University of New Hampshire

CS953

# Team1 Prototype1

*Author:*
Amith Ramanagar Chandrashekar

*Author:*
Pooja Himanshu Oza

*Author:*
Rachel E Cates

*Author:*
Ahmed M Alnazer

*Supervisor:*
Dr. Laura Dietz

February 21, 2019

## Contents

## 1 Problem Definition

There is an ever increasing demand of providing more efficient and effective ways of Knowledge Discovery to the users. With the explosion of the data in last decade and evolution in the information needs of the user, retrieving relevant information to meet the user demands is becoming a greater challenge. Our system is trying to find the solution to meet this challenge of retrieving complex information needs effectively. The motivation of the system is to retrieve relevant information from the collection, combine the retrieved information and present it to the user.

## 2 Approaches

### 2.1 Document-Similarity Re-Rank

Inspired by cluster hypothesis which states that document in the same cluster will behave similarly or has similar characteristics.The method is to re-rank the candidate set generated by BM25 using the document similarity. The document vector(Biased Vector) is created by using Top K retrieved documents,taking each terms of the top k document, use their word embedding vector to compute the document representation. Using the computed document representation, calculate the cosine similarity with all the documents. The score for each document is calculated as follows

$$document.score * cosine.score + biased.score \tag{1}$$

Biased score is the mean of Top k documents score

$$biased.score = \frac{\sum_{1}^{k} doc.score(k)}{k}$$

**Command Line Argument**

```
search -i D:\Data_sci\indexed_file
-q D:\Data_sci\benchmarkY1-train\train.pages.cbor-outlines.cbor
-we D:\glove.6B\glove.6B.300d.txt -dim=300 -k=300 --bias-fact=20 --rerank
```

#### 2.1.1 Document-Similarity Re-Rank IDF

As described above, here the terms from the Top k document is selected based on the top IDF terms (Inverse Document Frequency). Used half of the high IDF terms from the documents to build the document vector representation and performed similar calculation.

**Command Line Argument**

```
search -i D:\Data_sci\indexed_file
-q D:\Data_sci\benchmarkY1-train\train.pages.cbor-outlines.cbor
-we D:\glove.6B\glove.6B.300d.txt -dim=300 -k=300 --bias-fact=20 --rerank-idf
```

#### 2.1.2 Document-Similarity Re-Rank DF

As described above, here the terms from the Top k document is selected based on the top DF terms (Document Frequency). Used half of the high IDF terms from the documents to build the document vector representation and performed similar calculation.

**Command Line Argument**

```
search -i D:\Data_sci\indexed_file
-q D:\Data_sci\benchmarkY1-train\train.pages.cbor-outlines.cbor
-we D:\glove.6B\glove.6B.300d.txt -dim=300 -k=300 --bias-fact=20 --rerank-df
```

## 2.2 Spam Detection

In the first phase of spam detection, qrels annotated with quality rankings are parsed by the IndexHamSpamRunner. Those with annotations of 2 or -2 are extracted, and their paragraph ids are checked against those in the Lucene index. Ids found in the index are used to create the training and test data for the spam classifiers in phases two.

In phase two, the FilterRunner gets the tokens from the training and test data, and hands them off to a series of naive Bayes predictor classes. There are several variations on the base LabelPredictor class, including unigrams, bigrams, trigrams, and quadgrams. Each classifier is trained on the data from the training set. Then, during the test phase, ham and spam scores for the test set are returned.

In future iterations of this project, I will make the creation and access of the training and test data more efficient. Currently, I iterate through the entire Lucene index, which is very inefficient. Nevertheless, the above approaches worked on a subset af the Lucene index, and provide proof of concept.

## 2.3 Entity Popularity

This approach is a combination of Entity Popularity and graph based methods to identify the most important entity for a particular query from the passages retrieved using BM25.

#### 2.3.1 Entity Degree

In this approach, the first level outlinks relations between entities is taken into consideration, where each entity is taken as node and edge is created if the other entities are connected to the entity through outlinks. Using the passages retrieved from BM25, a graph for each query is constructed based on the entities. The degree of each entity node is calculated and taken as weight for calculating the score. The entities are ranked based on the score and accordingly the passages are re-ranked.

```
final score = BM25 paragraph score * degree of the entity node

search --entity-degree
-i /home/poojaoza/Documents/projects/test200/paragraph.lucene
-q /home/poojaoza/Documents/projects/train.pages.cbor-outlines.cbor
--entity-index /home/poojaoza/Documents/projects/test200/entity.lucene

Entity Degree:

For Article Qrels:
    MAP: 0.0875
    R-Prec: 0.1466
For Hierarchical Qrels:
    MAP: 0.0095
    R-Prec: 0.008
```

### 2.3.2 Entity Similarity

In the second approach, each entity is taken as node and the connection between the entities is established based on the similarity of the abstract text of each entity using Glove embedding. The value 0.5 is taken as threshold to decide whether the edge is connected between two entities. The edge weight is the similarity score between the abstract text of two entities. The edge weight is taken as weight in calculating the score and the passages are re-ranked.

```
final score = BM25 paragraph score + cosine similarity

search --entity-sim
-i /home/poojaoza/Documents/projects/test200/paragraph.lucene
-q /home/poojaoza/Documents/projects/train.pages.cbor-outlines.cbor
--entity-index /home/poojaoza/Documents/projects/test200/entity.lucene
-we /home/poojaoza/Downloads/glove.6B/glove.6B.50d.txt -dim 50

Entity Similarity:

For Article Qrels:
    MAP: 0.125
    R-Prec: 0.1997
For Hierarchical Qrels:
    MAP: 0.0088
    R-Prec: 0.0059
```

Note: This approach is taking around 1 hour for calculating the similarity score for 50 dimensions in train data as well as test data. The reason being the brute-force approach taken of comparing entity against every other entity found in the passages retrieved from BM25.

### 2.3.3 Query Expansion using Entity Degree Ranking

In the third approach, the query is expanded with the first entity of highest degree retrieved based on the entity degree approach. BM25 is run on the expanded query.

```
search --entity-expand
-i /home/poojaoza/Documents/projects/test200/paragraph.lucene
-q /home/poojaoza/Documents/projects/train.pages.cbor-outlines.cbor
--entity-index /home/poojaoza/Documents/projects/test200/entity.lucene

Expanded BM25:

For Article Qrels:
    MAP: 0.124
    R-Prec: 0.1999
For Hierarchical Qrels:
    MAP: 0.0088
    R-Prec: 0.0059
```

### 2.3.4 Evaluation Results

| Method | Article MAP | Article RPRec | Hierarchical MAP | Hierarchical RPrec |
|---|---|---|---|---|
| Expanded BM25 | 0.124 | 0.1999 | 0.0088 | 0.0059 |
| Entity Degree | 0.0875 | 0.1466 | 0.0095 | 0.008 |
| Entity Similarity | 0.125 | 0.1997 | 0.0088 | 0.0059 |
| BM25 | 0.125 | 0.1997 | 0.0088 | 0.0059 |

## 2.4 Query Expansion using Top n Entity

### 2.4.1 Summary

It get all the entity in the document and count how many time it appear Then get top n entity then has the highest number (specify by the command line -top) The expansion basically append all the selected entity text to the query text

### 2.4.2 Command

```
search --query-expansion Or -qe
    -i   "indexed_file"
    -q   "training cbor file"
```

```
        -top "number of top entity used in the expansion default 3"

example:
    search -qe
        -i /home/team1/indexed_file
        -q /home/ama1003/DataScienceCS953/Prototype1/train.pages.cbor-outlines.cbor
        -top 3
```

#### 2.4.3 Validation

If the top number is less than 0 show message ”Please pass the number of entity to be returned” and exit Default 3

#### 2.4.4 Result

After running successfully the result saved in ”result” folder with name ”output-doc-reranking-With-QueryExpansion-k100-top3-ranking.txt”

#### 2.4.5 Extra

The firmware allow display some result to print out on the stdout Usage add to command ”-V”,”–verbose”

## 3 Evaluation

### 3.1 Data set

We use the TREC-CAR v2.0 for data set.

### 3.2 Corpus

ParagraphCorpus.v2.0 is used as corpus for Paragraph Index. UnprocessedAll-butBenchmark.v2.1 is used as corpus for indexing Entities.

### 3.3 Training

benchmarkY1-train.v2.0 is used as the training data set.

### 3.4 Test

benchmarkY1-test is used as test data set

### 3.5 Queries

We use benchmarkY1-test outlines for article queries.

### 3.6 Ground Truth

We use benchmark-y1-test article level qrels and hierarchical qrels as the ground truth for article level queries and hierarchical qrels respectively.

### 3.7 Evaluation Measure

We use the following evaluation metrics:

- Mean Average Precision (MAP)
- Precision at R (p@R)

## 4 Paired T-test Results

### 4.1 Entity Popularity Approaches

| Run | Mean/Stderr | |
|---|---|---|
| outputrankingBM25-perq.txt | 0.124995726 | 0.012814888 |
| outputrankingentityDegreeperq.txt | 0.087487179 | 0.009526593 |
| outputrankingexpandedBM25perq.txt | 0.123988034 | 0.012641695 |
| outputrankingentitySim50perq.txt | 0.124995726 | 0.012814888 |

### 4.2 Document similarity Approaches

| Run | Mean/Stderr | |
|---|---|---|
| `doc_sim_DF_reranking_k300_b20_d300_ranking` | 0.311951140845 | 0.755573612489 |
| `doc_sim_DF_reranking_k300_b20_d200_ranking` | -0.375459606408 | 0.70792593118 |
| `doc_sim_DF_reranking_k300_b20_d50_ranking` | 0.984799063196 | 0.326538159113 |
| `doc_sim_reranking_k300_b20_d100_ranking` | 1.85256124366 | 0.0661958252345 |
| `doc_sim_reranking_k300_b20_d200_ranking` | 1.1555070907 | 0.249986032125 |
| `doc_sim_reranking_k300_b20_d50_ranking` | 2.42971575084 | 0.0164657043074 |
| `doc_sim_reranking_k300_b20_d300_ranking` | 2.37558886765 | 0.0189700727032 |
| `doc_sim_IDF_reranking_k300_b20_d50_ranking` | 2.52117876152 | 0.0128954690563 |
| `doc_sim_IDF_reranking_k300_b20_d100_ranking` | 2.50540950913 | 0.0134566715269 |
| `doc_sim_IDF_reranking_k300_b20_d300_ranking` | 3.17158560158 | 0.00188860801165 |
| `doc_sim_IDF_reranking_k300_b20_d200_ranking` | 2.46218855882 | 0.0151083539741 |

## 5 Results

### 5.1 Entity Popularity Approaches

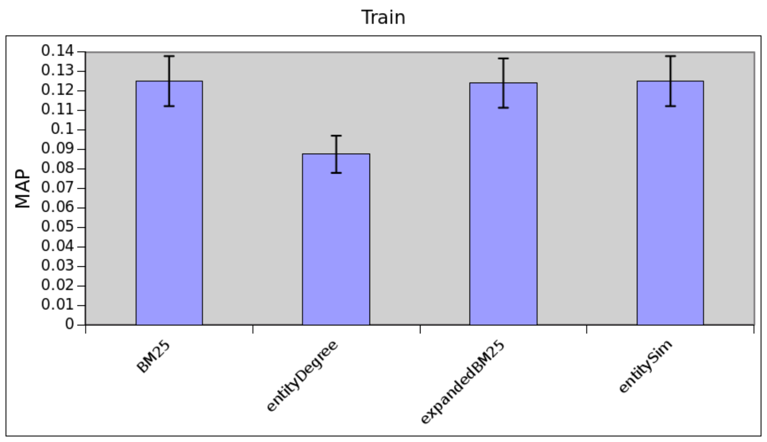

Figure 1: Train MAP Result for Entity Popularity Approaches

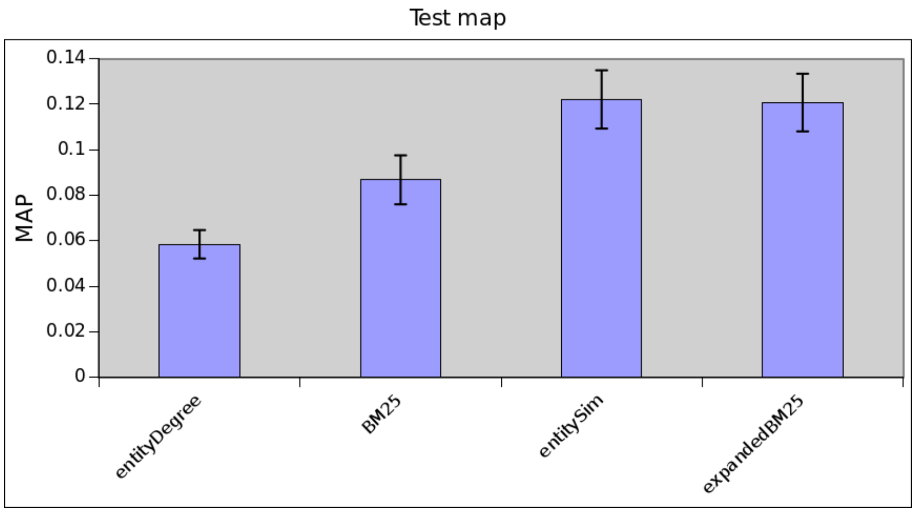

Figure 2: Test MAP Result for Entity Popularity Approaches

### 5.2 Document-Similarity Ranking

Results for different Document-similarity model. Here K indicates the initial candidate set retrieval,b indicates the number of top documents used to represent a vector and d indicates the dimension of the word embedding file used.

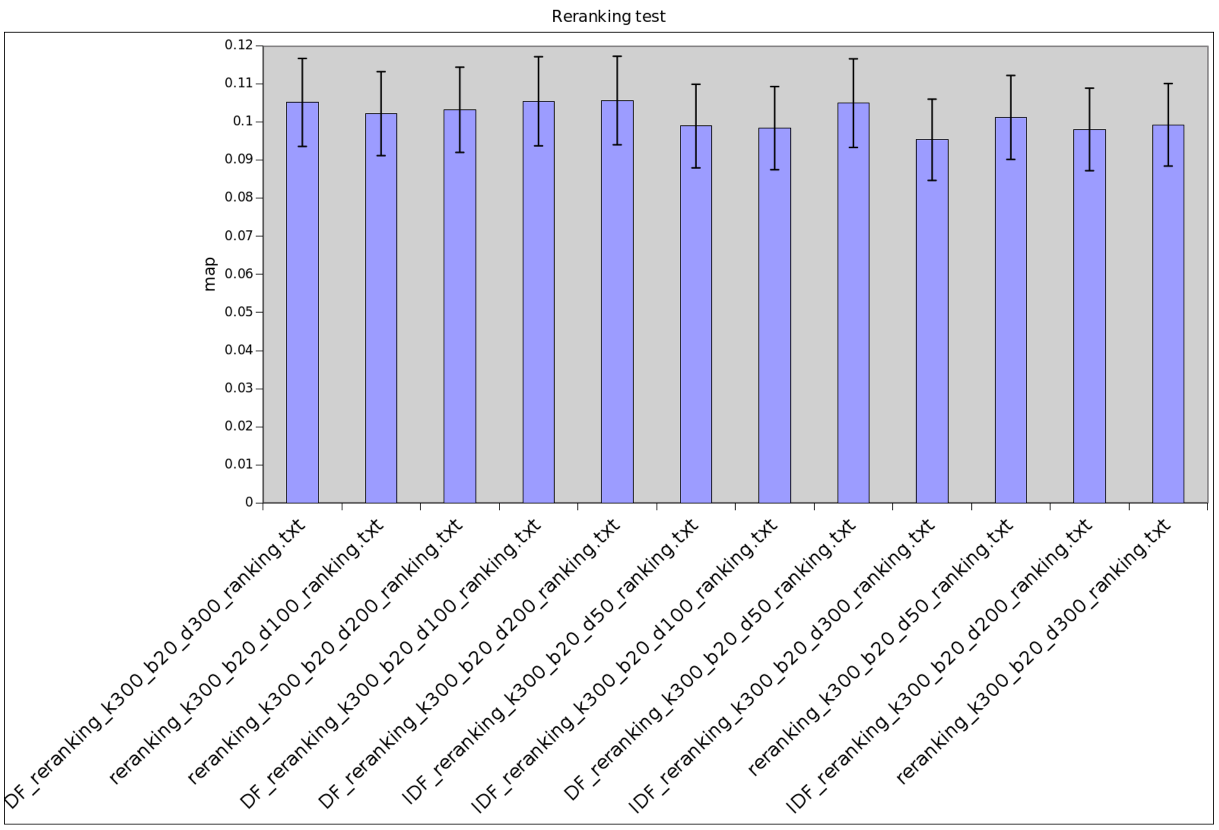

Figure 3: Test MAP Result for Document-similarity

## 6 Conclusion

Based on the observations from the results of Entity Popularity approaches and Document Similarity approaches, it can be concluded that using document similarity approach of using vector representation and combining that with entities improves the result significantly. Further exploration on approach of using entities abstracts to represent a document looks promising. Other venues such as using textual context or using document entity dual embedding can also be explored.

## 7 Contributions

- **Rachel:** Implemented the IndexHamSpamRunner, FilterRunner, LabelPredictor, BayesCounter, and NaiveBayesPredictor with variations. Helped update the install.sh script.

- **Amith:** Implemented Document-Similarity, Document-Similarity-IDF, Document-Similarity-DF, Jcommander (Command line argument), BaseBM25 searcher, contributed in Utils, Install script and java based similarity functions (Not submitting to this prototype for now)
- **Pooja:** Implemented the GraphDegree, GraphSimilarity, LeadTextSearcher, PageSearcher. Contributed in Utils. Contributed in maintenance of Gitlab repository.