Relevance feedback strategies for reducing review effort in recall-oriented neural information retrieval

Timo Kats[1 [0000−0003−1650−1814]], Peter van der Putten[1 [0000−0002−6507−6896]], and Jan Scholtes[2 [0000−0003−0534−2491]]

[1] LIACS, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands
t.p.a.kats@liacs.leidenuniv.nl, p.w.h.van.der.putten@liacs.leidenuniv.nl
[2] Maastricht University, Minderbroedersberg 4-6, 6211 LK Maastricht, The Netherlands
scholtes@msc.nl

Abstract. In a number of information retrieval applications, such as patent search, literature review, and due diligence, preventing false negatives is more important than preventing false positives. However, approaches designed to reduce review effort, such as 'technology assisted review', can create false negatives, since these are often based on active learning systems that exclude documents automatically based on user feedback. To address this issue, we propose a recall-oriented approach to reducing review effort, through iteratively re-ranking the relevance rankings based on user feedback. We propose and experiment with various relevance feedback strategies, and in our best-performing method relevance rankings are produced by a BERT-based dense-vector search, and 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%, compared to a baseline approach of no feedback, given a fixed recall target.

Keywords: high recall information retrieval · dense-vector search · relevance feedback · neural information retrieval

1 Introduction

In a diverse range of information retrieval applications, reaching certain minimal recall goals is key, when identifying which documents in a large corpus are relevant to an information need. This is particularly important in various contexts, such as intelligence and law enforcement, searching for adverse effects of medicines, patent due diligence, and legal search applications. In these scenarios, preventing false negatives often takes precedence over preventing false positives [26].

Manually reviewing all documents is cost intensive and error-prone [20]. Therefore, technology is often used to reduce review effort. However, technologies aimed at reducing review effort, such as 'technology assisted review' can create

false negatives, since they often rely on active learning systems that exclude documents automatically based on user feedback [12].

Therefore, this research aims to evaluate a more recall-oriented approach to reducing review effort while cumulatively identifying relevant documents. For this, we combine two major components: textual similarity and relevance feedback. Textual similarity is used to produce relevance rankings, while relevance feedback is used to iteratively refine these relevance rankings based on user feedback. Hence, our first research question is formulated as: 'what text similarity methods are most suitable for relevance feedback?' Building upon this, our second research question is: to what extent can relevance feedback contribute to reducing review effort?

For textual similarity, we evaluate TF-IDF and BERT-based dense-vector representations. For relevance feedback, we compare text-based and vector-based feedback strategies. Additionally, we experiment with varying levels of textual granularity in both components. In the text similarity component this is done through paragraph-based document rankings and in the relevance feedback component this is done through feedback amplification.

This combination of representation methods, feedback strategies and levels of granularity has elements that contribute to scientific novelty, and leads to substantial improvements in results. With these two components, given a recall target of 80%, our proposed method reduces review effort between 17.85% and 59.04% compared to our baseline approach of no relevance feedback.

The remainder of this paper is structured as follows. Section 2 discusses related work. Next, we describe our methodology (section 3) and experimental setup (section 4). In section 5 we share the results of our experiments, and the paper ends with a discussion section (section 6) and conclusion (section 7).

2 Background

In this section we discuss work related to the two main components of our method: text similarity methods and relevance feedback strategies. For the text similarity methods we discuss term-based and context-based approaches. Next, for the relevance feedback strategies, we review text-based and vector-based strategies.

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 [7]. As a result, these methods don't incorporate contextual language properties like homonymy or synonymy in their similarity computations.

A simple and commonly used [14] implementation of term-based similarity is Jaccard similarity [13], which computes text similarity based on how many features two texts have in common divided by the total number of features across both texts:

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

with A and B the sets of unique words for both documents.

However, a shortcoming of Jaccard is that frequent terms (that are more likely to match) within a document are unlikely to be distinctive or important [6]. To deal with this shortcoming, there are two approaches. The first approach is based on manually filtering out frequent words based on a predefined 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 generally has a beneficial effect [6], though this is task dependent ([28]. Hence, we filter out stop words in our experiment.

The second approach is based on reducing the influence of words that are common in a corpus [6]. An advantage of this method compared to filtering stop words is that it's more dynamic. Certain words that are common within a specific context (e.g., the word "patient" in medical data) might not be included in predefined lists of stop words. This approach is implemented in our TF-IDF-based similarity method through *inverse document frequency*. Terms are given a measure of uniqueness by dividing the number of documents in total by the number 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 [25].

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

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 semantics. This is based on the "distributional hypothesis" [11], which is built on the idea that "words that occur in the same contexts tend to have similar meanings" [7]. 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) [7].

In this category of pre-trained word embeddings, we specifically focus on BERT [9], [29]. The transformer-based architecture of BERT allows it to capture context-sensitive word properties (like homonymy and synonymy) in its

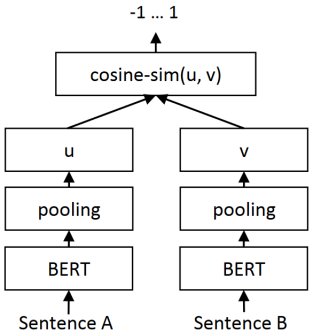

Fig. 1: Simplified architecture of SBERT [22]

embeddings. There are versions of BERT made for specific tasks. For example, Sentence-BERT (SBERT) is a BERT model specifically made for measuring text similarity using the cosine distance metric [22]. 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 or 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 [22]. As a result, this approach can't be implemented on a level of text granularity that exceeds this amount.

2.3 Text-based feedback strategies

Relevance feedback refers to changing the relevance ranking through user feedback, generally in multiple iterations [19]. A commonly used text-based relevance feedback strategy is "keyword expansion". Here, keywords from the selected search results are appended to the query. A frequently used implementation of this approach 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 research we will use this strategy for selecting keywords to expand our queries with.

2.4 Vector-based feedback 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

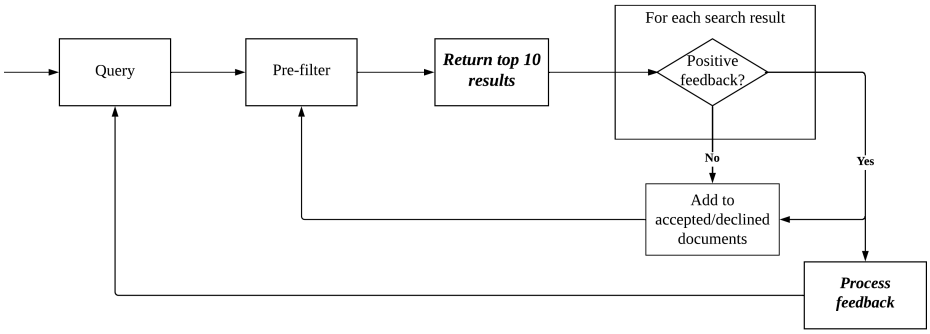

Fig. 2: Flowchart of the method

to as vector-based pseudo-relevance feedback. There are two commonly used methods for this [15]. 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 strategies are implemented in our research.

A commonly used variation of averaging query vectors is based on Rocchio's method for relevance feedback [15]. 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 [23]. The version of Rocchio implemented by most researchers today [5] [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.

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 2, 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. These methods are explained in the first and second subsections. Next, for processing the feedback given after each iteration, we experiment with different feedback strategies. These are explained in the third and fourth subsections.

3.1 TF-IDF-based text similarity

Our baseline method is based on TF-IDF, which 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. 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).

In our research we use TF-IDF to find similar documents using "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 [10].

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, we use MLT instead.

3.2 BERT-based text similarity

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 [21]. 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.

$$\textit{cosine similarity} = \cos(\theta) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \tag{3}$$

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 [18]. HNSW is an algorithm that finds the k most similar documents to a query with logarithmic time and space complexity (O(log(N)).

3.3 Paragraph-based document rankings

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 number of paragraphs per document in the ranking, and using that 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 number of paragraphs per document determines the ranking.

3.4 Baseline feedback strategies

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. If a user accepts/declines documents, the ranking of documents remains constant. This is referred to as "no feedback" or "original".

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 is 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. In our research, this selection is done based on the (highest) *inverse document frequency* value.

3.5 Vector-based feedback strategies

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).

Also, for text similarity methods implemented on the paragraph level, relevance feedback can be amplified to the document level. 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

This section provides an overview of the experiments and our data (and preprocessing). 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.

4.1 Data and preprocessing

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. Due to hardware constraints, we did not use the complete dataset in our experiment. Instead, we randomly sampled 300 articles per topic. For this, we sampled 15 (unrelated) topics that are equally split in train, validation, and test. We call this data set the "default set". Moreover, besides these topics we also sampled 4 topics that share the same parent topics. This set will be referred to as the "ambiguous" set.

As for preprocessing, in TF-IDF we filter out stop words, numbers, and convert the text to lowercase. The stop word list used for this is publicly available on GitHub[3]. For the SBERT-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 number of words in a paragraph can't exceed 384. Hence, we verified that the paragraphs in the topic sets do not exceed that limit.

4.2 Text similarity methods

For the text similarity methods, we compare two approaches: MLT (which is based on TF-IDF) and DVS (which is based on SBERT). For both methods, we iterate through our set using "query by document" (QBD). Each time we query a document/paragraph to return other documents/paragraphs that belong to the same topic.

For MLT, we set two parameters. First, the minimum document frequency for words to be considered (minDf). Second, the maximum document frequency for words to be considered (maxDf). Due to limited computing resources, we didn't conduct a full grid search to find the optimal values for these parameters. Instead,

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

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 1: Selected BERT models' characteristics according to [22]

we conducted a manual search on the train and validation sets and used the average of the most performant parameter values on our test (and ambiguous) set. Note, we conduct these experiments on the paragraph *and* document level.

For DVS, we don't have any parameters to set. Hence these experiments are conducted directly on our test set and our ambiguous set. However, we do experiment with three different pre-trained SBERT models. These are selected based on being the general-purpose models in the SBERT documentation [22]. An overview of these models can be found in Table 1. Note, due to SBERT's maximum context length, we conduct these experiments on the paragraph level only.

Finally, we evaluate the similarity methods based on (the macro averaged) recall, precision and F1 scores. 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 document returned (as similar) should also be annotated as "sports" to be considered a true positive.

4.3 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. An overview of the layout of the experiment can be found in Algorithm 1. 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. Finally, for performance evaluation, we record the average number of iterations needed to achieve exactly 80% recall (which is a commonly used threshold in information retrieval [24]).

4.4 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 [17], which is built on top of Lucene [10]. Finally, the source code used in this experiment is publicly available on GitHub[4].

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

Algorithm 1: Pseudo relevance feedback experiment

```
input  : paragraph, maxRecall
output: Iterations needed to achieve recall
iterations = 0
while recall(acceptedDocuments) ≤ maxRecall do
    filter = acceptedDocuments + declinedDocuments
    results = query(paragraph, filter)          // Returns top 10 results.
    for result in results do
        if feedback for result is positive then
            paragraph = processFeedback(result)
            acceptedDocuments += result
        else
            declinedDocuments += result
    iterations + = 1
return iterations
```

5 Results

This section provides 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 present the results of the different relevance feedback strategies.

5.1 Text similarity methods

For our TF-IDF-based approach, the manual search on the train and validation sets resulted in the parameter values of maxDf=0.8 and minDf=0 on both the paragraph and document level. Hence, the TF-IDF related results are based on these parameter values. Next, for the DVS experiments, *all-mpnet-base v2* was the best performing pre-trained SBERT model. Hence, the results of this model are shown in this section.

For our default test set, the recall-precision graphs are shown in Figure 3 and Figure 4. In both Figures, it's apparent that deriving a document ranking from its *highest* ranked paragraph outperforms ranking documents based on *counting* the paragraphs. Moreover, for both methods querying the first paragraph of a document slightly outperforms querying random paragraphs. Next, for the TF-IDF-based method specifically, the experiments on the document level outperform the experiments on the paragraph level.

Finally, when comparing the performance of the optimal configuration of both approaches, DVS (with a first-based ranking) outperforms all TF-IDF-based approaches. Next, when comparing the optimal configurations of the approaches on the *ambiguous* set, we again see that DVS outperforms the TF-IDF-based approach (see Figure 5).

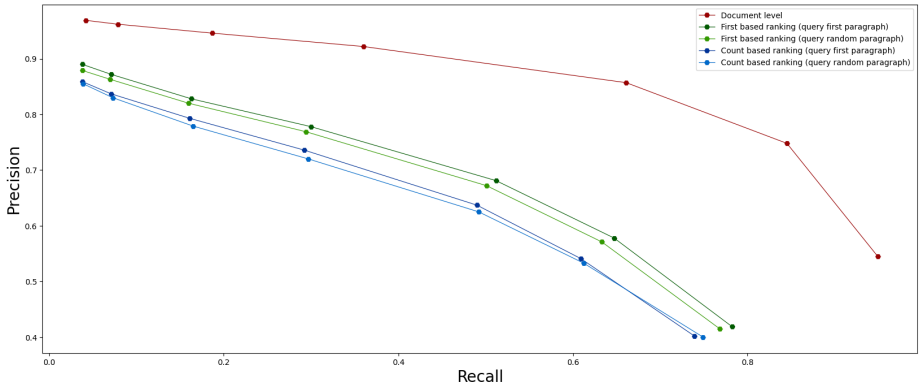

Fig. 3: Different configurations for the TF-IDF-based approach on the default set

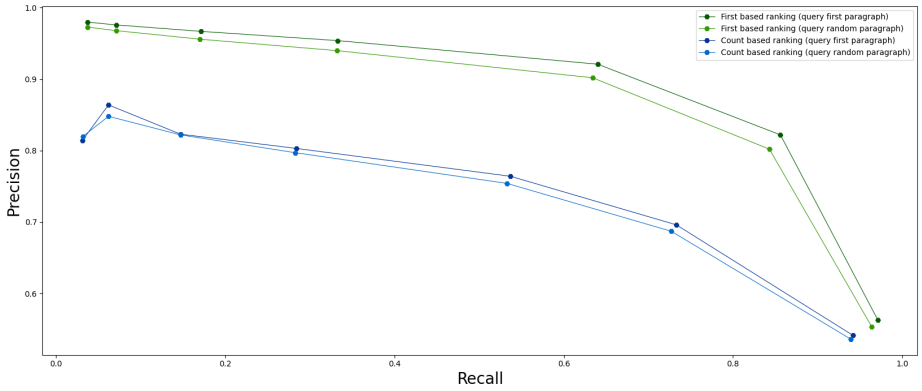

Fig. 4: Different configurations for DVS on the default set

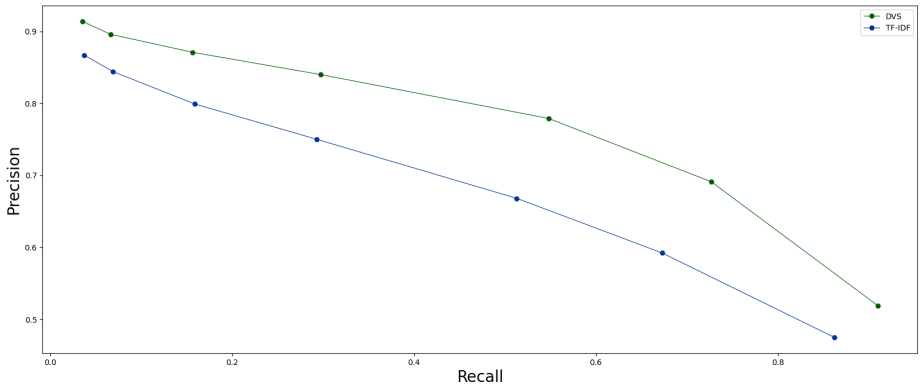

Fig. 5: Identified configurations of DVS and TF-IDF-based approach on the ambiguous set

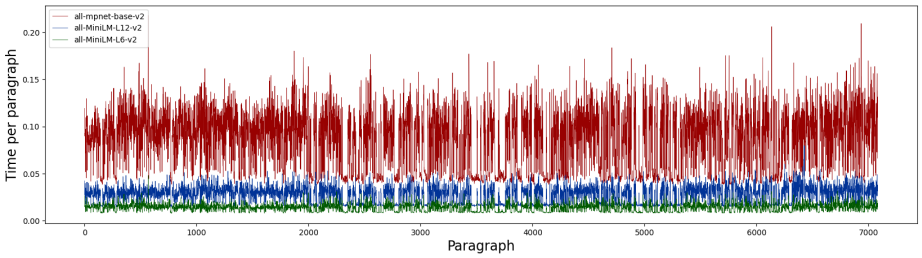

Fig. 6: Time consumed to create SBERT embeddings per paragraph

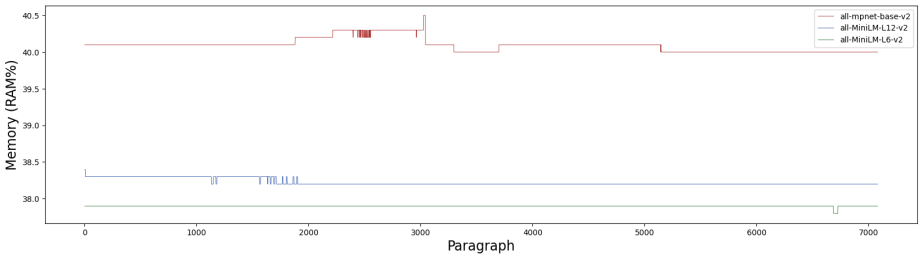

Fig. 7: RAM% consumed to create SBERT embeddings per paragraph

Note, for all pre-trained SBERT models in our experiments, the RAM usage per created embedding is shown in Figure 6 and the computation time per embedding is shown in Figure 7. 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 6, both these Figures confirm the pattern described in the documentation from SBERT (shown in Table 1), which is that the larger models are more time and memory consumptive. Hence, the results of *all-mpnet-base v2* shown in this section do have extra computational costs.

5.2 Relevance feedback

For the second set of experiments, we experimented with different feedback strategies using the identified text similarity method (DVS). For these experiments, the average number of iterations (and standard deviation) needed to achieve 80% recall for all methods and datasets is shown in Table 2. 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.

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 simi-

Feedback strategy	**Mean (Default)**	**Std Dev (Default)**	**Mean (Ambiguous)**	**Std Dev (Ambiguous)**
No feedback	33.36	8.84	46.77	10.04
Keyword expansion	32.24	8.03	44.23	9.94
Rocchio ($\alpha = 0.5$, $\beta = 0.5$)	29.50	3.79	37.98	5.11
Average	29.43	1.51	32.65	2.97
Average (amp)	30.17	1.03	32.99	1.80
Sum	***28.29***	0.75	***29.41***	2.13
Sum (amp)	28.46	***0.50***	30.54	***1.31***
Average	31.54	1.14	38.68	4.41
Average (amp)	32.81	1.08	38.41	2.64
Sum	32.80	1.12	38.80	5.28
Sum (amp)	32.20	0.87	38.20	3.77

Table 2: 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).

larity 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.

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 number 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 3. 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 3: Average execution times for different relevant feedback strategies

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 Text similarity methods

For all of text similarity methods, some common denominators emerge from the results. First, querying the first paragraph gives better results than querying a random paragraph. This coincides with findings from related work, since prior research found that the effectiveness of "query by document" approaches depends on the prevalence of relevant terms in the queried text [30]. 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 apply to the data used in other domains

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 TF-IDF-based approach. Considering our DVS approach is based on BERT instead of TF-IDF, this finding makes sense. Because, BERT's bidirectional self-attention mechanism [9] enables it to understand more ambiguous and context-sensitive language properties. For example, the usage of synonymy, homonymy and referrals.

However, it must also be noted that our findings are based on a pre-trained SBERT model (*all-mpnet-base v2*) that had higher computational costs with regards to RAM usage and computation time than the other pre-trained models in our experiment.

6.2 Relevance feedback

For the second experiment, the best performing text similarity method (DVS, with a first-based document ranking) 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 3 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.

6.3 Reflection on implementation and contributions

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 active learning based approach). This is particularly important in use-cases that require a high recall.

In terms of scientific novelty and contributions, we should state that the concept of relevance feedback has been studied before [31]. 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 [16]) 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 is novel. Moreover, a more "stable" reduction of review effort could be favorable in real-world scenarios.

Hence, our findings are not just novel, but also applicable and practically implementable.

6.4 Limitations

First, data in real-world information retrieval scenarios tends to be heterogeneous (e.g. emails, documents, direct messages, etc.) [27]. 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 real-world data this technique could be applied to. Thus, the lack of data heterogeneity in this experiment is a limitation of this research.

Second, the size of the dataset (only 300 articles per topic) is smaller than most real-world information retrieval scenarios. Moreover, our data only consists of news articles. Therefore, certain findings in our research (e.g., the fact that querying the first paragraph slightly outperforms querying random paragraphs) might not apply on other datasets. Still, given the fact that the RCV-1 v2 dataset is commonly used as a benchmark dataset in information retrieval [8], the results are still an adequate indication of our method's performance.

6.5 Future work

Our first suggestion for future work is related to the size of our dataset. As mentioned, due to computational constraints we only sampled 300 articles per topic. However, datasets in real-world information retrieval applications can be much larger than that. As a result, future work could run these methods on larger datasets to research their scalability.

Our second suggestion for future work is related to the level of data heterogeneity in the experiments. As mentioned in the previous subsection, data used in real-world information retrieval scenarios tend 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.

Finally, it will be interesting to further experiment with our relevance feedback strategies in combination with using large language models for the embeddings, or potentially also for pseudo or proxy feedback, both in the context of discovery and information retrieval tasks, as well as other tasks such as retrieval augmented generation.

7 Conclusion

This research evaluates the impact of changing the (text similarity-based) relevance rankings based on relevance feedback. For this, the first research question was formulated as follows: what text similarity methods are most suitable for relevance feedback? Our results show that the most suitable text similarity

method for this is DVS (using BERT-based dense-vector representations) where the highest ranked paragraph determines the document ranking.

Next, the second research question was formulated as follows: to what extent can relevance feedback help to reduce review effort? We have outlined a variety of relevance feedback strategies. Our results show that (compared to processing no relevance feedback) our best-scoring the relevance feedback strategy reduces review effort between 17.85% and 59.04%, given a target recall level of 80%.

Given the recall-oriented nature of many information retrieval applications [26], 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.

References

1. Adebayo, K.J.: Multimodal Legal Information Retrieval. Ph.D. thesis, University of Luxembourg, Luxembourg (2018)
2. Agundez, R.: ChunckDot Python library (May 2023), https://pypi.org/project/chunkdot/
3. Amini, Massih-Reza & Goutte, C.: Reuters RCV1 RCV2 Multilingual, Multiview Text Categorization Test collection. UCI Machine Learning Repository (2013)
4. Arampatzis, A., Peikos, G., Symeonidis, S.: Pseudo relevance feedback optimization. Information Retrieval Journal pp. 269–297 (2021)
5. Cai, T., He, Z., Hong, C., Zhang, Y., Ho, Y.L., Honerlaw, J., Geva, A., Panickan, V.A., King, A., Gagnon, D.R., et al.: Scalable relevance ranking algorithm via semantic similarity assessment improves efficiency of medical chart review. Journal of Biomedical Informatics p. 104109 (2022)
6. Chai, C.P.: Comparison of text preprocessing methods. Natural Language Engineering pp. 509–553 (2023)
7. Chandrasekaran, D., Mago, V.: Evolution of semantic similarity—a survey. ACM Computing Surveys pp. 1–37 (mar 2022)
8. Deolalikar, V.: How valuable is your data? A quantitative approach using data mining. In: 2015 IEEE International Conference on Big Data (Big Data). pp. 1248–1253. IEEE (2015)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1. pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019)
10. Foundation, A.S.: Apache Lucene - scoring (2011), http://lucene.apache.org/java/340/scoring.html
11. Gorman, J., Curran, J.R.: 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. pp. 361–368 (2006)

12. Grossman, M.R., Cormack, G.: Continuous active learning for TAR. The Journal pp. 1–7 (2016)
13. Jaccard, P.: Nouvelles recherches sur la distribution florale. Bull. Soc. Vaud. Sci. Nat. pp. 223–270 (1908)
14. Joshi, S., Contractor, D., Ng, K., Deshpande, P.M., Hampp, T.: Auto-grouping emails for faster e-discovery. Proc. VLDB Endow. p. 1284–1294 (June 2020)
15. Li, H., Mourad, A., Zhuang, S., Koopman, B., Zuccon, G.: Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. ACM Transactions on Information Systems pp. 1–40 (2023)
16. Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z., Wu, Z., Zhao, L., Zhu, D., Li, X., Qiang, N., Shen, D., Liu, T., Ge, B.: Summary of ChatGPT-related research and perspective towards the future of large language models. Meta-Radiology **1**(2), 100017 (2023)
17. lucene.apache.org: Solr (June 2023), http://lucene.apache.org/solr/
18. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. IEEE Computer Society p. 824–836 (apr 2020)
19. Manning, C.D.: Introduction to information retrieval. Syngress Publishing (2008)
20. Piramuthu, O.B.: Multiple choice online algorithms for technology-assisted reviews. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing. pp. 639–645 (2023)
21. Rahutomo, F., Kitasuka, T., Aritsugi, M.: Semantic cosine similarity. In: The 7th international student conference on advanced science and technology ICAST. p. 1 (2012)
22. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (Nov 2019)
23. Rocchio, J.J.: Document retrieval system-optimization and evaluation. DIR 2009 Dutch-Belgian Information Retrieval Workshop p. 99 (2009)
24. Roitblat, H.L.: Probably reasonable search in eDiscovery. arXiv e-prints p. 2201 (2022)
25. Sánchez, D., Batet, M.: A semantic similarity method based on information content exploiting multiple ontologies. Expert Systems with Applications pp. 1393–1399 (2013)
26. Song, J.J., Lee, W., Afshar, J.: An effective high recall retrieval method. Data & Knowledge Engineering **123**, 101603 (2019)
27. Sperling, M., Jin, R., Rayvych, I., Li, J., Yi, J.: Similar document detection and electronic discovery: So many documents, so little time. In: ICAIL 2013 Workshop on Standards for Using Predictive Coding, Machine Learning, and Other Advanced Search and Review Methods in E-Discovery (DESI V Workshop). Rome, Italy (2013)
28. Teernstra, L., van der Putten, P., Noordegraaf-Eelens, L., Verbeek, F.: The morality machine: Tracking moral values in tweets. In: Boström, H., Knobbe, A., Soares, C., Papapetrou, P. (eds.) Advances in Intelligent Data Analysis XV. pp. 26–37. Springer International Publishing, Cham (2016)
29. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems (2017)

30. Yang, E., Lewis, D.D., Frieder, O., Grossman, D.A., Yurchak, R.: Retrieval and richness when querying by document. In: DESIRES. pp. 68–75 (2018)
31. Zhang, H., Cormack, G.V., Grossman, M.R., Smucker, M.D.: Evaluating sentence-level relevance feedback for high-recall information retrieval. Information Retrieval Journal pp. 1–26 (2020)