Effective Distributed Representations for Academic Expert Search

Mark Berger
University of Amsterdam
Amsterdam, The Netherlands
mark@maberger.nl

Jakub Zavrel
Zeta Alpha Vector
Amsterdam, The Netherlands
zavrel@zeta-alpha.com

Paul Groth
University of Amsterdam
Amsterdam, The Netherlands
p.groth@uva.nl

Abstract

Expert search aims to find and rank experts based on a user's query. In academia, retrieving experts is an efficient way to navigate through a large amount of academic knowledge. Here, we study how different distributed representations of academic papers (i.e. embeddings) impact academic expert retrieval. We use the Microsoft Academic Graph dataset and experiment with different configurations of a document-centric voting model for retrieval. In particular, we explore the impact of the use of contextualized embeddings on search performance. We also present results for paper embeddings that incorporate citation information through retrofitting. Additionally, experiments are conducted using different techniques for assigning author weights based on author order. We observe that using contextual embeddings produced by a transformer model trained for sentence similarity tasks produces the most effective paper representations for document-centric expert retrieval. However, retrofitting the paper embeddings and using elaborate author contribution weighting strategies did not improve retrieval performance.

1 Introduction

To help navigate a large body of academic knowledge, it can be useful to identify expert individuals. Identifying such individuals may be useful to find collaborators (Zhan et al., 2011; Schleyer et al., 2012; Sziklai, 2018), to find paper reviewers (Silva, 2014; Price and Flach, 2017), to find supervisors (Alarfaj et al., 2012a), or to investigate literature in a certain domain. This process of identifying experts given a particular topic is called expert finding (Balog et al., 2009), expertise retrieval (Gonçalves and Dorneles, 2019), or expert search. Expert search systems are information retrieval systems that can automatically rank candidate experts based on their expertise on a certain subject (Husain et al., 2019). In this study, we target the domain of retrieving academic experts based on papers they authored.

Given the central role of papers to defining expertise in this domain, we focus on document-centric expert search systems (Balog et al., 2006). These systems largely rely on statistical language modeling, topic modeling, or term frequency-based approaches to represent documents (Gonçalves and Dorneles, 2019; Husain et al., 2019). Surprisingly, given the rapid advances in the field of contextualized text embeddings (Wang et al., 2020b), little work has been done in applying these approaches to document representation for this task. We hypothesize that considering single words, which is common in the bag-of-words and probabilistic term-based approaches, may significantly reduce the system's "understanding" of the underlying academic documents. To achieve a potentially deeper understanding of these papers, contextualized text embeddings could be used.

Thus, in this paper, we explore the impact of contextualized text embeddings on the performance of the expert search. Specifically, we make the following contributions:

- a comparison of expert search performance using token-based (i.e. BERT (Devlin et al., 2018)) and sentence-based (Sentence-BERT (Reimers and Gurevych, 2019)) contextualized embeddings, non-contextualized embeddings (e.g. GloVe (Pennington et al., 2014)) and classic term frequency representations;

- measurement of the impact on performance when incorporating citation information into contextualized representations through *retrofitting* (Faruqui et al., 2015); and (Zhang, 2019).

- a comparison of two different *strategies* for combining embeddings of the title and abstract of papers.

Additionally, all experiments are conducted using different techniques for assigning author weightings based on author order. Overall, this paper provides evidence for the efficacy of contextualized embeddings for the task of academic expert search. Note that this paper primarily focuses on investigating the performance of various contextualized embeddings and expert ranking aggregation methods within expert retrieval, and not on the entire retrieval process. Therefore, some aspects of neural information retrieval systems such as query understanding, query expansion, or reranking are out of the scope of this study.

Source code for the methods and data processing used in this paper can be found at `https://github.com/mabergerx/SDP500_expert_search`. The processed data used by our methods is available at (Berger, 2020).

The rest of this paper is organized as follows. We begin with a discussion of related work. Afterwards, the data used in this study is described. This is followed by a description of the various embeddings used and our approach to author ranking. Section 6 defines the evaluation and Section 7 details its results. We, then, briefly describe a prototype implementation using these representations. Finally, we discuss the limitations of the work, potential future work and conclude.

2 Related work

In this section, we introduce the primary paradigm for expert search. We then discuss work on voting models, document representations, and the use of text embedding techniques within expert search.

Probabilistic models A driving force behind expertise retrieval research was the launch of the TREC Enterprise Track in 2005 (Craswell et al., 2005). This evaluation campaign led to the emergence of probabilistic models, in particular in the form of language models, as the primary paradigm for expertise retrieval. The core idea behind these approaches is to estimate a language model for each document and then rank the documents by the likelihood of the user query according to the language models (Balog et al., 2009).

Voting models We can see documents authored by experts as evidence for their expertise. A particular type of models, based on *data fusion* methods that aggregate document scores into expert rankings, are called voting models (Husain et al., 2019; Balog et al., 2012).

Given a query, the retrieved documents are assumed to provide evidence about a possible ranking of the authors. This aggregation of the final author list can then be modelled as a voting process, where the document scores are aggregated into author scores (Macdonald, 2009; Macdonald and Ounis, 2008, 2006b,a).

Paper embeddings Document-centric expert search systems rely on the documents to aggregate an expert ranking. However, effectively embedding longer documents is still an open research problem (Beltagy et al., 2020; Zhang et al., 2016; Liu and Lapata, 2017).

Unsupervised document embedding techniques include Sent2Vec (Pagliardini et al., 2018) and Doc2VecC (Chen, 2017), while supervised document embedding techniques include the Universal Sentence Encoder (Cer et al., 2018) and InferSent (Conneau et al., 2018). Recently, the Longformer (Beltagy et al., 2020) was proposed to embed even longer sequences of text than sentences. One research proposed evaluating various sentence encoding techniques in re-ranking of BM25-based research paper recommendations and found that the sentence encoding could be a beneficial method in addition to the BM25 retrieval, but not on its own (Hassan et al., 2019). Adding the BERT [CLS] token embedding into other ranking model's signal has been proposed and is shown to improve the underlying neural ranking architecture (MacAvaney et al., 2019).

As for the embedding of academic papers, most of the research focuses on learning the paper embeddings using linkage information and considers this a graph problem (Wang et al., 2016; Zhang et al., 2019; Mai et al., 2018).

Embedding expertise Given the amount of research on document embedding techniques, there has been surprisingly little attention given to the application of contextualized embedding techniques in the field of expertise retrieval. Three recent surveys and reviews on the field of expertise retrieval (Gonçalves and Dorneles, 2019; Husain et al., 2019; Lin et al., 2017) contained little to no information about the application of embedding techniques.

One of the first works to introduce this concept into expertise retrieval was Author2Vec (J et al., 2016), which uses two models, the *content-info model* and the *link-info model* within the context of the co-authorship network. In the context-info model, the text of the written papers is represented using Paragraph2Vec (Le and Mikolov, 2014).

As briefly mentioned in the introduction, authors that cite each other can be considered having similar interests (Tho et al., 2007; Shibata et al., 2008). Zhang (Zhang, 2019) suggested using retrofitting in the domain of academic papers as a means of introducing this network information into the representation of a paper. Retrofitting is a concept introduced by Faruqui et al. (Faruqui et al., 2015) which proposes the incorporation of the information from semantic lexicons such as WordNet into word embeddings.

3 Data description

The Microsoft Academic Graph (MAG) (Wang et al., 2020a) was used at the primary data source. The data consists of over 200 million papers (titles & abstracts) as well as a variety of metadata. We accessed the November 2018 snapshot of the MAG data through the Open Academic Graph initiative[1], in particular the OAG v2 release.

Due to the very large size of the MAG, we created a custom subset of the data that mainly consisted of Computer Science (CS) related papers. This domain allows us to interpret results better than other science domains.

Our approach in extracting Computer Science (CS) related papers was to take the 113.864 paper titles obtained from arXiv[2] - a widely used preprint server - and do an exhaustive title matching on the full MAG dataset. This search resulted in 29.237 exact title matches, which corresponds to 26,6% of the arXiv data. This set provided us with a substantial initial seed of papers to extract more CS papers from the MAG data.

To allow retrofitting later in the process and create a larger dataset, we expanded this set with the references of all the 29.237 papers, which resulted in a set of 221.347 papers. These references were retrieved by accessing the *references* field of each of the 29.237 paper in the MAG data. Note that these references are not necessarily always complete: some cited articles may not be present in our data due to incompleteness of the source MAG data.

From these 221.347 papers, we then performed bounded stratified sampling for the authors to retrieve a subset of 5.000 authors who are representative of both highly-, medium- and less prolific author populations. The full sampling method is described in Algorithm 5 in the Algorithms appendix.

This set of 5.000 authors served as a starting point for a second, final round of data retrieval. For these authors, we retrieved all their papers and references, resulting in a set of 127.716 papers, which included authors of the referenced papers. For all these new authors, we collected the metadata from the MAG authors dataset and aggregated this information into a single final authors dataset. The reason for expanding the set of authors beyond the 5.000 sampled authors is that a larger pool of papers is beneficial for the retrieval due to the larger search space.

For all titles and abstracts in our dataset, we performed data cleaning. Specifically, (corpus-specific) stopwords were removed, redundant whitespace and Unicode characters were both normalized. URLs and e-mails were removed.

4 Paper embedding methodology

In this section, we describe the paper embedding techniques employed and discuss our approach to embedding indexing and search, as well as retrofitting embeddings.

4.1 Embedding techniques

Various approaches have been used to embed the papers. We divide our approaches into the custom contextual approach and the baseline approaches. In all our baseline approaches, we use the concatenation of the title and the abstract to represent the paper.

Custom contextual approach We use the title and the abstract as representative texts for a paper. Although the title and abstract of a paper are both relevant representations, they may contain information that differs in importance and granularity. In order to capture the possible semantic weight differences between the title and the abstract, we deploy two different embedding combination strategies: the **merge strategy** and the **separate strategy**.

In the merge strategy, we assume that the semantic weights of the title sentence and the abstract sentences are equal. That means that we want to

[1] https://www.openacademic.ai/oag/
[2] https://arxiv.org

take the average over the title- and abstract sentence embeddings without assigning any extra weight to neither. A detailed specification is given in the Algorithm 1 in the appendix.

In the separate strategy, we do want to differentiate between the title and the abstract. In particular, we want to assign more weight to the title than to the individual abstract sentences. We achieve this by first computing the average abstract embedding and then taking the average between that and the title embedding. A detailed specification is given in the Algorithm 3 in the appendix.

We refer to the actual text embedding model as the *embedder*. The *embedder* of choice is Sentence-BERT (Reimers and Gurevych, 2019), which is specifically designed for producing meaningful sentence-level embeddings, suited for Semantic Textual Similarity (STS). Specifically, we make use of the RoBERTa-base model fine-tuned on the combination of NLI datasets, and then further fine-tuned on the STS benchmark training set [3].

Baseline approach: Latent Semantic Indexing (LSI) (Deerwester et al., 1990) TF-IDF vectors with applied singular value decomposition. For our experiments, we chose to set the LSI vector's dimensionality to 768 dimensions, the same as the Sentence-BERT embedding dimensionality.

Baseline approach: BERT- and GloVe pooling To provide a comparison between specifically tuned for document-level representations Sentence-BERT and conventional pooling document embedding techniques, we produced paper embeddings by averaging both BERT and GloVe token embeddings. In both averaging operations, we perform double pooling: first, all tokens within each sentence are embedded and averaged into a single sentence embedding, and then these sentence embeddings are once again averaged into a single paper embedding. The details of this embedding process are shown in Algorithm 2 in the appendix and is identical for both BERT and GloVe embeddings. For both BERT (*bert-base-uncased*) and GloVe embedding calculations, we used the *Flair* (Akbik et al., 2018) Python library.

4.2 Retrofitting

Authors that cite each other can be considered as having similar interests. In the context of having a semantic representation of expertise, it could be helpful to "expand" a paper embedding to broaden the expertise scope of the author beyond a particular paper. To achieve this broadening, we use a technique called retrofitting, which introduces network information into the embeddings.

Inspired by (Zhang, 2019), we adapt the original implementation[4] of retrofitting (Faruqui et al., 2015) to work with academic papers that have been contextually embedded. The retrofitting process is performed for ten iterations. Algorithm 4 in the appendix shows the details about the algorithm.

4.3 Embedding storage and search

We chose the FAISS (Johnson et al., 2017) library by Facebook for our indexing purposes. It is optimized for memory usage and speed and can handle a large number of vectors.

For our embeddings, we chose the *IndexHNSWFlat* index [5]. We use cosine similarity as the measure of similarity between the query embedding $\vec{Q}$ and any of the indexed embeddings $\vec{V}$.

5 Author ranking via voting

From the FAISS index we can, given a query, retrieve top N similar papers. To produce a final author ranking, we adopt a *voting model* based approach.

We can consider the retrieved paper results as the "expertise evidence" for the authors of these papers. A range of different voting approaches based on data fusion techniques has been proposed (Macdonald and Ounis, 2008; Afzal and Maurer, 2011; Alarfaj et al., 2012b) to produce an author ranking given the documents.

Each retrieved document d from the set of retrieved documents $R(Q)$ has an associated similarity score $s(d, Q)$ to it, with regard to the query Q. We can then combine these document scores into aggregated author scores using the *ExpCombSUM (eCS)* data fusion function (Macdonald, 2009):

$$eCS(C, Q) = \sum_{d \in R(Q) \cap D_C} e^{(s(d,Q)))} \quad (1)$$

where C is a candidate expert, D_C is the set of documents associated with candidate C.

[3] https://github.com/UKPLab/sentence-transformers

[4] https://github.com/mfaruqui/retrofitting

[5] https://github.com/facebookresearch/faiss/wiki/Faiss-indexes

This algorithm (Macdonald and Ounis, 2008; Macdonald, 2009), assumes that each document produces a static score per related author. In the case of academic papers, that is not the case, as most papers have multiple authors. These authors mostly have a different level of involvement in a particular paper and, therefore, may have a different vote produced by the document, depending on their authorship role. Recent research has shown that because the research is increasingly more interdisciplinary, evaluating authors based on their rank within the order of authors is becoming increasingly difficult (Júnior et al., 2017). Therefore, it could be valuable to assign different weights to different authors of the same document. To the best of our knowledge, no previous work has been done on exactly defining weights on the authorship scores within a voting model. We define four different weighting strategies:

1. **Binary weighting**. Each author gets the full score for a document. This strategy assumes that each author contributed equally.

2. **Uniform weighting**. Each author gets $\frac{\textit{fullScore}}{\textit{\# authors}}$ for a document. This strategy assumes that each author contributed equally but does normalize the score by the number of authors.

3. **Descending weighting**. The first author gets the full document score. Each following author gets $\textit{fullScore} * decayFactor$, where $decayFactor$ starts at 0.8 and decreases with 0.2 for each consecutive author. This strategy assumes that the authors are listed in descending involvement order.

4. **Parabolic weighting**. The first and last author get the full document score. All authors in between follow the **descending weighting**. This strategy assumes that the first author is similar to the descending weighting, but also takes the possible importance of the last author as the project supervisor.

Using these data fusion approaches, a fairness problem may occur: highly prolific authors, that may be associated with many documents (for instance, because they are the head of a lab) may receive an unfairly large number of votes, which does not necessarily indicate their expertise. Candidate length normalization is proposed to deal with this unfairness, just as document length normalization is often performed in document retrieval systems (Macdonald, 2009).

The use of a classical document normalization technique based on the *Divergence From Randomness framework* (Amati, 2003) is proposed (Macdonald, 2009), and has the following formula:

$$s_N(C,Q) = s(C,Q) \cdot log_2(1 + \alpha \cdot \frac{aL}{lP}) \quad (2)$$

where α is a hyperparameter controlling the amount of normalization, aL is the average amount of publications, and lP is the length of the profile of the candidate C. The lower the α parameter is, the more less prolific authors are boosted, and the more highly prolific authors are suppressed.

In practice, we discovered that because of the nature of our dataset, if we apply the above normalization technique, many authors from the long tail are retrieved, even when we use high α values. Therefore, we experimented with introducing another term to the equation: β, which serves as a profile length "booster":

$$s_N(C,Q) = s(C,Q) \cdot log_2(1 + \alpha \cdot \frac{aL}{lP + \beta}) \quad (3)$$

While it does introduce bias and eases the normalization, in our case, it provided an extra parameter to tune and resulted in a better mix of well- and lesser-known authors.

6 Evaluation methods

To evaluate the different retrieval strategies, we developed a method which uses the *field of work* tags present in the MAG dataset for the authors as a proxy for evaluating the relevance of an author. Because we use a document-centric retrieval strategy which uses strictly only the embeddings of the paper's title and abstract, the *field of work* tags are not used in the retrieval process, allowing us to use these tags in the evaluation process.

The query test set for our research was selected from the full distribution of author tags. The final query test set contained a hundred Computer Science related queries. The set is available in Table 3 in the appendix.

6.1 Relevance metrics

Before we can use any of the existing binary information retrieval metrics, we need to define a notion of *relevance* of an author given a query.

Based on the *field of work* tags, we define two relevance checks:

1. **Exact topic query evaluation (Brochier et al., 2018)**. This approach takes a description of a topic and uses it directly as a query. The experts associated with that topic are then the ground truth list of candidates to be retrieved. In our case, the *field of work* tags of the authors are used as queries, and the retrieved authors are labelled relevant if they have that tag.

2. **Approximate topic query evaluation**. Sometimes, a query retrieves authors who have an incomplete *field of work* tag list or have tags which are very similar to the query but do not exactly match it. For instance, author A may have *"automatic summarization"* in their tags list, while the query was *"automatic text summarization"*. This author is clearly highly relevant to the query but would be labelled as irrelevant by the exact topic query evaluation method. Therefore, we introduce a fuzzy relevance checking method which, given a query, calculates the cosine similarity between the query embedding and each of the author's tags. If any of the similarities are higher than a chosen threshold, then we deem the author relevant.

Once we can label each retrieved author as relevant or not relevant, we can use different evaluation metrics. We evaluate our system by using three binary relevance metrics, all measured @N and using both the exact and approximate topic query evaluation: **Mean Reciprocal Rank** ($MRR@N_{exact}$, $MRR@N_{approx}$), **Mean Precision @ N** ($MP@N_{exact}$, $MP@N_{approx}$), and **Mean Average Precision** ($MAP@N_{exact}$, $MAP@N_{approx}$)

Some authors are more relevant than others. To incorporate this into our evaluation, we also use the **Normalized discounted cumulative gain** (nDCG@N) score, which is sensitive to the position of the relevant items in the produced ranking.

To produce the nDCG scores, we first need to have the score for the ideal ranking given a query, IDCG. To calculate the IDCG scores, for each query in our test set, we created a mapping between the query and the corresponding top authors. Each author in such mapping got a relevance label in relation to the query. In our implementation of the author's relevance, we use the citations of the relevant papers of an author as a proxy for the expertise. Although any expertise measure of an author is not fully objective, and many factors seem to have effect on the citation activity (Yan et al., 2011), we chose the citation counts of the papers as a proxy for expertise for the following reasons:

- This measure is explainable.
- It prevents highly prolific but rarely cited authors to be labeled as highly relevant for multiple topics just based on their output

Given this final query-to-expert mapping, we precalculated the IDCG@10 score for each of the test queries in our dataset, so we could later calculate the nDCG@10 score per query. Once we calculated all the scores for our test set, we can take the average of those scores to have a single nDCG@10 score for our current system.

7 Results

In this section, first the overall quantitative evaluation results are discussed and then we zoom in on the performance of retrofitting and author contribution weighting.

7.1 Voting model results

The results of the voting model approach are presented in Table 1. Here we only present the exact topic query evaluation results, as we observe that approximate topic query evaluation results correspond to the exact results but are overall higher. In particular, for MRR, the approximate results are 0.06 higher on average; for MAP, the approximate results are 0.16 higher; for MP@10 the approximate results are 0.15 higher; and finally the MP@5 approximate results are, again, 0.15 higher. The full results table with the approximate query evaluation results included, are presented in Table 2 in the Appendix.

We can observe that the LSI baseline produces strong results, outperforming both embedding pooling baselines. From the four used author contribution weighting schemes, the binary score weighting is the best performing weighting. However, the overall performance difference between the weightings is quite small.

PAPER EMBEDDING KIND	DATA FUSION TECHNIQUE	MRR@10 EXACT	MAP@10 EXACT	MP@10 EXACT	MP@5 EXACT	NDCG @10	NDCG @5
LSI	$expCombSUM_{uniform}$	0.75	0.399	0.462	0.496	0.39	0.42
	$expCombSUM_{binary}$	0.753	0.428	0.491	0.512	0.41	0.44
	$expCombSUM_{descending}$	0.755	0.411	0.476	0.484	0.39	0.42
	$expCombSUM_{parabolic}$	0.763	0.392	0.457	0.482	0.39	0.42
Average pooled BERT	$expCombSUM_{uniform}$	0.554	0.117	0.198	0.206	0.1	0.11
	$expCombSUM_{binary}$	0.558	0.12	0.203	0.222	0.12	0.13
	$expCombSUM_{descending}$	0.556	0.117	0.2	0.208	0.11	0.12
	$expCombSUM_{parabolic}$	0.56	0.107	0.18	0.218	0.1	0.12
Average pooled GloVe	$expCombSUM_{uniform}$	0.626	0.272	0.369	0.398	0.27	0.29
	$expCombSUM_{binary}$	0.672	0.304	0.402	0.414	0.3	0.31
	$expCombSUM_{descending}$	0.661	0.286	0.383	0.398	0.28	0.3
	$expCombSUM_{parabolic}$	0.676	0.254	0.33	0.392	0.25	0.28
Merged Sentence-BERT	$expCombSUM_{uniform}$	0.834	0.419	0.491	0.528	0.43	0.47
	$expCombSUM_{binary}$	0.83	0.437	0.509	0.546	0.42	0.46
	$expCombSUM_{descending}$	0.818	0.419	0.493	0.526	0.41	0.46
	$expCombSUM_{parabolic}$	0.812	0.4	0.484	0.508	0.41	0.45
Separate Sentence-BERT	$expCombSUM_{uniform}$	0.838	0.495	0.572	0.616	0.53	0.58
	$\mathbf{expCombSUM_{binary}}$	0.837	0.518	**0.59**	**0.626**	**0.54**	**0.6**
	$Norm(expCombSUM_{binary})$ $\beta = 0$ and $\alpha = 1$	0.619	0.174	0.282	0.272	0.15	0.14
	$Norm(expCombSUM_{binary})$ $\beta = 0$ and $\alpha = 1000$	0.694	0.218	0.318	0.324	0.16	0.17
	$Norm(expCombSUM_{binary})$ $\beta = 10$ and $\alpha = 1000$	0.777	0.293	0.381	0.406	0.22	0.25
	$Norm(expCombSUM_{binary})$ $\beta = 50$ and $\alpha = 1000$	0.769	0.362	0.455	0.466	0.31	0.33
	$Norm(expCombSUM_{binary})$ $\beta = 1000$ and $\alpha = 1000$	0.813	0.404	0.491	0.52	0.37	0.4
	$expCombSUM_{descending}$	0.839	0.501	0.581	0.612	0.52	0.58
	$expCombSUM_{parabolic}$	0.819	0.486	0.565	0.592	0.52	0.56
Retrofitted merged Sentence-BERT	$expCombSUM_{uniform}$	0.792	0.384	0.45	0.482	0.38	0.42
	$expCombSUM_{binary}$	0.83	0.404	0.467	0.496	0.39	0.44
	$expCombSUM_{descending}$	0.813	0.39	0.454	0.486	0.38	0.42
	$expCombSUM_{parabolic}$	0.775	0.38	0.445	0.474	0.38	0.4
Retrofitted separate Sentence-BERT	$expCombSUM_{uniform}$	0.821	0.51	0.577	0.606	0.5	0.54
	$expCombSUM_{binary}$	**0.841**	**0.519**	0.584	0.616	0.51	0.54
	$expCombSUM_{descending}$	0.831	0.505	0.569	0.61	0.49	0.54
	$expCombSUM_{parabolic}$	0.808	0.509	0.583	0.596	0.5	0.53

Table 1: Results for the voting model author retrieval strategy. The best results are formatted in bold.

The best performing configuration is the separate embedding strategy with the binary distributed paper scores. We see that normalizing the *expCombSUM* function with a low α and $\beta = 0$ leads to steep decrease in performance. With higher α and β, the performance goes up but can be explained by "cancelling out" the normalization effect. One of the reasons for this bad performance could be that the dataset contains many authors from the long tail; some of the authors naturally may have worse metadata resulting in missing expertise tags. Moreover, for lesser-known authors in the MAG, we have encountered a problem that the profiles get deleted or get different author ids, which also corrupts the metadata and the results.

Retrofitting the embeddings did not improve the results, except for two evaluation metrics: $MRR@10_{exact}$ and $MAP@10_{exact}$ in case of the retrofitted separate embeddings.

7.2 Performance of retrofitting

Retrofitting the embeddings did not improve our retrieval performance with the exception of two metrics. One of the explanations could be that the relatively small size of our dataset can hurt the retrofitting process. Combined with the variance in the number of neighbours per paper, some of the resulting retrofitted embeddings might be driven away too much from the original embedding, while other embeddings are not modified "enough".

7.3 Effect of different author contribution weightings

Throughout the experiment, we observed that the binary author contribution weighting performs the best. Therefore, we can conclude that, for our voting model configuration, introducing elaborate author contribution combinations does not improve the retrieval performance.

8 Prototype implementation

We implemented our model int a prototype, which consists of a REST API, where the users can, given a search query, look for N most relevant experts. Each individual expert representation is contained within a JSON object which contains not only the author's name and MAG id, but also the authors affiliation information (retrieved using GRID [6]), the list of papers which voted for the author and their corresponding document scores in relation to the query, and additional author information from WikiData (Vrandečić and Krötzsch, 2014).

9 Discussion & Future work

This study faced multiple challenges regarding data quality, data freshness, embedding strategy considerations, and the retrieval base on embeddings. In this section, we discuss these issues.

Data completeness and variability. The *field-of-work* tags used in the evaluation are not always complete and are subject to the specific format used by the MAG. In addition, for some authors, no tags are present in our snapshot of the MAG data. This has an effect on the evaluation performance by introducing both false positives and -negatives into the relevancy determining process. Overall, our system would profit significantly from a larger pool of papers and authors.

Shifting expertise. Many authors have varying interests during their academic career. Our system is inherently a snapshot of their academic activity: we only search and aggregate within a bounded set of papers. Introducing temporal aspect into the author search, which would intelligently account for shifting expertise could improve the retrieval results.

Representing expertise domains. Many authors are not experts in just one niche field, but rather are knowledgeable about a pretty broad field of science, with more in-depth knowledge about a few specific sub-fields. Clustering within the author expertise to find expertise sub-clusters might help to create more nuanced author representations by taking the cluster centroids as the individual author embeddings. This approach, while interesting, requires more data, as clustering within the papers of one author requires having a significant amount of papers per author.

Performance of individual retrieval strategies. Different types of embeddings may perform well in different scenarios. For example, retrofitting of the paper embeddings leads to the "widening" of the semantic representation of a paper into the direction of its neighbours. For example, retrofitted embeddings may perform better for more broad/general queries and worse on more specific queries. The same applies for normalization in the voting model: retrieving less prolific authors may be beneficial for some user's needs, while it technically hurts the quantitative performance of the system.

Document pooling strategies. In the two paper embedding strategies, we perform pooling over multiple sentence embeddings. While these strategies seem to work well to represent the papers for our task, it would still be interesting to use new, state-of-the-art approaches for embedding longer texts, such as the Longformer (Beltagy et al., 2020), to avoid using any pooling strategies and loosing semantic value. Techniques like the recently introduced SPECTER (Cohan et al., 2020) could also be used to produce better citation-informed document embeddings. Finally, we could employ the SciBERT (Beltagy et al., 2019) model in our baseline pooling strategies or even fine-tune a SciBERT model on longer sequences, similarly to Sentence-BERT, as it is better suited for academic texts.

Graph embeddings. In our approach, we used retrofitting to introduce citation network information into our initial paper embeddings. However, we could also go a step further and use graph embedding techniques to create native graph-based embeddings for our papers (Mai et al., 2018; Wang et al., 2016; Zhang et al., 2019).

10 Conclusion

In this study, we investigated different approaches for embedding academic papers and using the embeddings in an expert search task.

Overall, we found that Transformer-based contextual text embeddings work well on the domain of academic papers. By using the Sentence-BERT model trained on NLI and SNS tasks, we outperformed the strong LSI baseline often employed in information retrieval systems on all ten evaluation metrics. We also outperformed the baseline strategies of average pooled BERT and GloVe embeddings.

Employing a weighted embedding combination strategy to represent a paper can however be valu-

[6] https://www.grid.ac/

able, as we found that using a separate embedding combination strategy outperformed the "default" merged strategy on nine out of ten metrics.

We hypothesized that enriching the paper embeddings with citation information, in a process called retrofitting, could improve improve retrieval performance. Our experiments did not confirm this hypothesis, as non-retrofitted embeddings performed better in our task on all but two evaluation metrics.

Finally, we employed various data fusion techniques to convert the top *N* retrieved papers given a query into a ranking of authors. A voting model was used, where each document served as evidence for the corresponding author's expertise. We investigated whether using author contribution weighting strategies within the voting process would improve expertise retrieval. We observed no performance gain over the "default" binary strategy.

Given the direction of this study, we think that the most suitable application areas for the methodology we proposed are reviewer finding, supervisor finding and investigating literature on a topic. The reasoning behind this is that finding a collaborator may require and involve more sophisticated information about the institution, availability and current field of interest of the found experts. The proposed three areas, however, allow for less specificity and can better benefit from the improved retrieval.

While research in the field of expertise retrieval is not as active in the second half of the 2010s as it was in the first half, the area of text representations and retrieval has seen dramatic improvements. This study was an effort to apply these new techniques into the field of expertise retrieval and has shown that substantial improvements can be made over the existing retrieval algorithms. We hope that this study can contribute to a new research wave within the field of (academic) expertise retrieval.

Acknowledgements

We would like to thank the reviewers for their insightful commentary. Additionally, we would like to thank dr. Wouter Weerkamp from Zeta Alpha for his valuable feedback and assessment of the work.

References

Muhammad Tanvir Afzal and Hermann A Maurer. 2011. Expertise recommender system for scientific community. *J. UCS*, 17(11):1529–1549.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Fawaz Alarfaj, Udo Kruschwitz, David Hunter, and Chris Fox. 2012a. Finding the right supervisor: Expert-finding in a university domain. In *Proceedings of the NAACL HLT 2012 Student Research Workshop*, pages 1–6, Montréal, Canada. Association for Computational Linguistics.

Fawaz Alarfaj, Udo Kruschwitz, David Hunter, and Chris Fox. 2012b. Finding the right supervisor: Expert-finding in a university domain. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*, NAACL HLT '12, page 1–6, USA. Association for Computational Linguistics.

G Amati. 2003. *Probabilistic Models for Information Retrieval based on Divergence from Randomness. University of Glasgow, UK.* Ph.D. thesis, PhD Thesis.

Krisztian Balog, Leif Azzopardi, and Maarten De Rijke. 2006. Formal models for expert finding in enterprise corpora. *Proceedings of the Twenty-Ninth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006:43–50.

Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. 2009. A language modeling framework for expert finding. *Information Processing and Management*, 45(1):1–19.

Krisztian Balog, Yi Fang, Maarten De Rijke, Pavel Serdyukov, and Luo Si. 2012. Expertise retrieval. *Foundations and Trends in Information Retrieval*, 6(2-3):127–256.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: Pretrained language model for scientific text. In *EMNLP*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Mark Berger. 2020. Datasets for effective distributed representations for academic expert search.

Robin Brochier, Adrien Guille, Benjamin Rothan, and Julien Velcin. 2018. Impact of the query set on the evaluation of expert finding systems. *CEUR Workshop Proceedings*, 2132:32–45.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil Google Research Mountain View. 2018. Universal Sentence Encoder.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. *CoRR*, abs/1707.02377.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. Specter: Document-level representation learning using citation-informed transformers. pages 2270–2282.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2018. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data.

Nick Craswell, Arjen P. de Vries, and Ian Soboroff. 2005. Overview of the trec 2005 enterprise track. In *TREC*.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. *NAACL HLT 2015 - 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, (i):1606–1615.

Rodrigo Gonçalves and Carina Friedrich Dorneles. 2019. Automated expertise retrieval: A taxonomy-based survey and open issues. *ACM Comput. Surv.*, 52(5).

Hebatallah A Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, Alessandro Micarelli, and Joeran Beel. 2019. BERT, ELMo, USE and InferSent Sentence Encoders: The Panacea for Research-Paper Recommendation?

Omayma Husain, Naomie Salim, Rose Alinda Alias, Samah Abdelsalam, and Alzubair Hassan. 2019. Expert finding systems: A systematic review. *Applied Sciences (Switzerland)*, 9(20):1–32.

Ganesh J, Soumyajit Ganguly, Manish Gupta, Vasudeva Varma, and Vikram Pudi. 2016. Author2vec: Learning author representations by combining content and link information. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, page 49–50, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734.

E. A. C. Júnior, F. N. Silva, L. Costa, and Diego R. Amancio. 2017. Patterns of authors contribution in scientific manuscripts. *ArXiv*, abs/1609.05545.

Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents.

Shuyi Lin, Wenxing Hong, Dingding Wang, and Tao Li. 2017. A survey on expert finding techniques. *Journal of Intelligent Information Systems*, 49(2):255–279.

Yang Liu and Mirella Lapata. 2017. Learning structured text representations. *CoRR*, abs/1705.09207.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. Cedr: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 1101–1104, New York, NY, USA. Association for Computing Machinery.

Craig Macdonald. 2009. *The voting model for people search*. Ph.D. thesis, University of Glasgow.

Craig Macdonald and Iadh Ounis. 2006a. A Belief Network Model for Expert Search. *Computing*.

Craig Macdonald and Iadh Ounis. 2006b. Voting for candidates. In *Proceedings of the 15th ACM international conference on Information and knowledge management - CIKM '06*, page 387, New York, New York, USA. ACM Press.

Craig Macdonald and Iadh Ounis. 2008. Voting techniques for expert search. *Knowl Inf Syst*, 16:259–280.

Gengchen Mai, Krzysztof Janowicz, and Bo Yan. 2018. Combining text embedding and knowledge graph embedding techniques for academic search engines. In *Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data (NLIWOD-4) and 9th Question Answering over Linked Data challenge (QALD-9) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, California, United States of America, October 8th - 9th, 2018*, volume 2241 of *CEUR Workshop Proceedings*, pages 77–88. CEUR-WS.org.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Simon Price and Peter A Flach. 2017. Computational support for academic peer review: A perspective from artificial intelligence. *Communications of the ACM*, 60(3):70–79.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Titus Schleyer, Brian S. Butler, Mei Song, and Heiko Spallek. 2012. Conceptualizing and advancing research networking systems. *ACM Trans. Comput.-Hum. Interact.*, 19(1).

Naoki Shibata, Yuya Kajikawa, Yoshiyuki Takeda, and Katsumori Matsushima. 2008. Detecting emerging research fronts based on topological measures in citation networks of scientific publications. *Technovation*, 28(11):758 – 775.

Attulugamage Thushari Priyangika Silva. 2014. *A research analytics framework for expert recommendation in research social networks*. Ph.D. thesis, City University of Hong Kong.

Balázs Sziklai. 2018. How to identify experts in a community? *International Journal of Game Theory*, 47:155–173.

Quan Thanh Tho, S.C. Hui, and A.C.M. Fong. 2007. A citation-based document retrieval system for finding research expertise. *Information Processing & Management*, 43(1):248 – 264.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020a. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413.

Suhang Wang, Jiliang Tang, Charu Aggarwal, and Huan Liu. 2016. Linked document embedding for classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 115–124, New York, NY, USA. Association for Computing Machinery.

Yuxuan Wang, Yutai Hou, Wanxiang Che, and Ting Liu. 2020b. From static to dynamic word representations: a survey. *International Journal of Machine Learning and Cybernetics*.

Rui Yan, Jie Tang, Xiaobing Liu, Dongdong Shan, and Xiaoming Li. 2011. Citation count prediction: Learning to estimate future citations for literature. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, page 1247–1252, New York, NY, USA. Association for Computing Machinery.

Zhenjiang Zhan, Lichun Yang, Shenghua Bao, Dingyi Han, Zhong Su, and Yong Yu. 2011. Finding appropriate experts for collaboration. In *Proceedings of the 12th International Conference on Web-Age Information Management*, WAIM'11, page 327–339, Berlin, Heidelberg. Springer-Verlag.

Ye Zhang, Md Mustafizur Rahman, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, and Matthew Lease. 2016. Neural Information Retrieval: A Literature Review. (November).

Yi Zhang. 2019. *Learning Embeddings for Academic Papers*. Ph.D. thesis.

Yi Zhang, Fen Zhao, and Jianguo Lu. 2019. P2V: large-scale academic paper embedding. *Scientometrics*, 121(1):399–432.

Algorithm 1: Paper embedding creation following the merge strategy.

```
Result: Paper embeddings $ME$ following the merge strategy
Input: A set of paper titles $T$, a set of batches of abstract sentences $A$ (with $|T| = |A|$) and an
       embedder model $Embedder$

mergedEmbeddings ← []
abstractEmbeddingBatches ← []
titleEmbeddings ← $Embedder.embed(T)$
for aBatch ∈ A do
    batchEmbeddings ← $Embedder.embed(aBatch)$
    abstractEmbeddingBatches.append(batchEmbeddings)
end
assert $|titleEmbeddings| = |abstractEmbeddingBatches|$
for tE ∈ titleEmbeddings,   aEB ∈ abstractEmbeddingBatches do
    aEB.append(tE)
    N ← $dim(aEB)$
    mergedEmbedding ← $\frac{1}{N}\sum_{i=1}^{N}(emb_i \in aEB)$  ▷ Take the element-wise average of all
     the embeddings, resulting in one embedding
    mergedEmbeddings.append(mergedEmbedding)
end
return mergedEmbeddings
```

Algorithm 2: Paper embedding creation following baseline BERT or GloVe strategy.

```
Result: Baseline paper embeddings $ME$ created by either BERT or GloVe embedding model.
Input: A set of batches of abstract sentences $A$ with elementwise appended corresponding paper
       titles $T$ (with $|T| = |A|$) and an embedder model $Embedder$ (either BERT or GloVe)

embeddings ← []
abstractEmbeddingBatches ← []
for aBatch ∈ A do
    batchEmbeddings ← $Embedder.embed(aBatch)$
    abstractEmbeddingBatches.append(batchEmbeddings)
end
for aEB ∈ abstractEmbeddingBatches do
    N ← $dim(aEB)$
    pooledEmbedding ← $\frac{1}{N}\sum_{i=1}^{N}(emb_i \in aEB)$  ▷ Take the element-wise average of all
     the embeddings, resulting in one embedding
    embeddings.append(pooledEmbedding)
end
return embeddings
```

Algorithm 3: Paper embedding creation following the separate strategy.

```
Result: Paper embeddings $ME$ following the separate strategy
Input: A set of paper titles $T$, a set of batches of abstract sentences $A$ (with $|T| = |A|$) and an
       embedder model $Embedder$

mergedEmbeddings ← []
abstractAverageEmbeddings ← []
titleEmbeddings ← $Embedder.embed(T)$
for aBatch ∈ A do
    batchEmbeddings ← $Embedder.embed(aBatch)$
    N ← $dim(batchEmbeddings)$
    averageBatchEmbedding ← $\frac{1}{N} \sum_{i=1}^{N} (emb_i \in batchEmbeddings)$   ▷ Create an average
      embedding for abstract sentences only.
    abstractAverageEmbeddings.append(averageBatchEmbedding)
end
assert $|titleEmbeddings| = |abstractAverageEmbeddings|$
for tE ∈ titleEmbeddings,  aE ∈ abstractAverageEmbeddings do
    separateArray ← $[tE, aE]$   ▷ Create a two item array consisting of the title
      embedding and the average abstract embedding.
    N ← $dim(separateArray)$
    separateEmbedding ← $\frac{1}{N} \sum_{i=1}^{N} (emb_i \in separateArray)$  ▷ Create an average embedding
      over just two embeddings.
    separateEmbeddings.append(separateEmbedding)
end
return separateEmbeddings
```

Algorithm 4: The retrofitting algorithm.

Result: Retrofitted paper embeddings

Data: A mapping between paper id's and their embeddings $\rightarrow$ ***C*** (corpus), the mapping between paper id's in lexicon and their references $\rightarrow$ ***L*** (lexicon), and the amount of algorithm iterations $\rightarrow$ ***numIter***.

```
Retrofit $(C, L, numIter)$
    $newCorpus \leftarrow$ deepcopy $(C)$   ▷ Deep copy the corpus into a new variable to alter
    $I \leftarrow newCorpus$.keys()   ▷ Extract all the keys (paper id's) from the corpus
    $corpusVocabulary \leftarrow$ set $(I)$   ▷ Consider only the unique paper id's
    // Two lines below exist for the case where the paper id's in lexicon differ from paper id's in corpus, normally not the case in our environment. Otherwise, they have not effect.
    $LI \leftarrow L$.keys()   ▷ Extract all the keys (paper id's) from the lexicon
    $relevantVocabulary \leftarrow corpusVocabulary \cap$ set $(LI)$   ▷ Consider only the overlapping id's from lexicon and corpus
    for $it \leftarrow 0$ to $numIter$ do
        foreach $paper \in relevantVocabulary$ do
            $paperNeighbours \leftarrow$ set $(L[word]) \cap corpusVocabulary$   ▷ Extract the set of neighbours of the current paper that actually have an embedding in our corpus
            $numNeighbours \leftarrow$ len $(paperNeighbours)$
            if $numNeighbours = 0$ then
                continue   ▷ If the paper has no neighbours, do not adjust the embedding and go to next paper
            end
            $newEmbedding \leftarrow numNeighbours * C[paper]$   ▷ Initialize the new embedding by weighing in the original embedding
            foreach $neighbour$ in $paperNeighbours$ do
                $newEmbedding += newCorpus[neighbour]$   ▷ Add the neighbour embedding with weight 1.
            end
            $newCorpus[paper] \leftarrow \frac{newEmbedding}{2*numNeighbours}$   ▷ Finalize the new embedding by dividing the current new embedding by (2 * number_neighbours), essentially putting the embedding back in the same space.
        end
    end
    return $newCorpus$
```

Algorithm 5: The author sampling strategy from the arXiv subset.

Result: A stratified sample of authors $randomAuthorSample$ of size $s = 5000$.

Input: A set of authors $authorsArxiv$ extracted from the the arXiv subset with the corresponding metadata and the final sample size $s = 5000$ (arbitrarily chosen).

1. Filter $authorsArxiv$ to only contain authors which, for their papers, have references in the data. That is needed to be able to do retrofitting later.
2. Given this new proper set of authors, $properA$, initialize four bins (strata) based on the amount of author publications:
   (a) 5 - 10 publications
   (b) 10 - 50 publications
   (c) 50 - 100 publications
   (d) 100+ publications
3. Calculate the bin size for each bin and also the total amount of authors in all the bins.
4. Perform proportionate allocation using a sampling fraction in each of the strata that is proportional to that of the total population. We went for a sample size of 5000 authors. For a single bin, the allocation process is as following:

   Bin 1 (5 - 10 publications) has 22.943 authors. The total population of authors in all the bins is 35.450 authors ($|properA|$). The share of Bin1 in the final 5.000 authors set is then

   Bin 1 => 22.943 * (5.000 / 35.450) = 3.235 authors
5. Proceed doing a simple random sampling from those pools of authors, resulting in *randomAuthorSample*.

return *randomAuthorSample*

PAPER EMBEDDING KIND	**DATA FUSION TECHNIQUE**	**MRR@10 EXACT**	**MRR@10 APPRX**	**MAP@10 EXACT**	**MAP@10 APPRX**	**MP@10 EXACT**	**MP@10 APPRX**	**MP@5 EXACT**	**MP@5 APPRX**	**NDCG@10**	**NDCG@5**
LSI	$expCombSUM_{uniform}$	0.75	0.864	0.399	0.603	0.462	0.662	0.496	0.696	0.39	0.42
	$expCombSUM_{binary}$	0.753	0.844	0.428	0.626	0.491	0.693	0.512	0.708	0.41	0.44
	$expCombSUM_{descending}$	0.755	0.857	0.411	0.603	0.476	0.668	0.484	0.684	0.39	0.42
	$expCombSUM_{parabolic}$	0.763	0.864	0.392	0.587	0.457	0.651	0.482	0.68	0.39	0.42
Average pooled BERT	$expCombSUM_{uniform}$	0.554	-	0.117	-	0.198	-	0.206	-	0.1	0.11
	$expCombSUM_{binary}$	0.558	-	0.12	-	0.203	-	0.222	-	0.12	0.13
	$expCombSUM_{descending}$	0.556	-	0.117	-	0.2	-	0.208	-	0.11	0.12
	$expCombSUM_{parabolic}$	0.56	-	0.107	-	0.18	-	0.218	-	0.1	0.12
Average pooled GloVe	$expCombSUM_{uniform}$	0.626	-	0.272	-	0.369	-	0.398	-	0.27	0.29
	$expCombSUM_{binary}$	0.672	-	0.304	-	0.402	-	0.414	-	0.3	0.31
	$expCombSUM_{descending}$	0.661	-	0.286	-	0.383	-	0.398	-	0.28	0.3
	$expCombSUM_{parabolic}$	0.676	-	0.254	-	0.33	-	0.392	-	0.25	0.28
Merged Sentence-BERT	$expCombSUM_{uniform}$	0.834	**0.903**	0.419	0.586	0.491	0.658	0.528	0.684	0.43	0.47
	$expCombSUM_{binary}$	0.83	0.893	0.437	0.606	0.509	0.677	0.546	0.708	0.42	0.46
	$expCombSUM_{descending}$	0.818	**0.903**	0.419	0.59	0.493	0.657	0.526	0.688	0.41	0.46
	$expCombSUM_{parabolic}$	0.812	0.869	0.4	0.555	0.484	0.634	0.508	0.666	0.41	0.45
Separate Sentence-BERT	$expCombSUM_{uniform}$	0.838	0.875	0.495	0.673	0.572	0.744	0.616	0.764	0.53	0.58
	$\mathbf{expCombSUM_{binary}}$	0.837	0.892	0.518	**0.69**	**0.59**	**0.751**	**0.626**	**0.784**	**0.54**	**0.6**
	$Norm(expCombSUM_{binary})$ $\beta = 0$ and $\alpha = 1$	0.619	0.678	0.174	0.284	0.282	0.42	0.272	0.398	0.15	0.14
	$Norm(expCombSUM_{binary})$ $\beta = 0$ and $\alpha = 1000$	0.694	0.737	0.218	0.331	0.318	0.452	0.324	0.456	0.16	0.17
	$Norm(expCombSUM_{binary})$ $\beta = 10$ and $\alpha = 1000$	0.777	0.807	0.293	0.409	0.381	0.509	0.406	0.542	0.22	0.25
	$Norm(expCombSUM_{binary})$ $\beta = 50$ and $\alpha = 1000$	0.769	0.81	0.362	0.49	0.455	0.589	0.466	0.602	0.31	0.33
	$Norm(expCombSUM_{binary})$ $\beta = 1000$ and $\alpha = 1000$	0.813	0.853	0.404	0.548	0.491	0.638	0.52	0.658	0.37	0.4
	$expCombSUM_{descending}$	0.839	0.894	0.501	0.666	0.581	0.737	0.612	0.758	0.52	0.58
	$expCombSUM_{parabolic}$	0.819	0.878	0.486	0.654	0.565	0.729	0.592	0.742	0.52	0.56
Retrofitted merged Sentence-BERT	$expCombSUM_{uniform}$	0.792	0.839	0.384	0.551	0.45	0.61	0.482	0.638	0.38	0.42
	$expCombSUM_{binary}$	0.83	0.865	0.404	0.563	0.467	0.618	0.496	0.652	0.39	0.44
	$expCombSUM_{descending}$	0.813	0.843	0.39	0.551	0.454	0.609	0.486	0.644	0.38	0.42
	$expCombSUM_{parabolic}$	0.775	0.847	0.38	0.542	0.445	0.598	0.474	0.638	0.38	0.4
Retrofitted separate Sentence-BERT	$expCombSUM_{uniform}$	0.821	0.893	0.51	0.658	0.577	0.716	0.606	0.732	0.5	0.54
	$expCombSUM_{binary}$	**0.841**	0.893	**0.519**	0.661	0.584	0.718	0.616	0.75	0.51	0.54
	$expCombSUM_{descending}$	0.831	0.895	0.505	0.647	0.569	0.702	0.61	0.734	0.49	0.54
	$expCombSUM_{parabolic}$	0.808	0.863	0.509	0.658	0.583	0.724	0.596	0.732	0.5	0.53

Table 2: Results for the voting model author retrieval strategy. The best results are formatted in bold.

Queries			
'cluster analysis'	'Bayesian statistics'	'world wide web'	'Novelty detection'
'Image segmentation'	'kernel density estimation'	'gibbs sampling'	'semantic grid'
'Parallel algorithm'	'learning to rank'	'user interface'	'Knowledge extraction'
'Monte Carlo method'	'relational database'	'belief propagation'	'Computational biology'
'Convex optimization'	'activity recognition'	'interpolation'	'Web 2.0'
'Dimensionality reduction'	'wearable computer'	'wavelet transform'	'Network theory'
'Facial recognition system'	'ensemble learning'	'transfer of learning'	'Video denoising'
'k-nearest neighbors algorithm'	'wordnet'	'topic model'	'Quantum information science'
'Hierarchical clustering'	'medical imaging'	'clustering high-dimensional data'	'Color quantization'
'Automatic text summarization'	'deconvolution'	'game theory'	'social web'
'Dynamic programming'	'Latent Dirichlet allocation'	'biometrics'	'entity linking'
'Genetic algorithm'	'Euclidian distance'	'constraint satisfaction'	'information privacy'
'Human-computer interaction'	'web service'	'combinatorial optimization'	'random forest'
'Categorial grammar'	'multi-task learning'	'speech processing'	'cloud computing'
'Semantic Web'	'Linear separability'	'multi-agent system'	'Knapsack problem'
'fuzzy logic'	'OWL-S'	'mean field theory'	'Linear algebra'
'image restoration'	'Wireless sensor network'	'social network'	'batch processing'
'generative model'	'Semantic role labeling'	'lattice model'	'rule induction'
'search algorithm'	'Continuous-time Markov chain'	'automatic image annotation'	'Uncertainty quantification'
'sample size determination'	'Open Knowledge Base Connectivity'	'computational geometry'	'Computer architecture'
'anomaly detection'	'Propagation of uncertainty'	'Evolutionary algorithm'	'Best-first search'
'sentiment analysis'	'Fast Fourier transform'	'web search query'	'Gaussian random field'
'semantic similarity'	'Security token'	'eye tracking'	'Support vector machine'
'logic programming'	'machine translation'	'query optimization'	'ontology language'
'Hyperspectral imaging'	'middleware'	'Newton's method'	'big data'

Table 3: The full test queries set used for the system evaluation.