Learning from Medical Summaries: The University of Michigan at TREC 2015 Clinical Decision Support Track

Fengmin Hu[1], Danny T.Y. Wu[1], Qiaozhu Mei[1,2], V.G.Vinod Vydiswaran[3,1]
[1]School of Information [2]Department of Electrical Engineering and Computer Science
[3]Department of Learning Health Sciences
University of Michigan
{hufm, tzuyu, qmei, vgvinodv}@umich.edu

ABSTRACT

The goal of TREC 2015 Clinical Decision Support Track was to retrieve biomedical articles relevant for answering three kinds of generic clinical questions, namely diagnosis, test, and treatment. In order to achieve this purpose, we investigated three approaches to improve the retrieval of relevant articles: modifying queries, improving indexes, and ranking with ensembles. Our final submissions were a combination of several different configurations of these approaches. Our system mainly focused on the summary fields of medical reports. We built two different kinds of indexes – an inverted index on the free text and a second kind of indexes on the Unified Medical Language System (UMLS) concepts within the entire articles that were recognized by MetaMap. We studied the variations of including UMLS concepts at paragraph and sentence level and experimented with different thresholds of MetaMap matching scores to filter UMLS concepts. The query modification process in our system involved automatic query construction, pseudo relevance feedback, and manual inputs from domain experts. Furthermore, we trained a re-ranking sub-system based on the results of TREC 2014 Clinical Decision Support track using Indri's Learning to Rank package, RankLib. Our experiments showed that the ensemble approach could improve the overall results by boosting the ranking of articles that are near the top of several single ranked lists.

1. INTRODUCTION

Recent years have seen significant advances in the field of healthcare, with smarter computerized clinical decision support platforms, better personalized healthcare plans, and availability of large data sets for research and quality improvement studies. According to Accenture's report of healthcare IT Vision of 2015 [9] and the survey of doctor's attitudes towards to healthcare IT functions [10], the digital channels are dramatically influencing the healthcare industry and are expected to do so in near future.

While making clinical decisions, physicians often seek out additional information to provide the best care for their patients. There is a growing interest among medical informatics researchers on clinical decision support (CDS) systems that are designed to assist physicians and other health professionals with clinical decision-making tasks. The TREC Clinical Decision Support Track aims to help develop techniques to address these tasks and link medical cases to information relevant to patient care. The track, in its second year, continued to focus on retrieving biomedical articles relevant for answering generic clinical questions about medical records. There were two tasks in TREC 2015 Clinical Decision Support track. Task A was identical to the TREC 2014 CDS track, while in Task B, participants were additionally provided with the diagnosis information for the test and treatment related queries.

The target document collection for this track was a snapshot of the Open Access Subset of PubMed Central. Participants were also provided a set of case reports, such as those published in biomedical articles, as idealized representations of actual medical records. There were two formats of the case narratives – a long paragraph "description" narrative of the complete account of a patient's visit, including details such as their vital statistics, drug dosage, etc.; and a simplified "summary" narrative that contained less irrelevant information. Participants were challenged with retrieving relevant biomedical articles for those specific case reports to answer questions belonging to one of three generic clinical categories; namely "What is the patient's diagnosis?", "What tests should the patient receive?", and "How should the patient be treated?".

We reviewed the main approaches following by numerous research teams that participated in the first TREC CDS track in 2014. After reviewing the papers and results from the previous year, we found that there was no clear choice between two versions of case narratives. "Summary" narratives were considered as the first choice by most groups since they contained less irrelevant information and were easier for query modification than "description" narrative. However, several groups got slightly better performance by using the "description" narrative. In [14], Xu et al. concluded that using summaries as queries and applying the standard bag-of-words retrieval function, BM25, with pseudo-relevance feedback, could provide a satisfactory baseline performance. Teams also investigated various query modification and reformulation techniques. Some groups applied query reduction approaches by filtering terms with the help of external medi-

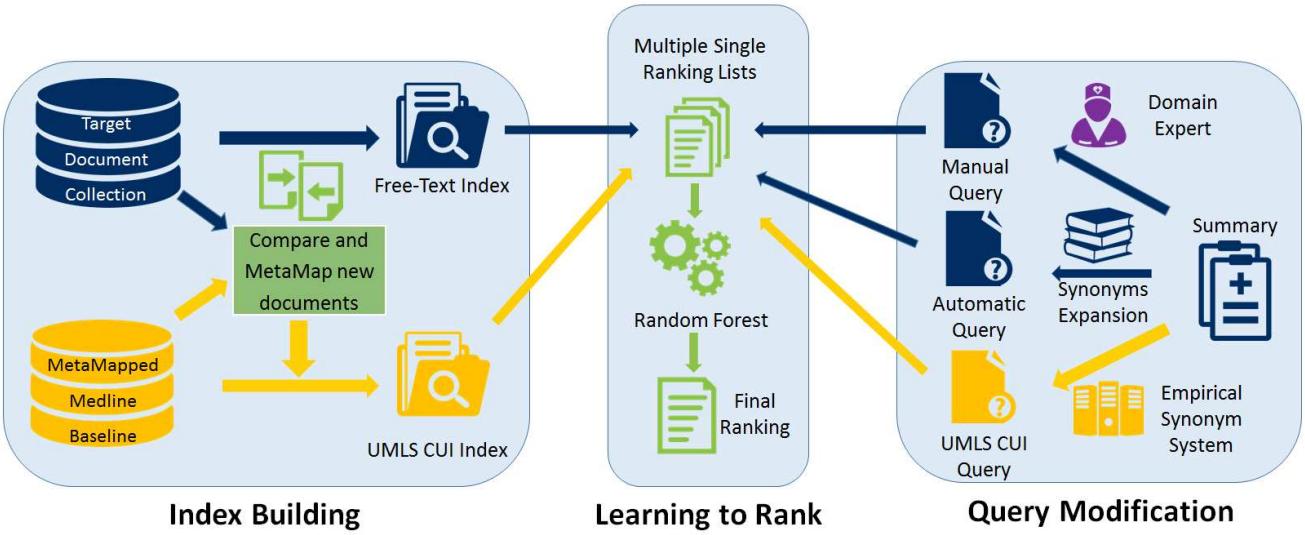

Figure 1: System Architecture

cal knowledge resources or domain experts [11, 12]. Other participating teams employed query expansion approaches, such as pseudo-relevance feedback [3, 8, 11, 14] and synonyms expansion [12]. Those techniques mostly appeared effective and they improved the overall retrieval performance by boosting articles which were more similar to the top articles. However, they were also prone to query drifting.

Besides traditional information retrieval methods, re-ranking methods were also widely studied in [3, 4, 8, 11]. Soldaini et al. [11] proposed several re-rankers including biographical re-ranker, MetaMap similarity re-ranker, etc. that respectively took features such as the biographic characteristics and the number of concepts in Unified Medical Language System (UMLS) recognized by MetaMap into consideration to generate a final article score. Fusion-based re-ranking approach also proved to be an effective approach and improved the performance significantly [3, 8, 11]. Soldaini et al. [11] also developed a voting-based fusion algorithm to compute the final ranking as an average of four other rankers. André Mourão et al. [8] applied Reciprocal Rank Fusion (RRF) algorithm to combine different retrieval functions. In [3], Choi and Choi trained two kinds of task-specific classifiers and combined the relevance ranking with task-specific ranking using the Borda-fuse algorithm [1].

Though many traditional and state-of-the-art information retrieval approaches have been investigated last year, the mean retrieval scores were relatively poor and lower than expected in general. Most participants only focused on the free text of target document collection and employed the basic pseudo-relevance feedback model as well as straightforward re-ranking strategies. In our system, we focused on two kinds of information: free text information and UMLS concept information recognized by MetaMap. To achieve this, we built two indexes – an inverted index on the free text and a second index on the UMLS concepts recognized by MetaMap in the entire article. Queries were mostly based on the "summary" narrative, and were modified either automatically or with the help of a domain expert. Two versions of queries were presented, a free-text version for the first inverted index and a UMLS Concept Unique Identifier (CUI) version for the second UMLS concept index. We also considered multiple variations of including UMLS concept information at paragraph or sentence level and experimented with different thresholds to filter UMLS concepts based on their MetaMap scores. Finally, we designed and trained a re-ranking sub-system using the Random Forest algorithm [2] to combine the ensemble of single ranked lists into a final ranked list.

In the following sections, we will describe in more detail how we deployed the three major approaches we studied, viz. improving indexes, modifying queries, and re-ranking with ensemble, and how these approaches impacted the retrieval performance of our system.

2. SYSTEM ARCHITECTURE

First, we provide a high-level overview of our system architecture and describe the key components briefly. The system architecture is shown in Figure 1. In the figure, blue and yellow colors represent the components that work with free-text information and UMLS concepts, respectively.

The primary components of our approach are as follows:

1. **Building Indexes:** We built two kinds of indexes. First, we built an inverted index over the Target Document Collection provided to all TREC-CDS participants. Next, we processed the title and abstracts of all documents through MetaMap. This let us identify all medical concepts mentioned in the title and abstract sections. The MetaMap matching scores were used to retain only those concepts that were scored higher than a preset threshold. The second set of indexes were built over the unique identifiers assigned by MetaMap to the filtered concepts.

2. **Query Modification:** We processed the "summary" narratives through MetaMap to identify medical concepts mentioned in the narratives. Two kinds of queries were formulated: one consisting of the concept words themselves, and the other consisting of the unique identifiers for the concepts (CUIs). The final queries were constructed by either expanding the queries automatically or using inputs from a domain expert.
3. **Retrieval:** The free-text queries and the UMLS CUI queries were submitted to appropriate indexes to get an ensemble of single ranked list results.
4. **Re-ranking with ensembles:** We designed and developed a re-ranking sub-system to combine the ranked results from individual queries. This sub-system considered the rank orders (scores) for articles from individual ranked lists as features and trained a re-ranking model. We used the Random Forest algorithm [2] to learn the weights of individual rankers and combine them to get the final ranked list.

Since the setup for the Task A of TREC-CDS 2015 was identical to the task in the previous year's TREC-CDS task, we could train over models on the queries and results from the previous year. On the other hand, the Task B of TREC-CDS 2015 was a new task. So, we could not train new models for Task B. Instead, we adopted the model learned in Task A by adding additional information about the diagnosis that was available for Task B.

3. IMPLEMENTATION

We now describe the detailed implementation of our system for TREC-CDS 2015 track.

3.1 Preprocessing

Our original goal was to parse the entire collection through MetaMap to identify all medical concepts in the documents. However, since MetaMap is quite time-consuming to run over the full-text articles in the target document collection, we decided to focus only on the titles and abstracts of the articles. We first checked the MetaMapped Medline Baseline results[1], provided by the National Library of Medicine (NLM). The Medline articles processed by NLM contain unique PubMed identifiers (PMIDs) that were also found in the TREC-CDS target document collection, thereby facilitating merging of the two datasets and avoiding duplication of efforts.

On parsing the TREC-CDS target document collection, we found that 46,648 articles (6.36% of the total documents in the collection) did not mention PMIDs, and hence could not be linked to the MetaMapped Medline Baseline corpus. We further analyzed these articles. Most of these articles belong to one of three categories, viz. "abstracts" (42.09%), "book-reviews" (23.45%), or "corrections" (10.53%). Only 3,566 (7.64%) articles were labeled as "research articles". Based on the analysis of the distribution of article types in the target collection and qrel files conducted by Gobeill et al. [5], we found that articles that belonged to the types "abstracts", "book-reviews", and "corrections" consist of less than 4.8% of files found in the qrel file, while research articles accounted for 52.2% of files. This finding also agrees with our intuition that the articles of the former three types are generally short and less likely to be found relevant for answering clinical questions. Hence, we decided to only process the missing articles that were classified as "research articles" and ignore the remaining types.

In addition to articles that did not have PMIDs, we also found 30,282 articles with PMIDs in the target document collection that were not found in our snapshot of MetaMapped Medline Baseline corpus. Those, when added to the 3,564 "research abstracts" identified above, gave us a set of 33,848 articles with missing MetaMap information. We extracted the titles and abstracts from these articles and ran MetaMap on this set.

3.2 Parsing

Once we collected the MetaMapped results for all documents in the target collection, the next task was to parse the MetaMap output. For each article, the MetaMap Machine Output[2] contains an 'utterance' field, which is a sequence of tokens into which the input text was chunked. Each utterance is followed by one or more sequences of the following three sub-components:

1. **phrase**: A sub-sequence of the utterance's tokens, along with its syntax. A phrase always appears with corresponding **candidates** and **mappings** objects.
2. **candidates**: A possibly empty list of UMLS candidate concepts identified in the phrase.
3. **mappings**: A possibly empty list of MetaMap's final mappings and a subset of the candidate set. The mapping list is empty only if the candidates list is empty.

For each candidate object, we collected the following three fields:

1. **NegScore**: The negative mapping score
2. **CUI**: The concept unique identifier
3. **String**: The string representation of the concept as it appears in the utterance

3.3 Indexing Text and Concepts

We built two kinds of indexes using the Indri search engine[3], viz. a free-text inverted index and several UMLS CUI indexes. For the free-text index, we used Indri's built-in XML parser directly to parse and index the article text. For the UMLS CUI indexes, however, some additional processing was required.

MetaMap assigns score to each concept identified in the text. Perfect matches get a score of 1000, and other matches with ambiguities get lower scores. Our previous experiments with MetaMap scores show that scores above 700 are potentially relevant while those above 800 are significant. We parse the title and abstracts of all articles and construct three versions of concept documents – one with all concepts identified in the input text, and two other versions where only concepts with scores over 700 and over 800 are retained. We will refer to these three versions as UMLS-CUI-all, UMLS-CUI-700, and UMLS-CUI-800, respectively. As the threshold value is increased, the concept documents have progressively lesser

[1] https://skr.nlm.nih.gov/resource/MetaMappedBaselineInfo.shtml

[2] https://metamap.nlm.nih.gov/Docs/2012_MMO.pdf

[3] http://www.lemurproject.org/indri.php

number of concepts in them.

We also experimented with the granularity of the documents themselves. Instead of treating the title and abstract as one document, we split the abstract into multiple sentences and treated each sentence as its own document. This approach increases the number of documents in the index, but each document (a sentence) was short and had fewer relevant concepts. We constructed a fourth set of concept documents this way, and retained all concepts identified within each sentence (i.e. we did not filter based on the concept scores). We will refer to this version as UMLS-CUI-sen.

Once the four versions of the concept documents are obtained, we build the four corresponding UMLS-CUI indexes using Indri.

3.4 Query Modification

Next, in the query modification stage, we constructed effective queries based on the "summaries" narrative and the query type. Since we built two kinds of indexes, we constructed two kinds of queries, correspondingly.

3.4.1 Queries for free-text index

We investigated and evaluated several approaches and configurations to construct keyword queries for the free-text index. First, using the Indri query language construct `#syn` to indicate synonyms, we expanded the question types thus:

- Diagnosis ⇒ `#syn`(Diagnosis, Dx)
- Test ⇒ `#syn`(Test, Examination)
- Treatment ⇒ `#syn`(Treatment, Drug, Therapy)

Next, we looked at the "summary" narratives in the given case reports. We constructed one query based on just the summary and another query after removing common stopwords. Finally, the queries were shown to a domain expert, who provided us useful recommendations and highlighted important terms in each case report. This input guided us to construct a set of manual queries.

3.4.2 Queries for UMLS-CUI indexes

We created three versions of queries for UMLS-CUI indexes as follows:

1. A query with summary words and the type information.
2. Based on the inputs from the domain expert who highlighted important terms in the summaries, we constructed queries using the CUIs of the highlighted terms as well as the MetaMapped type information.
3. We used an in-house repository of empirical synonyms for medical concepts. This synonym set was collected based on frequent query rewrites in EMERSE, an Electronic Medical Record Search Engine. [13]. We identified key terms in summaries and expand their CUIs automatically using the empirical synonym set.

We ran the generated queries generated against different Indri indexes to get an ensemble of single ranked list results.

3.5 Retrieval

In the retrieval phase, we explored different configurations of pseudo-relevance feedback and parameter settings for whether to include the query type information and whether to remove stopwords. We tested and evaluated the default setting of pseudo-relevance feedback and tuned three parameters, viz. the number of feedback documents (`fbDocs`), the number of feedback terms added to the query (`fbTerms`), and the weight assigned to the original query in contrast to the feedback terms (`fbOrigWeight`). This way, we created twelve individual ranking configurations that are listed in Table 1.

3.6 Learning to Rank Model

Once the individual ranked lists were available, we set forth to combine them using a Learning to Rank framework. Learning to Rank refers to a set of machine learning techniques using to train a model to combine an ensemble of ranked lists to improve the overall ranking performance [6]. Since the TREC 2015 CDS track was similar to the previous year's track, we could utilize the annotated list of relevant documents (in the qrel file) to design and learn a supervised learning to rank model and determine the weights of different retrieval configurations. We used the RankLib library[4] to train a learning to rank model. RankLib is part of the Lemur project and supports eight popular machine learning algorithms, implements many retrieval metrics, and provides multiple ways to carry out the evaluation. We first split the case reports of each type from TREC 2014 CDS track randomly into training and test sets, and compared several models using different learning algorithms. Some of the techniques we tried included Multiple Additive Regression Trees (MART), also known as Gradient boosted regression trees, AdaRank, LambdaMART, and Random Forest. The performance of Random Forest was found to be slightly better and more robust when evaluated using 5-fold cross-validation experiments. We chose Random Forests to implement our final learning to rank model.

The goal of the second task (Task B) was to investigate if availability of the diagnosis information can help retrieve more relevant articles in the test and treatment related queries. Unlike the first task, the second task was new and, hence, did not have any training data to train learning to rank models. However, we observed that most diagnosis information only contained one or two words. So, we assumed that adding diagnosis terms would not significantly affect the weights for the learning to rank models trained for the first task. Hence, although we constructed new queries using the diagnosis information, we used the same set of individual ranker configurations for both tasks and did not update the learning to rank models.

4. RESULTS AND DISCUSSION

We now describe and discuss the performance of our six submitted runs over the two tasks, along with additional analysis of the best performing retrieval configurations.

4.1 Description of submission runs

Each participating team was allowed to submit at most three runs to each of the two tasks. All our submissions except the last submission for Task B were based on the same Random

[4]http://www.lemurproject.org/ranklib.php

Table 1: Individual Ranking Configurations

#	RunID	Source[1]	TypeInfo	Pseudo Relevance Feedback[2]			Stopword Removal	Index
				#Docs	#Terms	Weight		
1	Text_0	summary	Yes	20	10	0.5	No	Text
2	Text_1	summary	No	20	10	0.5	No	Text
3	Text_2	summary	No	20	10	0.5	Yes	Text
4	Text_3	summary	Yes	20	10	0.5	Yes	Text
5	Manual_0	highlighted terms	Yes	10	5	0.5	Yes	Text
6	Manual_1	highlighted terms	Yes	10	5	0.67	Yes	Text
7	MetaMap_0	EMERSE	Yes	10	5	0.5	Yes	UMLS-CUI-700
8	MetaMap_1	EMERSE	Yes	10	5	0.5	Yes	UMLS-CUI-all
9	MetaMap_sent	EMERSE	Yes	10	5	0.5	Yes	UMLS-CUI-sen
10	MetaMap_Manual_0	highlighted CUIs	Yes	10	5	0.5	Yes	UMLS-CUI-700
11	MetaMap_Manual_1	highlighted CUIs	Yes	10	5	0.5	Yes	UMLS-CUI-all
12	MetaMap_Manual_sen	highlighted CUIs	Yes	10	5	0.5	Yes	UMLS-CUI-sen

[1] **summary**: summary field of a topic; **highlighted terms**: terms highlighted by a domain expert; **EMERSE**: CUIs suggested by an Electronic Medical Record Search Engine; **highlighted CUIs**: CUIs of terms highlighted by a domain expert.
[2] **#Docs**: the number of feedback documents (`fbDocs`); **#Terms**: the number of feedback terms added to the query (`fbTerms`); **Weight**: the weight assigned to the original query terms (`fbOrigWeight`).

Forest re-ranking model; and the submissions differed in the subset of ranking lists used in the learning to rank model ensembles.

The detailed descriptions of our official submissions are listed below. The numbers refer to the row numbers in Table 1.

Task A

1. ***FusionAuto***: This submission run ensembles all individual ranking lists that are automatically produced; specifically #1–4 (*Text_0/1/2/3*) and #7–9 (*MetaMap_0/1/sen*).
2. ***FusionManual***: This submission run ensembles all individual ranking lists that involved some manual input, and two automatically produced ranked list; specifically #3–4 (*Text_2/3*), #5–6 (*Manual_0/1*), and #10–12 (*MetaMap_Manual_0/1/sen*).
3. ***FusionMAll***: This submission ensembles all individual ranking lists we created (#1–12).

Task B

1. ***FusionAutoB***: This submission run is generated based on the *FusionAuto* run of Task A. We add the text and UMLS-CUIs of diagnosis results into test and treatment queries, and apply the same learning to rank model to create the final ranking list. The submission run ensembles all individual ranking lists that are automatically produced; specifically #1–4 (*Text_0/1/2/3*) and #7–9 (*MetaMap_0/1/sen*).
2. ***FusionManB***: This submission run is generated based on the *FusionManual* run of Task A. We add the text and UMLS-CUIs of diagnosis results into test and treatment queries, and apply the same learning to rank model to create the final ranking list. The submission run ensembles all individual ranking lists that involved manual input, and two automatically produced ranked lists; specifically #3–4 (*Text_2/3*), #5–6 (*Manual_0/1*), and #10–12 (*MetaMap_Manual_0/1/sen*).
3. ***FusionAdv***: This submission run is generated based on the intuition that adding diagnosis results into test and treatment queries may lead to the results being more relevant to answering diagnosis clinical question. So, we constructed three different ranking lists: first, the *FusionManB* run; second, the *Notype* run, an ensemble of all individual ranking lists that did not include the type information; and third, the *Diagnosis* run, generated by changing all type information to Diagnosis and then merging the ranked lists.
Every article in the three ranking lists was assigned a ranking score, formulated as $score = 1001 - rank$, based on the article's rank in that ranking list. In other words, the top ranked article got a score of 1000, the next ranked article a score of 999, and so on. Articles that did not show up in a ranking list received a score of 0 for that list. The final score was computed as:

$$score = 0.8 \times score_{FusionManB} + 0.4 \times score_{Notype} - 0.2 \times score_{Diagnosis}$$

4.2 Results and Discussion

The performance evaluations of our runs and the median results among submitted TREC runs in Tasks A and B are shown in Table 2 and Table 3, respectively. The best values for each of the four metrics (viz. infAP, infNDCG, R-prec, and Precision@10) are highlighted in boldface. In addition to the submitted runs, we also evaluated the performance of the individual rankers used in the two tasks. The re-

Table 2: Performance evaluation of Task A

	infAP	infNDCG	R-prec	P@10
FusionAuto	0.0641	0.2445	0.1937	0.3733
FusionManual	0.0738	0.2815	**0.2256**	0.46
FusionMAll	**0.0816**	**0.2954**	0.2246	**0.47**
Median(Auto)	0.0414	0.2038	0.1615	0.3433
Median(Manual)	0.0499	0.2402	0.1735	0.3833
BIR-Text	0.0726	0.2614	0.2021	0.4
BIR-Manual	0.0787	0.2867	0.2205	0.4367
BIR-MetaMap	0.0244	0.1197	0.0996	0.1833

Table 3: Performance evaluation of Task B

	infAP	infNDCG	R-prec	P@10
FusionAutoB	0.0841	0.3298	0.2538	0.4633
FusionManB	**0.0955**	**0.3535**	**0.2767**	0.5233
FusionAdv	0.0951	0.3473	0.2731	**0.53**
Median(Auto)	0.0633	0.2794	0.2123	0.45
Median(Manual)	0.0666	0.2899	0.2035	0.4733
BIR-Text	0.0945	0.3453	0.2589	0.5
BIR-Manual	0.0985	0.3516	0.2618	0.51
BIR-MetaMap	0.0327	0.1505	0.1213	0.2333

sult tables also show the best performance obtained by the individual rankers of the following three types:

- **BIR-Text**: The best individual ranking configuration among the automatically generated ones using only the text information; specifically #1–4 (*Text_0/1/2/3*)
- **BIR-Manual**: The best individual ranking configuration among those that use only the text information, but involve some manual input from the domain expert; specifically #5–6 (*Manual_0/1*)
- **BIR-MetaMap**: The best individual ranking configuration among those that are automatically generated using only MetaMap information; specifically #7–9 (*MetaMap_0/1/sen*)

From the performance evaluation of our submissions, we found that our submissions are better than the median in all four metrics, suggesting that our approaches of index improvement, query modification, and re-ranking with ensembles are potentially useful in ranking articles for the clinical decision support task. The comparison between our submissions also verified some of our intuitions that the manual inputs from domain expert would improve the results significantly, and that including the diagnosis results could improve the results of test and treatment related queries.

By comparing individual rankings, we found that the best performing configurations all included the type information and removed stopwords. We also noticed that the results of best ranking configuration based on only the text information, *BIR-Text*, outperformed the results of *FusionAuto* and *FusionAutoB*, while the best ranking configuration using MetaMap, *BIR-MetaMap*, performed relatively poorly. This suggests that further investigation is required on the fusion algorithm to incorporate the MetaMap information more effectively.

4.3 Future work

We would like to develop a clinical decision support system that could retrieve relevant information in a fully automated fashion. In the submitted runs, we used inputs from a domain expert to alter the queries and it helped improve our results significantly. Our next step is to generate a set of rules to select keywords automatically, based on analyzing the nature of input we received from the domain expert. We want to explore the ReQ-ReC double loop system developed by Li et al. [7] as a potential framework to do so. Further, on comparing the results of *FusionManB* and *FusionAdv*, we found the results are comparable after adding the ranked lists for *Notype* and *Diagnosis*, even with manually set combination weights in *FusionAdv*. This suggests that additional research is needed to understand the type-specific ranking and learn the optimal combination weights.

5. CONCLUSION

In this paper, we described the participation of the University of Michigan in the TREC 2015 Clinical Decision Support track. We built an information retrieval system to retrieve biomedical articles for clinical decision support queries and investigated three major approaches to improve the retrieval performance, namely improving indexes, modifying queries, and re-ranking ensembles of ranked list results. The performance evaluations indicated that our submissions performed significantly better than the reported median performance. In future, we plan to continue our investigation on query modifications and re-ranking ensembles to improve clinical decision support and enable improved patient care.

6. ACKNOWLEDGMENTS

The authors acknowledge the timely contributions and assistance of the domain expert, Hanqi Tang, from Tsinghua University, during our participation in the TREC 2015 Clinical Decision Support track.

7. REFERENCES

[1] Javed A Aslam and Mark Montague. Models for metasearch. In *Proceedings of the 24th international ACM SIGIR conference on research and development in information retrieval*, pages 276–284, 2001.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] Sungbin Choi and Jinwook Choi. SNUMedinfo at TREC CDS track 2014: Medical case-based retrieval task. In *Proceedings of the 23rd Text REtrieval Conference Proceedings (TREC)*, 2015.

[4] Jean I. Garcia-Gathright, Frank Meng, and William Hsu. UCLA at TREC 2014 clinical decision support track: Exploring language models, query expansion, and boosting. In *Proceedings of the 23rd Text REtrieval Conference Proceedings (TREC)*, 2015.

[5] J. Gobeill, A. Gaudinat, E. Pasche, and P. Ruch. Full-texts representation with Medical Subject Headings, and co-citations network re-ranking strategies for TREC 2014 clinical decision support track. In *Proceedings of the 23rd Text REtrieval Conference Proceedings (TREC)*, 2015.

[6] Li Hang. A short introduction to learning to rank. *IEEE Transactions on Information and Systems*, 94(10):1854–1862, 2011.

[7] Cheng Li, Yue Wang, Paul Resnick, and Qiaozhu Mei. ReQ-ReC: High recall retrieval with query pooling and interactive classification. In *Proceedings of the 37th international ACM SIGIR conference on research and development in information retrieval*, pages 163–172, 2014.

[8] André Mourao, Flávio Martins, and Joao Magalhaes. NovaSearch at TREC 2014 clinical decision support track. In *Proceedings of the 23rd Text REtrieval Conference Proceedings (TREC)*, 2015.

[9] Kaveh Safavi and Rick Ratliff. Healthcare IT Trends. *Accenture Healthcare Technology Vision*, 2015. https://www.accenture.com/us-en/insight-healthcare-technology-vision-2015.aspx Last accessed January 31, 2016.

[10] Kaveh Safavi, Rick Ratliff, and Kip Webb. Healthcare IT Pain and Progress. *Accenture Doctors Survey*, 2015. https://www.accenture.com/us-en/insight-accenture-doctors-survey-2015-healthcare-it-pain-progress.aspx Last accessed January 31, 2016.

[11] Luca Soldaini, Arman Cohan, Andrew Yates, Nazli Goharian, and Ophir Frieder. Query reformulation for clinical decision support search. In *Proceedings of the 23rd Text REtrieval Conference Proceedings (TREC)*, 2015.

[12] Raymond Wan, Jannifer Hiu-Kwan Man, and Ting-Fung Chan. Query modification through external sources to support clinical decisions. In *Proceedings of the 23rd Text REtrieval Conference Proceedings (TREC)*, 2015.

[13] Danny T.Y. Wu, David A. Hanauer, Lei Yang, Kai Zheng, and Qiaozhu Mei. Towards intelligent and socially oriented query recommendation for electronic health records retrieval. In *Proceedings of the ACM SIGIR workshop on Health Search and Discovery (HSD)*, 2013.

[14] Tan Xu, Paul McNamee, and Douglas W Oard. HLTCOE at TREC 2014: Microblog and clinical decision support. In *Proceedings of the 23rd Text REtrieval Conference Proceedings (TREC)*, 2015.