE, 2015.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[22] Bill Dim. More efficient e-discovery with document clustering, March 2023. `http://www.cluster-text.com/electronic-discovery.php`.

[23] David Dowling. Tarpits: The Sticky Consequences of Poorly Implementing Technology-Assisted Review. *Berkeley Tech. LJ*, page 171, 2020.

[24] Apache Software Foundation. Apache lucene - scoring, 2011. letzter Zugriff: 20. Oktober 2011.

[25] James Gorman and James R Curran. Scaling distributional similarity to large corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 361–368, 2006.

[26] Ryan Greene. New and improved embedding model, June 2023. `https://openai.com/blog/new-and-improved-embedding-model`.

[27] Maura R Grossman and G Cormack. Continuous active learning for TAR. *The Journal*, pages 1–7, 2016.

[28] Maura R Grossman and Gordon V Cormack. Technology-assisted review in electronic discovery. *Data Analysis in Law*, pages 1–49, 2018.

[29] Rentao Gu, Zeyuan Yang, and Yuefeng Ji. Machine learning for intelligent optical networks: A comprehensive survey. *Journal of Network and Computer Applications*, page 102576, 2020.

[30] Duarte G Henriques. The Prague rules: competitor, alternative or addition to the IBA rules on the taking of evidence in international arbitration? *ASA Bulletin*, pages 351–366, 2018.

[31] Hans Henseler and Harm van Beek. ChatGPT as a Copilot for Investigating Digital Evidence, 2023.

[32] Jacqueline Hoelting. Skin in the Game: Litigation Incentives Changing as Courts Embrace a Loser Pays Rule for E-Discovery Costs. *Clev. St. L. Rev.*, page 1103, 2012.

[33] Paul Jaccard. Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.*, pages 223–270, 1908.

[34] Sachindra Joshi, Danish Contractor, Kenney Ng, Prasad M Deshpande, and Thomas Hampp. Auto-Grouping Emails for Faster e-Discovery. *Proc. VLDB Endow.*, page 1284–1294, jun 2020.

[35] Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. *ACM Transactions on Information Systems*, pages 1–40, 2023.

[36] Peng-Cheng Lin and Wan-Lei Zhao. Graph based nearest neighbor search: Promises and failures. *International Conference on Machine Learning*, pages 119–130, 2019.

[37] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. Summary of ChatGPT research and perspective towards the future of large language models. *arXiv preprint arXiv:2304.01852*, 2023.

[38] lucene.apache.org. Solr, June 2023. `http://lucene.apache.org/solr/`.

[39] Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Computer Society*, page 824–836, apr 2020.

[40] Christopher D Manning. *Introduction to information retrieval*. Syngress Publishing, 2008.

[41] Douglas W Oard, Fabrizio Sebastiani, and Jyothi K Vinjumur. Jointly minimizing the expected costs of review for responsiveness and privilege in e-discovery. *ACM Transactions on Information Systems (TOIS)*, pages 1–35, 2018.

[42] Douglas W. Oard and William Webber. Information retrieval for e-discovery. *Foundations and Trends® in Information Retrieval*, pages 99–237, 2013.

[43] Nicole Peinelt, Dong Nguyen, and Maria Liakata. tBERT: Topic models and BERT joining forces for semantic similarity detection. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7047–7055, 2020.

[44] Amelia Phillips, Ronald Godfrey, Christopher Steuart, and Christine Brown. *E-discovery: an introduction to digital evidence*. Cengage Learning, 2013.

[45] Thomas Prikkel. Reducing manual labor in technology-assisted review. Master's thesis, Leiden University, Netherlands, 2018.

[46] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, page 1, 2012.

[47] Nils Reimers. sentence-transformers, March 2023. `https://www.sbert.net/docs/publications.html`.

[48] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *CoRR*, pages 3980–3990, 2019.

[49] Stephen Robertson. Understanding Inverse Document Frequency: On Theoretical Arguments for IDF. *Journal of Documentation - J DOC*, pages 503–520, 10 2004.

[50] Joseph John Rocchio. Document Retrieval System-Optimization and Evaluation. *DIR 2009 Dutch-Belgian Information Retrieval Workshop*, page 99, 2009.

[51] Herbert L Roitblat. Probably Reasonable Search in eDiscovery. *arXiv e-prints*, page 2201, 2022.

[52] Herbert L Roitblat, Anne Kershaw, and Patrick Oot. Document categorization in legal electronic discovery: computer classification vs. manual review. *Journal of the American Society for Information Science and Technology*, pages 70–80, 2010.

[53] David Sánchez and Montserrat Batet. A semantic similarity method based on information content exploiting multiple ontologies. *Expert Systems with Applications*, pages 1393–1399, 2013.

[54] Robin M Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*, 2019.

[55] Johannes C Scholtes and Hendrik Jacob van den Herik. Big data analytics for e-discovery. In *Research Handbook on Big Data Law*, pages 253–284. Edward Elgar Publishing, 2021.

[56] Saurabh Singh. Adversarial role of magistrate in common law countries. *Issue 3 Int'l JL Mgmt. & Human.*, page 1031, 2022.

[57] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, pages 35–43, 2001.

[58] Michael Sperling, Rong Jin, Illya Rayvych, Jianghong Li, and Jinfeng Yi. Similar Document Detection and Electronic Discovery: So Many Documents, So Little Time.

[59] TREC. TREC Legal Track, December 2022. `https://trec-legal.umiacs.umd.edu/`.

[60] John Tredennick. *TAR for Smart People*. Catalyst, 2015.

[61] John Tredennick. Summary of ChatGPT research and perspective towards the future of large language models. *arXiv e-prints*, pages arXiv–2304, April 2023.

[62] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.

[64] Eugene Yang, David Grossman, Ophir Frieder, and Roman Yurchak. Effectiveness results for popular e-discovery algorithms. In *Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law*, pages 261–264, 2017.

[65] Eugene Yang, David D Lewis, Ophir Frieder, David A Grossman, and Roman Yurchak. Retrieval and Richness when Querying by Document. In *DESIRES*, pages 68–75, 2018.

[66] Eugene Yang, Sean MacAvaney, David D. Lewis, Ophir Frieder, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty. Goldilocks: Just-Right Tuning of BERT for Technology-Assisted Review. In *Advances in Information Retrieval*, pages 502–517. Springer International Publishing, 2022.

[67] John T Yip. Addressing the costs and comity concerns on international E-discovery. *Wash. L. Rev.*, page 595, 2012.

[68] Haotian Zhang, Gordon V Cormack, Maura R Grossman, and Mark D Smucker. Evaluating sentence-level relevance feedback for high-recall information retrieval. *Information Retrieval Journal*, pages 1–26, 2020.

[69] Jie Zou and Evangelos Kanoulas. Towards question-based high-recall information retrieval: Locating the last few relevant documents for technology-assisted reviews. *ACM Transactions on Information Systems (TOIS)*, pages 1–35, 2020.

Appendices

A Results

This Appendix contains results of the text similarity experiments and the relevance feedback experiments.

A.1 Text similarity methods

This section of the Appendix contains results related to the text similarity experiments. For this, the first subsection contains the results for Quorum search. The second subsection contains the results for the TF-IDF based approach. The third subsection contains the results for Dense Vector Search. The final subsection contains the results of all these approaches on the ambiguous dataset.

A.1.1 Quorum search

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.973	0.037	0.071
20	0.967	0.070	0.130
50	0.948	0.166	0.282
100	0.917	0.318	0.472
200	0.837	0.577	0.683
300	0.718	0.741	0.729
500	0.505	0.867	0.638

Table 7: Results on the document level (N=100% and M=7.5%)

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.944	0.040	0.077
20	0.928	0.076	0.140
50	0.887	0.176	0.292
100	0.835	0.327	0.468
200	0.736	0.570	0.640
300	0.622	0.710	0.661
500	0.466	0.835	0.598

Table 8: Results on the paragraph level (querying first paragraph), documents ranked on the first paragraph (N=40%, M=10%).

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.938	0.043	0.082
20	0.917	0.080	0.146
50	0.872	0.184	0.301
100	0.812	0.339	0.471
200	0.705	0.567	0.620
300	0.600	0.695	0.639
500	0.454	0.820	0.584

Table 9: Results on the paragraph level (querying random paragraph), documents ranked on the first paragraph (N=40%, M=10%).

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.738	0.031	0.059
20	0.730	0.058	0.106
50	0.690	0.132	0.221
100	0.643	0.244	0.352
200	0.576	0.435	0.493
300	0.511	0.568	0.536
500	0.417	0.725	0.529

Table 10: Results on the paragraph level (querying first paragraph), documents ranked on paragraph paragraph count (N=40%, M=10%).

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.710	0.030	0.057
20	0.710	0.056	0.104
50	0.660	0.127	0.212
100	0.617	0.235	0.339
200	0.555	0.422	0.477
300	0.490	0.547	0.515
500	0.406	0.703	0.515

Table 11: Results on the paragraph level (querying random paragraph), documents ranked on paragraph count (N=40%, M=10%).

A.1.2 TF-IDF based approach

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.969	0.042	0.079
20	0.962	0.079	0.145
50	0.946	0.187	0.311
100	0.922	0.360	0.517
200	0.857	0.661	0.745
300	0.748	0.845	0.793
500	0.545	0.949	0.691

Table 12: Results on the document level using maxDf=0.8 and minDf=0

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.890	0.038	0.073
20	0.872	0.071	0.131
50	0.828	0.163	0.270
100	0.778	0.300	0.429
200	0.681	0.512	0.579
300	0.578	0.647	0.605
500	0.419	0.782	0.541

Table 13: Results on the paragraph level using maxDf=0.8 and minDf=0 (querying first paragraph), documents ranked on the first paragraph

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.859	0.038	0.073
20	0.837	0.071	0.130
50	0.793	0.161	0.265
100	0.736	0.292	0.414
200	0.637	0.490	0.548
300	0.541	0.609	0.569
500	0.402	0.739	0.518

Table 14: Results on the paragraph level using maxDf=0.8 and minDf=0 (querying random paragraph), documents ranked on the first paragraph

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.879	0.038	0.072
20	0.863	0.070	0.129
50	0.820	0.160	0.266
100	0.769	0.294	0.421
200	0.672	0.501	0.569
300	0.571	0.633	0.595
500	0.415	0.768	0.535

Table 15: Results on the paragraph level using maxDf=0.8 and minDf=0 (querying first paragraph), documents ranked on paragraph count

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.855	0.039	0.075
20	0.830	0.073	0.133
50	0.779	0.165	0.269
100	0.720	0.297	0.414
200	0.625	0.492	0.544
300	0.533	0.612	0.565
500	0.400	0.749	0.519

Table 16: Results on the paragraph level using maxDf=0.8 and minDf=0 (querying random paragraph), documents ranked on paragraph count

A.1.3 Dense vector search

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.978	0.036	0.070
20	0.974	0.069	0.129
50	0.964	0.166	0.283
100	0.948	0.323	0.482
200	0.906	0.614	0.732
300	0.812	0.824	0.818
500	0.560	0.946	0.703

Table 17: Results using all-MiniLM-L6-v2 (querying the first paragraph), documents ranked on first paragraph

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.977	0.037	0.071
20	0.972	0.070	0.130
50	0.961	0.168	0.286
100	0.946	0.327	0.486
200	0.904	0.622	0.737
300	0.809	0.834	0.821
500	0.555	0.951	0.701

Table 18: Results using all-MiniLM-L12-v2 (querying the first paragraph), documents ranked on first paragraph.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.980	0.037	0.072
20	0.976	0.071	0.132
50	0.967	0.171	0.290
100	0.954	0.333	0.494
200	0.921	0.640	0.755
300	0.822	0.856	0.839
500	0.563	0.971	0.713

Table 19: Results using all-mpnet-base-v2 (querying the first paragraph), documents ranked on first paragraph.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.971	0.037	0.071
20	0.963	0.069	0.130
50	0.949	0.166	0.283
100	0.928	0.322	0.478
200	0.880	0.608	0.719
300	0.784	0.811	0.797
500	0.547	0.940	0.691

Table 20: Results using all-MiniLM-L6-v2 (querying a random paragraph), documents ranked on first paragraph.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.968	0.037	0.071
20	0.961	0.070	0.130
50	0.946	0.167	0.283
100	0.925	0.323	0.478
200	0.878	0.609	0.719
300	0.781	0.811	0.796
500	0.542	0.937	0.687

Table 21: Results using all-MiniLM-L12-v2 (querying a random paragraph), documents ranked on first paragraph.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.973	0.037	0.072
20	0.968	0.071	0.132
50	0.956	0.170	0.289
100	0.940	0.332	0.491
200	0.902	0.634	0.745
300	0.802	0.843	0.822
500	0.553	0.964	0.703

Table 22: Results using all-mpnet-base-v2 (querying a random paragraph), documents ranked on first paragraph.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.814	0.031	0.060
20	0.846	0.062	0.116
50	0.823	0.147	0.249
100	0.803	0.284	0.419
200	0.764	0.537	0.631
300	0.696	0.733	0.714
500	0.541	0.942	0.687

Table 23: Results using all-mpnet-base-v2 (querying the first paragraph), documents ranked on counting paragraphs.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.718	0.027	0.052
20	0.772	0.055	0.103
50	0.763	0.133	0.227
100	0.740	0.256	0.380
200	0.703	0.483	0.573
300	0.652	0.671	0.661
500	0.529	0.907	0.668

Table 24: Results using all-MiniLM-L12-v2 (querying the first paragraph), documents ranked on paragraph count.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.714	0.027	0.051
20	0.775	0.055	0.103
50	0.759	0.132	0.225
100	0.743	0.256	0.380
200	0.706	0.483	0.574
300	0.653	0.669	0.660
500	0.535	0.912	0.674

Table 25: Results using all-MiniLM-L6-v2 (querying the first paragraph), documents ranked on paragraph count.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.820	0.032	0.061
20	0.848	0.062	0.116
50	0.822	0.147	0.250
100	0.797	0.283	0.417
200	0.754	0.533	0.624
300	0.687	0.727	0.706
500	0.536	0.939	0.682

Table 26: Results using all-mpnet-base-v2 (querying a random paragraph), documents ranked on paragraph count.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.746	0.028	0.055
20	0.790	0.057	0.107
50	0.770	0.135	0.230
100	0.743	0.259	0.384
200	0.699	0.484	0.572
300	0.646	0.671	0.658
500	0.522	0.901	0.661

Table 27: Results using all-MiniLM-L12-v2 (querying a random paragraph), documents ranked on paragraph count.

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.732	0.028	0.053
20	0.778	0.056	0.105
50	0.755	0.133	0.226
100	0.734	0.255	0.379
200	0.692	0.479	0.566
300	0.639	0.662	0.651
500	0.522	0.899	0.660

Table 28: Results using all-MiniLM-L6-v2 (querying a random paragraph), documents ranked on paragraph count.

A.1.4 Ambiguous data

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.867	0.032	0.061
20	0.829	0.057	0.107
50	0.769	0.130	0.222
100	0.707	0.236	0.354
200	0.619	0.410	0.494
300	0.549	0.546	0.547
500	0.446	0.737	0.556

Table 29: Results of Quorum search on the document level with N=100% and M=7.5%

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.867	0.037	0.072
20	0.844	0.069	0.128
50	0.799	0.159	0.265
100	0.750	0.293	0.421
200	0.668	0.513	0.580
300	0.592	0.673	0.629
500	0.475	0.862	0.612

Table 30: Results of the TF-IDF based approach on the document level with maxDf=0.8 and minDf=0

Cutoff@	Macro Precision	Macro Recall	Macro F1 score
10	0.914	0.035	0.068
20	0.896	0.066	0.123
50	0.871	0.156	0.264
100	0.840	0.297	0.439
200	0.779	0.548	0.643
300	0.691	0.727	0.708
500	0.519	0.910	0.661

Table 31: Results of DVS using *all-mpnet-base v2*, querying the first paragraph, and ranking documents based on the first paragraph

A.2 Relevance feedback

This section of the Appendix contains results related to the relevance feedback experiments. The results display the recall for a given iteration (where every iteration resembles a review effort of 10 results). Table 32 refers to the experiments on the test set and Table 33 refers to the experiments on the ambiguous set.

Iteration	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75
No feedback	0.114	0.262	0.408	0.548	0.674	0.769	0.830	0.867	0.894	0.914	0.930	0.943	0.952	0.960	0.966
Keyword exp.	0.111	0.256	0.399	0.538	0.663	0.763	0.825	0.866	0.897	0.917	0.933	0.946	0.956	0.966	0.972
Rocchio	0.109	0.249	0.384	0.505	0.611	0.698	0.766	0.821	0.861	0.892	0.920	0.942	0.961	0.973	0.980
Average	**0.115**	0.262	0.411	0.551	0.686	0.801	0.908	0.960	0.974	0.981	0.986	0.988	0.991	0.993	0.995
Average (amp)	0.113	0.254	0.396	0.534	0.667	0.778	0.886	0.950	0.969	0.976	0.983	0.986	0.988	0.991	0.993
Sum	0.114	**0.267**	**0.424**	**0.575**	**0.725**	**0.854**	**0.945**	**0.971**	0.984	0.988	0.991	0.993	0.994	0.996	0.996
Sum (amp)	0.113	0.262	0.416	0.569	0.717	0.848	0.943	0.967	0.979	0.986	0.990	0.993	0.994	0.996	0.997
Average	0.111	0.243	0.387	0.529	0.668	0.793	0.901	0.977	**0.989**	**0.990**	0.991	0.994	**0.997**	0.998	**0.999**
Average (amp)	0.108	0.240	0.372	0.513	0.645	0.765	0.871	0.960	0.987	0.989	0.991	0.993	**0.997**	**0.999**	**0.999**
Sum	0.108	0.242	0.381	0.524	0.662	0.788	0.896	0.966	0.985	0.987	0.990	**0.995**	0.996	0.997	0.998
Sum (amp)	0.108	0.242	0.380	0.522	0.652	0.774	0.874	0.958	**0.989**	**0.990**	**0.993**	**0.995**	0.996	0.997	0.998

Table 32: Relevance feedback results on the test set. Top cells are baselines. Middle cells are cumulative strategies. Bottom cells are non-cumulative strategies.

Iteration	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75
No feedback	0.107	0.232	0.351	0.462	0.556	0.637	0.700	0.749	0.792	0.828	0.859	0.885	0.908	0.926	0.941
Keyword exp.	0.106	0.231	0.345	0.448	0.542	0.626	0.693	0.752	0.801	0.842	0.875	0.901	0.923	0.939	0.951
Rocchio	0.109	0.249	0.384	0.505	0.611	0.698	0.766	0.821	0.861	0.892	0.920	0.942	0.961	0.973	0.980
Average	0.116	0.267	0.418	0.553	0.659	0.763	0.826	0.856	0.876	0.902	0.930	0.951	0.965	0.973	0.980
Average (amp)	0.118	0.269	0.416	0.543	0.658	0.763	0.826	0.858	0.877	0.897	0.918	0.940	0.959	0.971	0.978
Sum	**0.119**	**0.275**	**0.431**	**0.573**	**0.703**	**0.803**	**0.870**	**0.907**	**0.932**	**0.947**	**0.963**	**0.976**	**0.985**	**0.988**	**0.993**
Sum (amp)	0.117	0.271	0.423	0.564	0.691	0.794	0.861	0.902	0.929	0.945	0.962	0.976	0.985	0.987	0.992
Average	0.110	0.251	0.390	0.488	0.607	0.707	0.780	0.829	0.865	0.898	0.931	0.946	0.963	0.972	0.976
Average (amp)	0.110	0.254	0.378	0.494	0.615	0.725	0.791	0.835	0.868	0.889	0.916	0.937	0.956	0.969	0.974
Sum	0.110	0.248	0.377	0.490	0.584	0.683	0.767	0.819	0.855	0.883	0.912	0.938	0.955	0.963	0.968
Sum (amp)	0.106	0.241	0.354	0.463	0.584	0.707	0.788	0.828	0.855	0.884	0.916	0.943	0.965	0.972	0.977

Table 33: Relevance feedback results on the ambiguous set. Top cells are baselines. Middle cells are cumulative strategies. Bottom cells are non-cumulative strategies.

B Data

This Appendix contains information related to the data used in the experiments.

B.1 Reuters RCV1 v2

This section of the Appendix contains information related to the RCV-1 v2 dataset from Reuters, which was used in all of the experiments. In this section, Table 34 shows the data of the sample used in the experiments. Next, Table 35 shows the complete contents of the RCV1 v2 dataset.

Subset	Topic	Prevalence within subset
Train	RELIGION	300
Train	TRAVEL AND TOURISM	300
Train	UNEMPLOYMENT	300
Train	ADVERTISING/PROMOTION	304
Train	HEALTH	305
Validation	METALS TRADING	300
Validation	WEATHER	300
Validation	FASHION	300
Validation	CRIME, LAW ENFORCEMENT	304
Validation	ARTS, CULTURE, ENTERTAINMENT	307
Test	SPORTS	302
Test	STRATEGY/PLANS	303
Test	REGULATION/POLICY	308
Test	WAR, CIVIL WAR	311
Test	ELECTIONS	312
Ambiguous	EC CORPORATE POLICY	300
Ambiguous	FOREX MARKETS	300
Ambiguous	ENERGY MARKETS	300
Ambiguous	EC INTERNAL MARKET	311

Table 34: Topics and prevalence of the sampled articles

Topic	#	%		Topic	#	%
CORPORATE/INDUSTRIAL	443808	46,83		PRIVATISATIONS	8812	0,93
GOVERNMENT/SOCIAL	286178	30,2		CREDIT RATINGS	8686	0,92
MARKETS	217550	22,95		SHARE LISTINGS	8674	0,92
PERFORMANCE	169358	17,87		EXTERNAL MARKETS	8506	0,9
ECONOMICS	141130	14,89		NEW PRODUCTS/SERVICES	8018	0,85
COMMODITY MARKETS	94884	10,01		INFLATION/PRICES	7720	0,81
ACCOUNTS/EARNINGS	88968	9,39		BIOGRAPHIES, PERSONALITIES, PEOPLE	7550	0,8
COMMENT/FORECASTS	85922	9,07		HEALTH	7366	0,78
DOMESTIC POLITICS	68922	7,27		ENVIRONMENT AND NATURAL WORLD	7066	0,75
OWNERSHIP CHANGES	61702	6,51		CONSUMER PRICES	6938	0,73
MONEY MARKETS	57920	6,11		LEADING INDICATORS	6348	0,67
EQUITY MARKETS	53554	5,65		ASSET TRANSFERS	5626	0,59
SOFT COMMODITIES	52784	5,57		MONOPOLIES/COMPETITION	5578	0,59
MERGERS/ACQUISITIONS	52362	5,52		DOMESTIC MARKETS	5456	0,58
FUNDING/CAPITAL	51714	5,46		EC EXTERNAL RELATIONS	5374	0,57
GOVERNMENT FINANCE	50728	5,35		ARTS, CULTURE, ENTERTAINMENT	4686	0,49
MARKETS/MARKETING	50372	5,31		EC INTERNAL MARKET	4212	0,44
INTERNATIONAL RELATIONS	46568	4,91		WEATHER	3982	0,42
REGULATION/POLICY	45530	4,8		BALANCE OF PAYMENTS	3694	0,39
SPORTS	42726	4,51		RELIGION	3492	0,37
WAR, CIVIL WAR	41594	4,39		HUMAN INTEREST	3368	0,36
CAPACITY/FACILITIES	39128	4,13		RESEARCH/DEVELOPMENT	3128	0,33
CRIME, LAW ENFORCEMENT	38274	4,04		EC AGRICULTURE POLICY	3024	0,32
GOVERNMENT BORROWING	33272	3,51		LOANS/CREDITS	2982	0,31
INTERBANK MARKETS	32054	3,38		OUTPUT/CAPACITY	2790	0,29
PRODUCTION/SERVICES	31638	3,34		EC CORPORATE POLICY	2724	0,29
MONETARY/ECONOMIC	31436	3,32		MONEY SUPPLY	2716	0,29
STRATEGY/PLANS	29986	3,16		EC INSTITUTIONS	2684	0,28
FOREX MARKETS	28208	2,98		UNEMPLOYMENT	2630	0,28
ANNUAL RESULTS	27700	2,92		SCIENCE AND TECHNOLOGY	2626	0,28
BOND MARKETS	27110	2,86		INSOLVENCY/LIQUIDITY	2506	0,26
TRADE/RESERVES	24912	2,63		CONSUMER FINANCE	2446	0,26
ENERGY MARKETS	24596	2,6		RESERVES	2354	0,25
EUROPEAN COMMUNITY	23506	2,48		EC COMPETITION/SUBSIDY	2316	0,24
SHARE CAPITAL	22000	2,32		WELFARE, SOCIAL SERVICES	2242	0,24
LABOUR ISSUES	21684	2,29		ADVERTISING/PROMOTION	2104	0,22
EMPLOYMENT/LABOUR	21162	2,23		INDUSTRIAL PRODUCTION	2038	0,22
CONTRACTS/ORDERS	18620	1,96		DEFENCE CONTRACTS	1532	0,16
EXPENDITURE/REVENUE	17674	1,86		RETAIL SALES	1456	0,15
MERCHANDISE TRADE	15760	1,66		MARKET SHARE	1384	0,15
LABOUR	15270	1,61		WHOLESALE PRICES	1110	0,12
ELECTIONS	13722	1,45		OBITUARIES	1044	0,11
MANAGEMENT	13670	1,44		TRAVEL AND TOURISM	916	0,1
BONDS/DEBT ISSUES	13574	1,43		HOUSING STARTS	504	0,05
LEGAL/JUDICIAL	13390	1,41		PERSONAL INCOME	412	0,04
METALS TRADING	13118	1,38		EC ENVIRONMENT ISSUES	370	0,04
MANAGEMENT MOVES	12398	1,31		FASHION	322	0,03
ECONOMIC PERFORMANCE	10482	1,11		CONSUMER CREDIT	236	0,02
DEFENCE	10470	1,1		INVENTORIES	132	0,01
DISASTERS AND ACCIDENTS	9798	1,03		CAPACITY UTILIZATION	64	0,01
EC MONETARY/ECONOMIC	9786	1,03		EC GENERAL	64	0,01

Table 35: All topics in the RCV-1 dataset sorted by prevalence (in count and percentage)