

Project Report

Abstract

It is really difficult for Physicians to find relevant documents based on their clinical questions. For this problem the TREC Clinical Decision Support Task challenges participants to retrieve most relevant documents based on the clinical questions. There are three types of clinical question which needs to be answered and they are diagnosis, test and treatments. There are total of thirty topics which needs to be treated as a clinical questions or query.

So the main focus is on extracting relevant documents based on these topics. These documents should be such that they answer the question of physicians or provide help to them regarding the past similar cases. Each topic includes the case narrative in two forms- description and summary. The summary provides an small and simplified version of the case test whereas the description provides more complete information regarding the case.

There are a total of 733138 documents from which we need to extract the relevant documents based on the topic. These documents are in NXML format, and each documents has various fields out of which article-ID is going to be used to extract the document.

Related Work

The TREC CDS 2014 was a competition in 2014 and so there are many results based on that. There are many approaches made for TREC CDS 2014 with many different methods, but I have referred to some of them and discussing how they have used different techniques and different methods for it.

The first paper that I referred to is "San Francisco State University at TREC 2014: Clinical Decision Support Track and Microblog Track". In this paper they have discussed the methods they have used in different runs and different approaches for the query.

In their approach they have indexed the documents using Indri. In Indexing they have used krovetz stemmer for morphological normalization of the documents. During indexing they have also included fields like abstract, body, article-id and reference list.

The Query evaluation is done on the basis of description and not summary. According to them all the important information that would be in summary would also be available from description. So they have used description as their main query. Also they have kept term proximity of ordered or unordered window of length 5. Then they have used MetaMap as an mapping tool for the biomedical texts to the UMLS metathesaurus. The MetaMap breaks down the input

sentence into individual terms or pair of phrases and also provides synonyms and hyponyms for them. The MetaMap also categorizes every individual concept into one of 133 available semantic type. Semantic type is also used to weight the input phrase like low weight to terms like language or age, and high weight to sign or symptoms etc.

Then after doing query expansion using MetaMap they have used the MeSH to expand the concept with semantic type. They have been choosing the top 2 terms for any given concepts. These terms were then assigned a weight of $\frac{1}{3}$ of the original term to avoid query drift.

For retrieval they have used the article field restriction using only the abstract and body of the index. They have also used filter in which the documents which do not contain high weighted concepts were filtered out from the results.

They have also applied some post processing steps which are based on the three scores one is of Indri, the second is the test score and the third one is the treatment score.

The results retrieved by this methods were slightly lower than the corresponding median score. However for some queries it did better than the median. In the query level analysis they have mentioned that using too many filters might degrade the performance. And also the term weight assignment rule should be reconsidered and many factors should be needed to add and remove. Thus according to me they have done a good job in indexing and then retrieving the documents based on different.

The second paper that I referred is "Query Refinement: Negation Detection and Proximity Learning Georgetown at TREC 2014 Clinical Decision Support Track". They have used Lemur search engine to build an index from the collection using krovetz stemmer and stopwords removal. They have used Language modelling using Dirichlet smoothing which is the default retrieval algorithm of Lemur.

For the Medical Concept Detection they have used two main methods MeSH and MetaMap. Initially for processing the query terms MeSH on demand is used. So the query would be send to MeSH on Demand and it will return some related keywords which will be replaced as query terms. However a single medical term may have multiple meaning or terminology and so for that they have used MetaMap. MetaMap provides synonyms for the query terms with maximum weight of 1000. Only the terms with maximum weight were selected which would be added to the query. Also each list of MeSH terms and their synonyms were filtered by Inverse Document Frequency(IDF). The terms which has IDF less than 1.2 were considered very common terms and thus they were ignored.

Topic sub-classification and query expansion- For this approach they have calculated mean number of MeSH terms per query, ignoring the synonyms and if number of terms are less than one-half of standard deviation they were considered as hard topic. For this hard topics they

have used Stanford NLP parser to add nouns and these nouns were filtered by IDF and queried in MetaMap. For query which has average number of terms less than or equal to one standard deviation were given IDF threshold of 0.7 and for other a threshold of 1.1 was given. After giving this IDF a second layer of filtering was done using the blacklist and whitelist.

Query Learning Approach- For query learning approach they have noticed that the terms which are much common often create additional noise and so their weights needs to be reduced. Also for the terms which has their IDF less than threshold had their weights reduced in the final query. However, their runs which had implemented this methods performed worse than other ones. They state that this might be due to the limitation of static topic and not the approach.

Negation Detection- To reduce the probability of false positive they have reduced the weights of MeSH terms and their synonyms which came from negated phrases. They have explained this topic by stating that for a patient with 'abdominal pain and no fever' the MeSH would calculate synonyms of abdominal, pain and fever. However, fever has nothing to do with this query. So to reduce this type of errors they have used NegEX to detect negated phrase in each query.

Methods/My Approach

The first task for this project was to index the documents and for that Indri search engine have been used. Indri is a text search engine developed by UMASS and is a part of Lemur project. The index can be easily done in Indri as it supports XML documents. The fields which were included during indexing were abstract, body, article-id, article-title and reference list.

For Indexing the documents Indri has been used. Indri is a search engine tool and is a part of Lemur toolkit. It is easy to index data and run query in Indri. It can be done by simple command line arguments. So the indexing has been done on server named Mahurti. During Indexing, Krovetz stemmer has been used for morphological normalization. Also the fields included in indexing the documents are article-ID, article-title, abstract and body. We would be using abstract and body for the searching of the final query from the documents.

After the Index is being generated the main task is to do query generation and retrieval. The queries are present in two forms- description and summary. For this project I have choose to use summary as a part of query. Description are generally very long and also contains many useless information. This useless information will then affect the final query generation and so I have used summary as a part of query generation. On the other hand summary generally has all the information that are necessary for the query generation. Consider for instance the description of a sample topic:

"An 85-year-old man is brought to the ER because of gradual decrease in his level of consciousness. In the last 3 days he stopped walking and eating by himself. He has had no

fever, cough, rash or diarrhea. His daughter recalls that he had been involved in a car accident 3 weeks prior to his admission and had a normal head CT at that time.”

And the summary for the same topic is :

“85-year-old man who was in a car accident 3 weeks ago, now with 3 days of progressively decreasing level of consciousness and impaired ability to perform activities of daily living.”

So by analyzing the above two text I came to know that there are some terms in description which can affect the query generation. For instance in the description it is written that ‘he has had no fever’, but in the query generation ‘no’ would be removed as a stopword and so we will left with ‘fever’ which will be added to the query. Thus through these we will be less likely to generate articles which would be related to a non-smoker and get articles related to smoking habits.

On the other hand in summary we can get mostly all the keywords that are being generated from the description. And we can get rid of that negation terms which can affect our query generation. There are some disadvantages of using summary that it sometimes misses out some important information, but this is really dependent on the query.

So, Initially we can get summary for the query and then process it further. We would be following the sequence below:

1. Using MeSHOnDemand, identify MeSH Terms for Summary text.
2. Using MeSHOnDemand, identify entry terms for the MeSH terms
3. For terms identified in step 1,
 - i) if term (Term_i) is in blacklist, discard the term
 - ii) otherwise, use meshbrowser to find EntryTerms for each term. if entryterm is available:
 - a) generate indri synonym query {Term_i entryterm1 entryterm2 ...entrytermN}
 - else
generate indri query term Term_i as query term
4. combine the synonym queries generated by all the terms using #combine() method of indri
5. If the unique terms after processing step 3(i) is less than 5, then repeat steps 3-4 for the mesh terms extracted from description (Select only those terms which are not present in mesh summary terms)
6. Fire the query thus generated to the indri index.

For the first step, we would be getting the MeSH terms for the query. The query here would be the summary of the topic.xml file. By this step we would be getting the MeSH terms which will be added to the final query. MeSH will generally provide us with the synonyms or hyponyms of the terms. But also we need to process these terms further because these terms might not be enough for the query generation.

So for step2 we are also including the Entry terms for all the terms which we get from the first step. Thus this is one the important step in query expansion. As by this step we would be able to generate more relevant terms.

After the completion of first two steps of query generation in which we are adding terms to the query. Now, in this step we will be assigning the weights to the terms and their order. It is basically the order in which it it will be processed to generate a query file.

So first if any of the terms in the query would be in the black list[4] it will be removed from the list. After the first step it will be generating a query which would be able to use in Indri. For running a query in Indri, IndriRunQuery command is used and so it will give a list of relevant files.

Another important factor to consider is that searching the whole file makes no sense, the information that we are searching for would be available from the abstract and the body and body of the file.

I also wanted to use the MetaMap tool for further processing of the query but unfortunately I did not get an access of that tool. We need to get permission from UMLS(Unified Medical Language System). However I would like to implement this tool further for my project as it has shown some great improvements in the results of the query and also the documents retrieved are much more relevant. MetaMap is tool used for mapping the query terms to UMLS metathesaurus. It is great tool which breaks the query into phrases or individual terms and provides synonyms or hyponyms where possible. But as I did not get the access to MetaMap as an alternative I have used the MeshOnDemand.

The final query can be generated using the file Indri_query_generation.py and the query would be as follow:

```
#combine(back {#1(chest pain) #1(Precordial Catch) #1(Precordial Catch Syndrome)
#1(Texidors Twinge)} {exercise #1(Acute Exercise) #1(Aerobic Exercise) #1(Exercise
Training) #uw2(Exercise Aerobic) #uw2(Exercise Isometric) #uw2(Exercise Physical)
#1(Isometric Exercise) #1(Physical Activity)} {hypertension #uw2(#1(Blood Pressure) High) }
obesity)|
```

Query Example

This is a sample query for the Topic no1. Similarly we can calculate query for all the topics from topic.xml file. Also we can generate query for our custom query

Results

The below are the result of the query generated by doing the query processing as mentioned above. For running the query in Indri, IndriRunQuery is used on the terminal. There are only ten

results per topic displayed here but we can change that by the command='-count=10' to any number of counts we want.

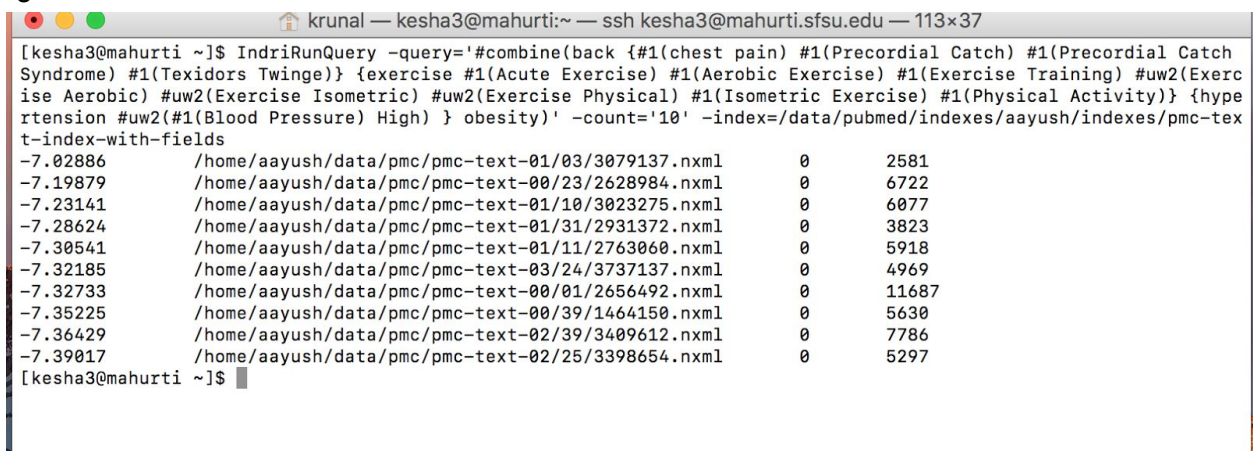
When we run the query which is shown in the above figure in the IndriRunQuery we would be able to generate a list of files as follow:

topic 1

-7.02886	/home/aayush/data/pmc/pmc-text-01/03/3079137.nxml	0	2581 R
-7.19879	/home/aayush/data/pmc/pmc-text-00/23/2628984.nxml	0	6722
-7.23141	/home/aayush/data/pmc/pmc-text-01/10/3023275.nxml	0	6077
-7.28624	/home/aayush/data/pmc/pmc-text-01/31/2931372.nxml	0	3823
-7.30541	/home/aayush/data/pmc/pmc-text-01/11/2763060.nxml	0	5918
-7.32185	/home/aayush/data/pmc/pmc-text-03/24/3737137.nxml	0	4969 R
-7.32733	/home/aayush/data/pmc/pmc-text-00/01/2656492.nxml	0	11687
-7.35225	/home/aayush/data/pmc/pmc-text-00/39/1464150.nxml	0	5630
-7.36429	/home/aayush/data/pmc/pmc-text-02/39/3409612.nxml	0	7786
-7.39017	/home/aayush/data/pmc/pmc-text-02/25/3398654.nxml	0	5297

We can display as many results as we want by using the '-count=10' in IndriRunQuery. By individually analyzing the files and marking relevant files as R at the end we are able to calculate precision at 10 for each individual topic and get an average p@10 for all the files. The average p@10 is 0.14

All the generated query are stored in Query_generated.xml and in that under <query></query>. So for running the query just copy the query from the file and run it. This can be shown as in the figure below.



```
krunal — kesh3@mahurti:~ — ssh kesh3@mahurti.sfsu.edu — 113x37
[kesh3@mahurti ~]$ IndriRunQuery -query='#combine(back {#1(chest pain) #1(Precordial Catch) #1(Precordial Catch Syndrome) #1(Texidors Twinge)} {exercise #1(Acute Exercise) #1(Aerobic Exercise) #1(Exercise Training) #uw2(Exercise Aerobic) #uw2(Exercise Isometric) #uw2(Exercise Physical) #1(Isometric Exercise) #1(Physical Activity)} {hypertension #uw2(#1(Blood Pressure) High) } obesity)' -count='10' -index=/data/pubmed/indexes/aayush/indexes/pmc-text-index-with-fields
-7.02886      /home/aayush/data/pmc/pmc-text-01/03/3079137.nxml      0      2581
-7.19879      /home/aayush/data/pmc/pmc-text-00/23/2628984.nxml      0      6722
-7.23141      /home/aayush/data/pmc/pmc-text-01/10/3023275.nxml      0      6077
-7.28624      /home/aayush/data/pmc/pmc-text-01/31/2931372.nxml      0      3823
-7.30541      /home/aayush/data/pmc/pmc-text-01/11/2763060.nxml      0      5918
-7.32185      /home/aayush/data/pmc/pmc-text-03/24/3737137.nxml      0      4969
-7.32733      /home/aayush/data/pmc/pmc-text-00/01/2656492.nxml      0      11687
-7.35225      /home/aayush/data/pmc/pmc-text-00/39/1464150.nxml      0      5630
-7.36429      /home/aayush/data/pmc/pmc-text-02/39/3409612.nxml      0      7786
-7.39017      /home/aayush/data/pmc/pmc-text-02/25/3398654.nxml      0      5297
[kesh3@mahurti ~]$
```

Running the query of topic 1

Analysis and Future Work

There can be many improvements be made in the query generation and query expansion. By analyzing the documents that have been retrieved in the top results we can see that there are many documents which are still not relevant to the query. There are very few documents relevant to the query. So I think I need to do some further major improvements in the query generation to retrieve relevant documents.

I am able to notice that there are certain documents that are retrieved based on some terms in the query which is of not much importance. For instance, for the query 'Chest pain and no fever' there are some documents related to fever, which should not be there. So for improving these types of result I need to do query weighting. In query weighting the most important terms are given the highest weight compared to some other terms in the query. And the terms which are more common or occurs several time should be given very less weight.

For future work what I think I can implement is weighting of the terms in the query. By doing this I would be able to weight query term which are more relevant more and also to do different weighting of terms according to their importance. Another important implementation would be using the MetaMap tool. MetaMap is highly effective tool used for medical terms. So I think by combining the MetaMap and weighting of terms would give a very good changes on my project. Also removal of the unnce

Reference

1. San Francisco State University at TREC 2014: Clinical Decision Support Track and Microblog Track
2. Query Refinement: Negation Detection and Proximity Learning Georgetown at TREC 2014 Clinical Decision Support Track
3. MeSH on Demand - <https://www.nlm.nih.gov/mesh/MeSHonDemand.html>
4. <https://metamap.nlm.nih.gov>
5. <https://meshb.nlm.nih.gov>