Technologically Assisted Reviews in Empirical Medicine

Illinois Institute of Technology

Abstract:

Finding all relevant documents out of large collection is an essential process in evidence based medicine. Evidence-based medicine has become an important strategy in health care and policy making. In order to practice evidence-based medicine, it is important to have a clear overview over the current scientific consensus. These overviews are provided in systematic review articles, that summarize all evidence that is published regarding a certain topic (e.g., a treatment or diagnostic test). In order to write a systematic review, researchers have to conduct a search that will retrieve all the documents that are relevant. This is a difficult task, known in the Information Retrieval (IR) domain as the total recall problem. With medical libraries expanding rapidly, the need for automation in this process becomes of utmost importance.

General Setup/ Approach

The library:

The PubMed Central library was used. PubMed is a search engine accessing millions of biomedical citations. This library was chosen because the full text of all articles is publicly available. A large part of PubMed Central (1,227,716 out of 1,317,348 articles) is also published in the important MEDLINE library. All documents were downloaded, and the meta fields (title, publish date, keywords), body, and abstract were extracted from the raw XML.

The search Engine:

The Elasticsearch 5.2.2 engine was used to index and search through the documents.

Measure

A search returns a (ranked) list of documents

- **Scoring work in Elasticsearch**:
  score(q,d) =
  queryNorm(q)
  * coord(q,d)
  * SUM (
  tf(t in d),
  idf(t)²,
  t.getBoost(),
  norm(t,d)
) (t in q)
- **Recall**: Recall expresses the proportion of documents that are correctly retrieved. This measure is also known as sensitivity in the systematic review domain. It is defined by recall = # relevant documents retrieved/ # all relevant documents.
- **Precision**: Precision expresses the proportion of the retrieved documents that are correct. It is defined by precision = # relevant documents retrieved /# all documents retrieved.

- **F1**: F1 defines the geometric mean between recall and precision. It is defined as F1 = 2 · recall · precision /recall + precision.

- **F**-β: In areas like systematic reviewing, recall may be more important than precision. The Fβ measure allows to put more weight on one of the two. It is defined so that a β value of 10 means that recall is 10 times as important as precision. It is defined by: $F\beta = (1 + \beta^2) * (recall * precision) / (recall + (\beta^2 \cdot precision))$

- **MAP:** The measures above define the performance of the search engine at a specific rank. As a result, each query has as many precision-, or Fβ-values as the number of documents that it retrieves. This makes it difficult to combine the performance for different queries. The average precision is a measure of performance over the entire ranklist. It equals to the area under the precisionrecall plot. The average precision can in turn be averaged over queries resulting in the Mean Average Precision (MAP).

DATA

Data Source:

We are going to used CLEF eHealth 2017 development dataset. The development set will consist of 20 topics for Diagnostic Test Accuracy (DTA) reviews. the development data in a text format which will include the aforementioned information, and two sets of qrels, (a) a qrel at the abstract level dictating which PubMed IDS (PIDs) were excluded after the abstract screening and which ones were further processed, and (b) a qrel at the document level dictating which PubMed IDS (PIDs) were excluded after the abstract or/and document screening and which ones were included in the review written.

Following table represent Topic Id and number of PIDs for that id

TOPIC ID	No. of PIDS	TOPIC ID	No. of PIDS
CD010438	3249	**CD011975**	8227
CD007427	1469	**CD009323**	3857
CD009593	15076	**CD009020**	1576
CD011549	12704	**CD011548**	12706
CD011134	1952	**CD011984**	8221
CD010409	43484	**CD007394**	2542
CD010771	316	**CD009944**	1225
CD009591	8082	**CD008643**	15078
CD008691	1322	**CD008686**	3963
CD010632	1508	**CD00804**	3148

Data Format:

CLEF data are in text format which contains different field like Topic, Title, Query and PIDS. The title of the review, written by Cochrane experts. The Boolean query manually constructed by Cochrane experts.

The set of PubMED Document Identifiers (PID's) returned by running the query in MEDLINE.

According to TREC the format of a qrels file is as follows: TOPIC ITERATION DOCUMENT# RELEVANCY where TOPIC is the topic number, ITERATION in our case is a dummy field always zero and not used, DOCUMENT# is the PubMed document identification number (PID), and RELEVANCY is a binary code of 0 for not relevant and 1 for relevant. The order of documents in a qrels file is not indicative of relevance or degree of relevance. Only a binary indication of relevant (1) or non-relevant (0) is given. Documents not occurring in the qrels file

were not judged by the human assessor and are assumed to be irrelevant in the evaluations.

Programming Language: Python 3

Python Packages:

- PubMed
- NLTK
- ElasticSearch
- Pandas
- NumPy
- Matplotlib

EXPERIMENTS

- Data Cleaning/ Preprocessing
- Fetch Data
- Load Data
- Rank Documents
- Query Expansion
- Data Splitting
- Search
- Generate CSV Files
- Clustering

IMPLIMENTATION

1.Data Cleaning/ Preprocessing:

All datafiles are in txt format which contains TOPIC, TITLE, QUERY and PIDS. We need to separate all this field for processing all PIDS which fetch all details related to TOPIC.

Preprocessing steps:

- Replace "\n" by ""
- Perform Tokenization for removing all numbers and non-alphanumeric character like punctuation by using regular expression "\d+" and "\W+"
- Perform Normalization for removing English stopwords. We are using Natural Language Toolkit for removing English stopwords.

2.Fetch Data:

We are using PubMed library for fetching data corresponding to PIDS. We parse PIDS as input to PubMed library. All documents were downloaded, and the meta fields (title, publish date, keywords), body, and abstract were extracted from the raw XML.

3.Load Data:

The Elasticsearch 5.2.2 engine was used as server to store, index and search through the documents.

Load Data Steps:

- Start ElasticSearch Server.
- Create index for storing data.
- Stored fetch data. We are storing only relevant require data by fetching it from XML

Fetch data and load data are continuous processes. We face different problems like stack overflow, connection timeout during this experiment. Because of this, all data are not stored on the server. To overcome this problem, we implement the buffer concept in which we keep the track of PIDS which are not loaded on the server. This process is continuing until PIDS buffer empty.

4. Rank Documents:

We are using ElasticSearch score function for ranking the documents. We parse query as parameter for ranking each document based on ElasticSearch score. ElasticSearch uses Lucene's practical scoring function which is similarity model based on Term Frequency (tf) and Inverse Document Frequency (idf) that also uses the Vector Space Model (vsm) for multi-term queries. We implement

Boolean search algorithm but we process with Elasticsearch for query search because the Query DSL (Domain Specific Language) in Elasticsearch is included in the Search APIs and provides a robust, flexible interface to query.

score(q,d) = queryNorm(q) * coord(q,d) * SUM (tf(t in d), idf(t)², t.getBoost(), norm(t,d)) (t in q)

- score(q,d) is the relevance score of document d for query q.
- queryNorm(q) is the query normalization factor.
- coord(q,d) is the coordination factor.
- The sum of the weights for each term t in the query q for document d.
  - tf(t in d) is the term frequency for term t in document d.
  - idf(t) is the inverse document frequency for term t.
  - t.getBoost() is the boost that has been applied to the query.
  - norm(t,d) is the field-length norm, combined with the index-time field-level boost, if any.

5. Query Expansion:

For query expansion, we are using '*most_field*' type which has the following functionality for searching:

• **Stemmer**: for example, use a stemmer to index jumps, jumping, and jumped as their root form: jump. Then it doesn't matter if the user searches for jumped; we could still match documents containing jumping.

• **Synonyms:** for example, include synonyms like jump, leap, and hop.

• **Remove diacritics, or accents**: for example, ésta, está, and esta would all be indexed without accents as esta.

We used 'or' operator for searching long query. We also used "minimum_should_match" parameter, which allows you to specify the number of terms that must match for a document to be considered relevant.
MeSH terms (a popular medical ontology) were omitted from the queries.
More parameters are explained in Appendix A.

6. Data Splitting:

Split the "qrels" file which contains filed like TOPIC ITERATION DOCUMENT# RELEVANCY. "qrels" file split with respect to TOPIC and after that split with respect to RELEVANCY. All these files are store in .csv format for further classification.

7. Search:

• **Search using TITLE:** We performed a search by TITLE as the title contains few words and is supposed to represent the most essential keywords. This search is performed on different part of the database like title and abstract. We performed a search only on the title, only on abstract and on both title-abstract. As a result, we get ranked document for different search.

• **Search using QUERY:** We performed a search by QUERY. This search is performed on different part of the database like title and abstract. We performed a search only on the title, only on abstract and on both title-abstract. As a result, we get ranked document for different search

8. Generate CSV File:

Rank documents and qrels file split data are stored in csv format for further analysis.

9. Clustering:

We performed following clustering algorithm for filtering data which are more relevant.

- K-Mean
- Mean Shift

10. Recall/Precision/F1-Score/ F-β/MAP Calculation:

Following tables represent result for different measures

Topic ID	Recall	Precision	F1 Score
CD011975	0.8497	0.1346	0.2325
CD011984	0.7202	0.10061	0.17656
CD009944	0.8119	0.0986	0.175926

Search by Topic TITLE on PIDS title field

Topic ID	Recall	Precision	F1 Score
CD011975	0.8691	0.137772	0.237843
CD011984	0.8127	0.113538	0.199244
CD009944	0.8205	0.099688	0.177778

Search by Topic TITLE on PIDS abstract field

Topic ID	Recall	Precision	F1 Score
CD011975	0.9499	0.150576	0.25994
CD011984	0.9052	0.126462	0.22192
CD009944	0.8376	0.101765	0.181481

Search by Topic TITLE on full text

Topic ID	Recall	Precision	F1 Score
CD011975	0.9483	0.081562	0.150205
CD011984	0.9493	0.059928	0.112739
CD009944	0.8290	0.093901	0.168696

Search by Topic QUERY on PIDS title field

Topic ID	Recall	Precision	F1 Score
CD011975	0.8901	0.07656	0.140993
CD011984	0.8854	0.05589	0.105153
CD009944	0.8205	0.092933	0.166957

Search by Topic QUERY on PIDS abstract field

Topic ID	Recall	Precision	F1 Score
CD011975	0.9709	0.083507	0.153787
CD011984	0.9691	0.061179	0.115093
CD009944	0.8376	0.094869	0.170435

Search by Topic QUERY on full text

	MAP
Full Text Search by TITLE	0.0221
Abstract Search by TITLE	0.0167
Title Search by TITLE	0.0271
Full Text Search by Query	0.0211
Abstract Search by Query	0.0168
Title Search by Query	0.0274

MAP

	F-β
Full Text Search by TITLE	0.60205
Abstract Search by TITLE	0.625825
Title Search by TITLE	0.670443
Full Text Search by Query	0.649437

Abstract Search by Query	0.607194
Title Search by Query	0.596124

Average F-β

ANALYSIS

I. Document Ranking

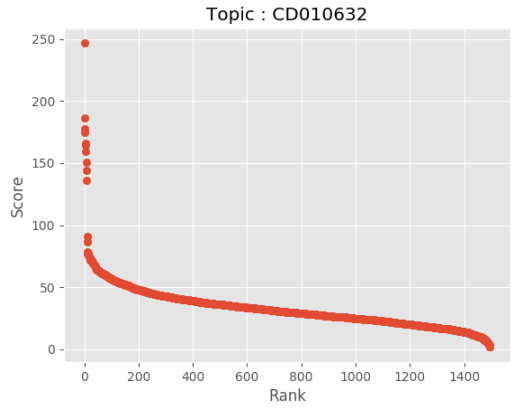

Fig. 1 Document Ranking

Fig. 1 represents document ranking with score for TOPIC: CD010632.
As we observed that document score decreases with increase in rank.
Rank 1 document have highest score while end rank have lowest score.

II. Clustering

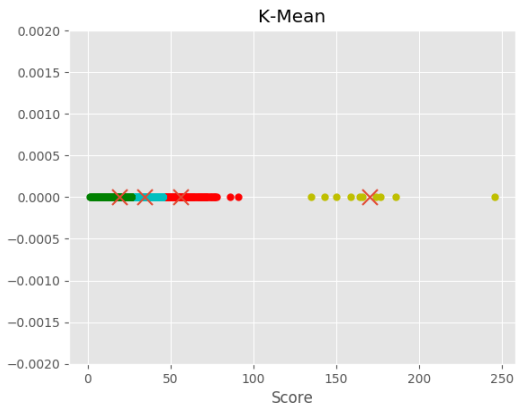

Fig. 2 K-Mean Clustering

Fig. 2 represents clustering classification of score of TOPIC: CD010632 using K-Mean algorithm.
We observed that clustering with yellow color have highest score i.e that documents are more relevant than others.

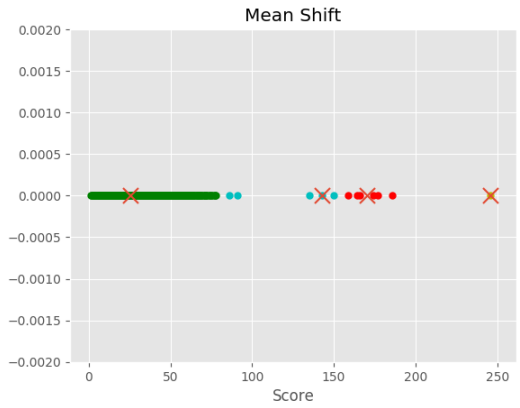

Fig. 3 Mean Shift Clustering

Fig. 3 represents clustering classification of score of TOPIC: CD010632 using Mean Shift algorithm.
We observed that clustering with yellow and red color have highest score compare to other i.e that documents are more relevant than other.

III. Search by TITLE

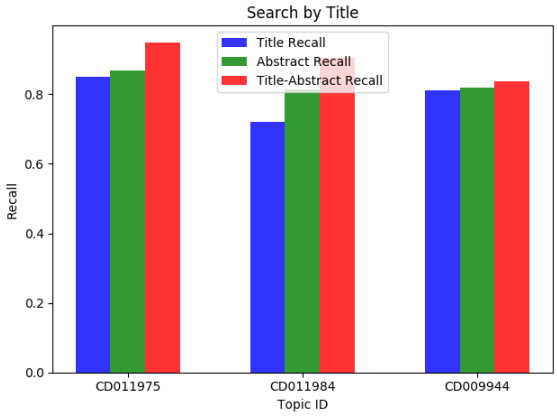

Fig. 4 Search by TITLE Performance

Fig. 4 shows the performance for search using Topic TITLE on PIDS title, abstract and full text (title-abstract).

As expected searching on full text improves recall as compared with search on title and abstract.

For topic id CD011975 we observed that recall on title field is 0.84, on abstract field is 0.86 while on full text recall is 0.94.

IV. Search by Query

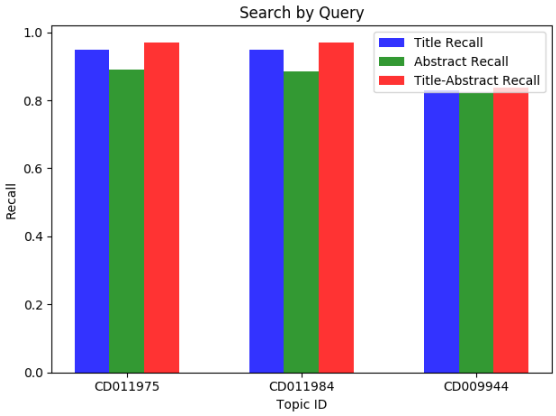

Fig. 5 Search by Query

Fig. 5 shows the performance for search using Topic QUERY on PIDS title, abstract and full text (title-abstract).

As expected searching on full text improves recall as compared with search on title and abstract.

For topic id CD011975 we observed that recall on title field is 0.94, on abstract field is 0.89 while on full text recall is 0.97.

V. Search Performance Comparison

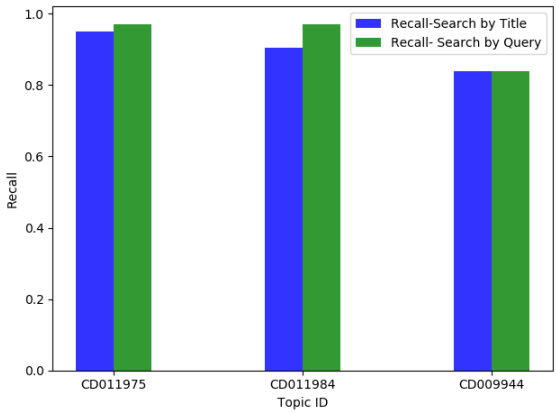

Fig. 6 Recall Comparison

Fig. 6 represent result of recall after performing search using Topic TITLE and QUERY.

As a result, we observed that performance is improve after search using QUERY.

For topic id CS011975 we observed that recall for search using TITLE is 0.94 while search using QUERY is 0.97.

VI. Recall and Precision

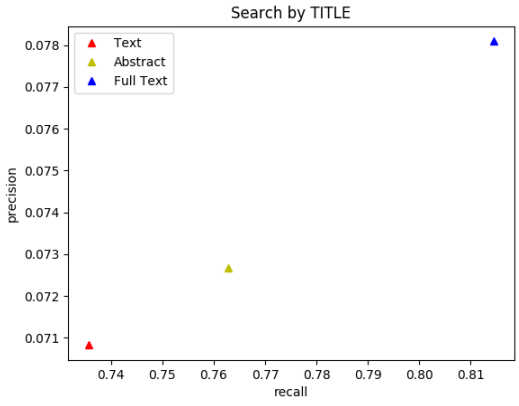

Fig. 7 Average recall and precision

Fig. 7 represents average recall and precision for all query after performing search using TITLE.

As a result, we observed that, performance is improved for full text as compare to title and abstract.

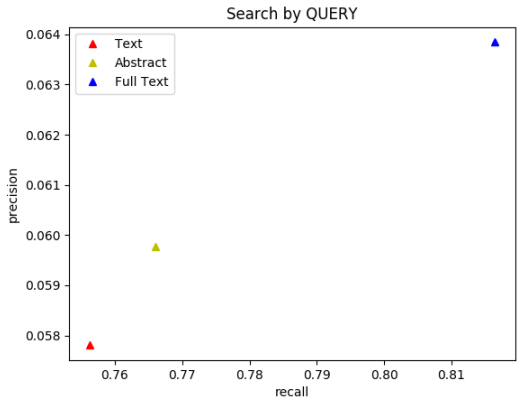

Fig. 8 Average recall and precision

Fig. 8 represents average recall and precision for all query after performing search using QUERY.

As a result, we observed that, performance is improved for full text as compare to title and abstract.

VII. All Measures

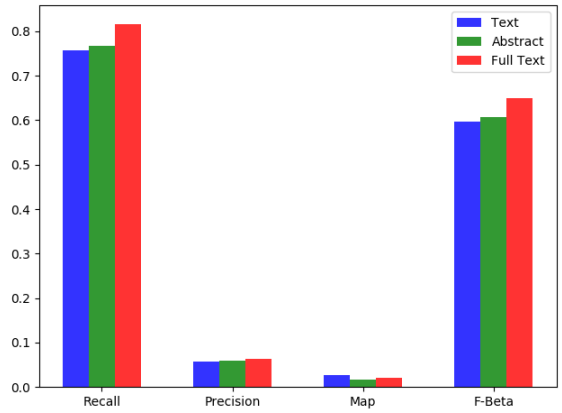

Fig. 9 All measures for Performance Evaluation

Fig. 9 represents comparison of all measures like recall, precision, MAP and F-β score on text, abstract and full text.

As a result, we observed that performance is improved for full text for all measures.

Summary:

Searching for full text improves recall as compared with search on text and abstract only. However, cost of this decrease precision, meaning that more documents have to be examined in order to find the same number of relevant documents. The MAP is also lower for Full Text search.

Search on title result in almost total recall. However, this is misleading result, because almost all the documents in the database were retrieved by this search. This is reflected by the extremely low precision as shown in Fig. 7 and Fig. 8. The low value of MAP and F-β confirm that search on title is not effective.

ERROR ANALYSIS

1. **Clustering**

   We performed clustering to get more relevant PIDS. But we get difficulty to find the best number of clusters as some cluster contains only few PIDS which have more score while some cluster contains more than 50% of PIDS which have less score.

   Fig. 3 shows clustering classification using Mean shift. But it classified only one PIDS which have more score while more than 50% of PIDS are classified in one cluster which have low score.

2. **Query Expansion**

   We used “minimum_should_match” for pruning some documents which are less important or contain only few information. For implementation, we manually passed some value which is useful for

some documents but not all. For some documents, it does not generate any output. So, for final implementation we skipped this parameter from search query.

CONCLUSION

- Finding relevant documents from large datasets.
- Searching on Full Text improve Recall.
- Performance improve after performing search using QUERY.

APPENDIX A:

Parameter	Description
stopword	All stopword like 'a','an','the' etc. are ignore.
multi_match	The multi_match query builds on the match query to allow multi-field queries:
type	The most_fields type is most useful when querying multiple fields that contain the same text analyzed in different ways. For instance, the main field may contain synonyms, stemming and terms without diacritics. A second field may contain the original terms, and a third field might contain shingles. By combining scores from all three fields we can match as many documents as possible with the main field, but use the second and third fields to push the most similar results to the top of the list.
fields	It is often useful to index the same field in different ways for different purposes. This is the purpose of *multi-fields*.
operator	'OR' operator used for searching long query.
size	Maximum size defines for document retrieval.