UNC SILS at TREC 2019 Precision Medicine Track

Jiaming Qu and Yue Wang

School of Information and Library Science
University of North Carolina at Chapel Hill
Chapel Hill, NC
jiaming@ad.unc.edu, wangyue@email.unc.edu

Abstract. This paper describes our participation in the scientific abstract retrieval task of TREC 2019 Precision Medicine Track. Our approach has two major components. First, we expand the original disease and gene terms using biomedical knowledge bases to improve recall of the initial retrieval. We then improve precision by promoting treatment-related publications to the top using a machine learning reranker trained on 2017 and 2018 relevance judgments combined. Batch evaluation results show that the proposed approach effectively improves P@10 compared to the baseline model.

1 Introduction

TREC Precision Medicine track aims to address an important medical retrieval problem: given a patient's disease, gene variation, demographic or other information, how to effectively retrieve relevant scientific papers or clinical trials for physicians to make decisions. This is the third year of the Precision Medicine track, which consists of two tasks as previous years. Both tasks are about document retrieval, but one for scientific abstracts in PubMed/MEDLINE[1] and another for clinical trials in ClinicalsTrial.gov[2].

The School of Information and Library Science (SILS) at the University of North Carolina at Chapel Hill (UNC) participated in the scientific abstract retrieval task in this year's PM Track. In this paper, we discuss our strategy to tackle the problem. In section 2, we provides a general overview of our strategy, including how to parse and index the documents. In section 3, we demonstrate in detail our retrieval strategy which consists of a two-direction approach. In section 4, we report the retrieval performances evaluated by TREC. In section 5, we summarize our work and propose potential improvements and directions for future research.

[1] https://www.ncbi.nlm.nih.gov/pubmed/
[2] https://clinicaltrials.gov/

2 Framework Overview

This year's scientific abstract retrieval task continues using the MEDLINE corpus which is a snapshot of PubMed abstracts, but does not include conference papers as extra corpus as previous years. All the PubMed article abstracts are in XML files, with rich information such as ID, title, abstract content, author, headings, journal information, and etc.

2.1 Document processing and Indexing

The first step is to index the corpus, for which we use Apache Lucene[3], an open-source, high-performance and publicly-available searching engine toolkit in Java. In this task, for each article we only index three fields, as is shown in Table 1. Throughout the whole task, we use the Okapi BM25 [4] ranking algorithm which has been widely used in various industry-level applications. In this paper, we use Lucene's default parameter settings of Okapi BM25 ($k = 1.25$ and $b = 0.75$). Check: and standard analyzer with lowercasing and removing English stopwords.

Table 1. Three fields indexed for each scientific paper

Fields	Analyzed	Stored
Title	True	True
Abstract	True	True
ID	False	True

There are documents with slightly different abstract but the same document ID (PMID), for a paper is added into the corpus every time after it is revised. To prevent duplicate document IDs, we simply index a document at the first hit of its PMID and ignore the later versions.

2.2 Two-stage retrieval framework

Our retrieval framework consists of two major components, namely query expansion and re-ranking, as is shown in Figure 1. Since the standard number of retrieved documents for each topic is 1000, we aim to maximize Recall@1000 by query expansion at the first step, during which we also tune weights of expansion terms. Then we train a pointwise learning-to-rank model (logistic regression model) that predicts the probability of a document's being relevant to the query, and the probability is used to re-rank the initially retrieved documents. All the training and parameter tuning are done on the 2017 and 2018 topics, using the released relevance judgement files with the true relevance grades. On 2019 topics, we first apply the query expansion module in the initial retrieval stage, and then rerank the retrieved results using the trained reranker.

[3] http://lucene.apache.org/
[4] https://en.wikipedia.org/wiki/Okapi_BM25

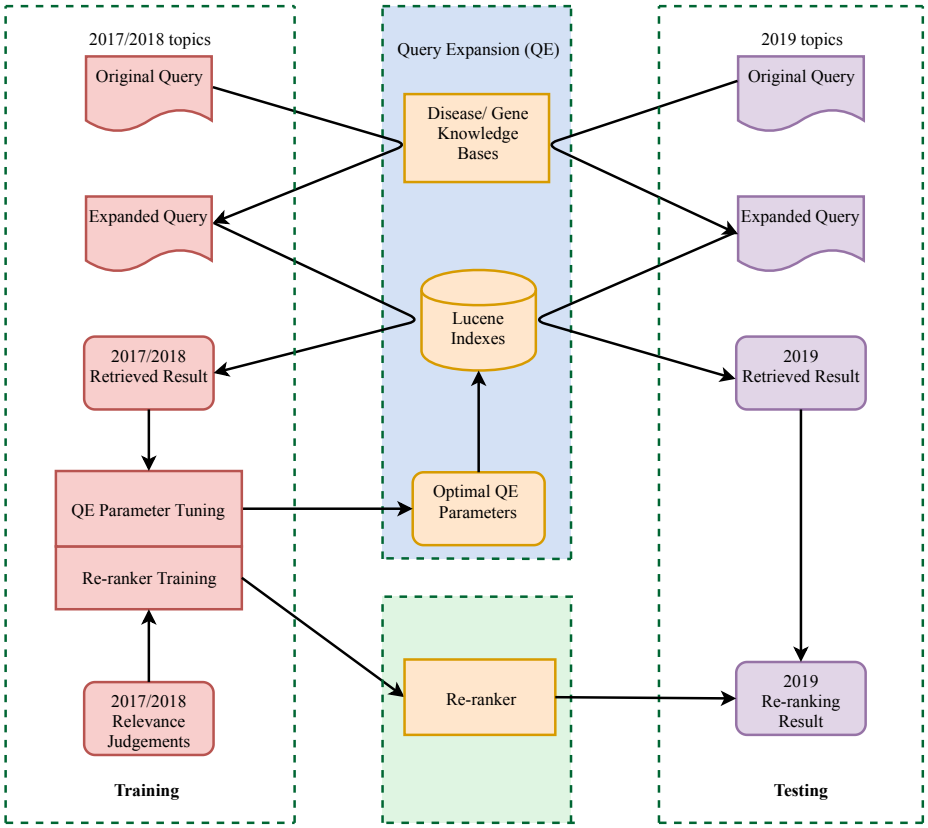

Fig. 1. An overview of the retrieval architecture.

3 Methodology

In this section, we describe our proposed retrieval framework. First we discuss how we consult external knowledge bases to expand disease and gene terms in the query to solve the mismatch problem. Then we discuss how we train a relevance model to further refine the ranking.

3.1 Query Expansion

The goal of query expansion is to use external knowledge bases to expand the original term which helps the query to match more documents which do not contain that term. Since both disease and gene information are given regarding to a patient, we expand both two fields in knowledge bases separately.

Diseases In this task we use a public knowledge base API called Lexigram[5]. It contains not only the MeSH ontology, but also other knowledge bases like the Systematic Nomenclature of Medicine Clinical Terms (SNOMED CT) and the International Classification of Diseases (ICD). By integrating three professional knowledge bases together, this knowledge base provides broader and more accurate terms for disease expansion.

In selecting an appropriate query expansion structure, at first we included as many expansion terms as possible, but later we found that including a disease's "parent" and "children" terms bring too many irrelevant terms into the query. Therefore, these terms are not included as expansion terms for a disease, and we only include the *preferred term* and *synonyms* of a disease for query expansion. The table below illustrates how expansion terms for a disease term are different in two medical knowledge bases (see Table 2).

Table 2. Example of disease expansion terms in two different knowledge bases

	Lexigram	**MeSH**
Original Disease Term	cholangiocarcinoma	cholangiocarcinoma
Preferred Disease Term	cholangiocarcinoma of biliary tract	cholangiocarcinomas
Synonyms	bile duct carcinoma, bile duct adenocarcinoma, cholangiocellular carcinoma	Intrahepatic Cholangiocarcinoma, Extrahepatic Cholangiocarcinoma

Another reason of using Lexigram is that it can recognize disease terms from input text and automatically generate expansion terms. However, if using the MeSH knowledge base, we need to look up the disease first to get a unique identifier, and then retrieve expansion terms according to that identifier. This brings a problem that when the disease in the query topic is a rare disease or does not match the standard disease name in MeSH, the identifier is hard to be found without human efforts or domain knowledge.

Apart from the preferred term and synonyms, acronyms are also used as expansion terms in this paper. Consider such a snippet in a paper's abstract which is relevant to *lung cancer*:

> "Microarray analyses have revealed significantly elevated expression of the proto-oncogene ROS1 receptor tyrosine kinase in 20-30% of **non-small cell lung carcinomas (NSCLC)**. Selective and potent ROS1 kinase inhibitors have recently been developed and oncogenic rearrangement of ROS1 in **NSCLC** identified. We performed immunohistochemical evaluation of expression of ROS1 kinase and its downstream molecules in 399 **NSCLC** cases. ROS1 expression in primary and recurring lesions of 92 recurrent **NSCLC** cases was additionally analyzed."

[5] `https://www.lexigram.io/`

In this text, we can see that the disease *non-small cell lung carcinomas* is only written in its full name when it appears in the paragraph for the first time, then it is written in its acronym form *NSCLC* afterwards for simplicity. This is very common in biomedical scientific papers that people use short acronyms instead of full names. For this paper, we use the simple regular expression `<Disease Name> \([A-Z]+\)` in the corpus to match a pattern that a disease name is followed by a capitalized term in parentheses to retrieve acronyms for each disease.

Genes For query expansion of the gene field, we enrich the original gene term with the genetic dataset provided by the National Center for Biotechnology Information (NCBI)[6]. We simply include the aliases for each gene term by looking it up in the dataset (see Table 3).

Table 3. Example of gene expansion terms in the NCBI gene list

	NCBI Gene List
Original Gene Term	KRAS
Aliases	C-K-RAS|CFC2|K-RAS2A|K-RAS2B|K-RAS4A|K-RAS4B K-Ras|KI-RAS|KRAS1|KRAS2|NS|NS3|RALD|c-Ki-ras2

To summarize, for each query topic the disease term is expanded to its preferred term, acronym and synonyms, and the gene term is expanded to its aliases. This helps enriching the information in the original query and matching those potentially relevant documents which do not mention the raw query terms but terms referring to the same disease or gene. We then tune grid search? the weights of expanded terms on the 2017 and 2018 topics to optimize R@1000, and the result is shown in Table 4.

Table 4. Weights for each expanded term group

Field	**Query Term**	**Weight**
Diseases	Original Disease Term	1
	Disease Preferred Term	0.1
	Disease Synonyms	0.1
	Disease Acronyms	0.5
Genes	Original Gene Term	1
	Gene Aliases	0.3

[6] ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/

Overall Query Structure For each topic, we expand the query in terms of disease and gene, and other fields are not used in this paper. After these two fields are expanded in knowledge bases, weights are assigned to each term. We use the OR operator between all the terms, because it is impossible for a paper to contain all the query terms after expansion. We use the SHOULD operator for the title field and the MUST operator for the content field, which means it is not required for a document to match the query in its title but required to match the query in its content.

3.2 Re-ranking

After expanding the original query to optimize R@1000, we aim to refine the ranking by pushing the most relevant documents to the top. We first explore a heuristic approach, and then discuss features to train a logistic regression classifier. The training step is done on the 2017 and 2018 data.

A Heuristic Approach Before starting training the reranker which helps to judge how much a document is relevant to the PM topics, we explore a heuristic approach to re-rank the retrieved results.

By exploring the title and content of top 10 retrieved results after the query expansion, we notice that almost all the relevant articles have the disease term in the title and most of the non-relevant articles do not. Therefore, we hypothesize that a relevant article should have the disease term in the title, otherwise it should be penalized to a lower relevance score. Based on this idea, we explore a heuristic approach of punishing those articles without the disease term by multiplying their original scores with a penalty factor between 0 to 1.

Based on the 2017 and 2018 data, we heuristically set the penalty factor to 0.6, and the performance of this retrieval shown in Section 4 proves that simple as this heuristic approach is, it does help to push relevant documents upwards and to improve P@10.

Learning to Rank The heuristic approach is the starting point of re-ranking the retrieved result to enhance P@10. However, this approach is based on human efforts of reading and exploring the results, therefore, we aim to leverage the pointwise learning-to-rank idea and machine learning techniques to learn a ranker which can automatically tell whether a paper is relevant to the PM track or not. According to the PM track overview [1], an article which is identified as relevant to the PM track should focus on treatments. Thus, we hypothesize that a relevant article should cover more words related to treatments instead of laboratory experiments. Also, in this year the treatments of each article are also extracted and provided as a new resource, and we include the count of treatments as a feature.

Table 5 lists query-document features and document-specific features used in the ranker. The features of whether the disease term appears in the title is categorical, and all the other features are numerical as we simply count the term

frequency of each keywords. We choose high-level features instead of pure word features from the bag-of-words model because such features would be too sparse to have a good performance.

Table 5. Features in the learning-to-rank model

Feature Description	Data Type
Whether the disease name in the query appears in the title	Categorical
Number of positive keywords in the title	Numerical
Number of positive keywords in the abstract	Numerical
Number of negative keywords in the title	Numerical
Number of negative keywords in the abstract	Numerical
Number of unique treatments in the article	Numerical

The next step is to generate the positive keywords and negative keywords. Oleynik *et al.* use the Latent Dirichlet Allocation model to do Topic Modeling among the relevant articles and non-relevant articles to find positive and negative keywords [2]. These keywords are supposed to have the strongest indication of two labels. The lists of positive and negative keywords are shown in Table 6.

Table 6. Positive, negative and heading keywords

Positive Keywords	treatment, survival, prognostic, clinical, prognosis, therapy outcome, resistance, targets, therapeutic, immunotherapy
Negative Keywords	pathogenesis, tumor, development, model, tissue, mouse specific, staining, dna, case, combinations

3.3 Ranker Construction

We train a Logistic Regression model to predict the probability that a document is relevant to the PM track using the scikit-learn[7] library. For each retrieved article, the feature vector is generated based on the feature table above and the probability of a positive (relevant) label is predicted using the model.

We use both the predicted probability of how relevant an article is to the PM track and the raw BM25 score for re-ranking. However, since the probability is a number between 0 to 1 and the raw score could be as large as 70, we should keep them at the same scale. Therefore, we do the min-max transformation on both scores and they are scaled to a number between 0 to 1. Then for each article, a new score is generated by adding up the raw BM25 score and the probability from the classifier, and is used for re-ranking.

[7] https://scikit-learn.org/

However, in experiments we find that applying the reranker to all the 1000 results do not improve the precision, but hurt the performance instead. We suppose that it is because the positive instances in the training set merely come from past teams' top retrieved results, which are much fewer than the negative instances. Thus, the ad-hoc relevance judgment by experts are only applied on documents which rank at top in each team's submitted results, which leads to a different distribution of instances in the training and testing set. Therefore, instead of applying the reranker on all the 1000 retrieved documents to do re-ranking, we only re-rank the top 50 documents, after testing different top K documents to re-rank by optimizing P@10 on the 2017 and 2018 topics.

In summary, we train the reranker on the 2017 and 2018 topics and find that the optimal number to re-rank is 50. When we run queries from the 2019 topics, we follow the heuristic approach first, *i.e.*, we punish a document which does not mention the disease term in its title by multiplying its raw score by a penalty factor of 0.6, based on which the results are re-ranked for the first time. We then transform the raw BM25 scores using a min-max scaler to a number between 0 and 1. Afterwards, we only input the top 50 documents into the reranker and add up the BM25 score and probability. Based on new scores, the re-ranking is done for the second time and a final ranked list is returned for relevance judgment.

4 Evaluation

We submit 4 runs for the scientific abstract retrieval task and the evaluation results are summarized in the table below.

Table 7. Evaluation Results

Run Name	P@10	R-prec	infNDCG
sils_run1	0.5225	0.2858	0.4490
sils_run2	0.5925	0.2805	0.4692
sils_run3	0.5925	0.2757	0.4620
sils_run4	0.5900	0.2757	0.4625

sils_run1 is the baseline which does not re-rank the results after retrieval. This run expands the query and returns a ranked list from Lucene according to BM25 relevance scores.

sils_run2 adds the heuristic re-ranking on the baseline.

sils_run3 adds the trained re-ranker on the baseline, with features of "Whether the disease name appears in the title", "Count of positive/negative keywords in title/abstract".

sils_run4 adds the trained re-ranker on the baseline, compared to ***sils_run3***, this run has one more feature of "count of unique treatments" in the classifier.

By comparing ***sils_run1*** and ***sils_run2*** or ***sils_run3***, we could see that re-ranking improves P@10 by pushing the most relevant documents to the top.

However, by comparing ***sils_run2*** and ***sils_run3***, we could see that P@10 does not differ too much from the heuristic approach or the learning-to-rank approach. Finally, by comparing ***sils_run3*** and ***sils_run4***, we could see that adding the new feature in 2019 does not help too much.

5 Conclusion

We participate in the 2019 Precision Medicine track and submit 4 runs for the scientific abstract retrieval task. Our best run incorporates two steps, which are query expansion and re-ranking. We expand disease and gene query terms by knowledge bases to improve recall, then train a Logistic Regression classifier to predict how much an article is relevant, based on which we re-rank the retrieval results. Results show that the proposed approach effectively improves the performance in terms of P@10 compared to the baseline model.

This work opens up interesting questions for future studies. In terms of learning-to-rank models, one could explore richer features (ranking scores produced by different relevance models) and more expressive function family (ensembles such as gradient boosted trees). It would also be interesting to study the actual information need of oncologists. For instance, it would be interesting to study if PM searches are mostly high-precision tasks (as indicated by the name *precision* medicine), or high-recall tasks (more like a systematic literature review of relevant treatments), or the recall levels being case-specific. A system that's aware of the desired level of recall would make informed decision in presenting (whether ranking or selecting) documents.

References

1. Kirk Roberts, Dina Demner-Fushman, Ellen M Voorhees, William R Hersh, Steven Bedrick, Alexander J Lazar, and Shubham Pant. Overview of the trec 2017 precision medicine track.
2. Michel Oleynik, Erik Faessler, Ariane Morassi Sasso, Arpita Kappattanavr, Benjamin Bergner, Harry Freitas Da Cruz, Jan-Philipp Sachs, Suparno Datta, and Erwin Böttinger. Hpi-dhc at trec 2018 precision medicine track. 2018.