TREC Car - Data Science Code Submission 3

Tucker Owens, Bahram Behzadian, Bryan Hang Zhang

April 3, 2017

1 Github Repositories

Cluster Ranking method & Baseline repository:
https://github.com/gummibearehausen/trecCarProject/graphs/contributors

Knowledge Base & Knowledge Base Dictionary derivation repository:
https://github.com/gummibearehausen/Wissensdatenbank

Query Expansion method:
https://github.com/tuckerowens/trec-car
note that the KB and Dictionary files are not provided due to size. Additionally the paths to these files are hardcoded into `trec-car/src/main/java/edu/unh/cs/trec/entity/EntityHelper.java`

2 Implementation

0. Baseline finish 100% We are planning to use a probabilistic retrieval framework such as Okapi BM25 and state-of-the-art retrieval model: Vector Space Model(VSM) to (1) create a baseline of retrieving passages from half Wikipedia. (2) work as the initial retrieval architecture to generate a pseudo search result.

1. Query processing finish 100% we plan to process the query in an optimal way that basic semantic and conceptual units can be generated. It also improve the concept mapping to the knowledge base. For the baseline, we just use the raw text (child heading text plus page name) as a query; We use parent heading/ stubs as an expansion; We like to tokenize the query text then we would like to convert the query text into N-grams.

2.2. Knowledge base & entity dictionary building finish 100% We map the N-grams to the DBpedia knowledge base and link as many N-grams (semantic unit) to the entities as possible;.

2.3. Knowledge base mapping finish 100% We map the N-grams to the DBpedia knowledge base and link as many N-grams (semantic unit) to the entities as possible;.

2.4. Knowledge base mapping finish 100%(1) We map the N-grams to our Wikipedia knowledge base with our Wikipedia entity knowledge dictionary.

3. Conceptual neighborhood finish 100% A conceptual neighborhood which is a sub-graph of entities can be generated. With the union of all conceptual neighborhoods used as query expansion, the topical structure can be captured, and the ranking of task 1 can be improved. We hope to further improve the entity filtering method to augment the query.

4.1. Neighborhood entities pruning finish 100% in Step 3, we generate a set of relevant entities, we further prune the entities as expansion. We would like to try to prune the most popular entities according to the number of their undirected linkages, then use them for query expansion.

4.2 Re-ranking the pseudo search result finish 100% We plan to explore the entities of passages in the pseudo search result. We can also re-rank the passages according to the similarity (Jacquard as default) of entity distributions between the neighborhood and passages, then re-rank the passages with the combination of the two rankings.

4.3 Re-rank the pseudo search result with a topic model finish 100% We will try to use a topic model (Dirichlet Mixture Model with one topic assignment) to topically cluster algorithm to cluster the top-K pseudo search result at query time. Next, we re-rank the cluster against the query based on similarity, finally we re-rank the document with the combination of the two rankings.

3 Experiment Results

3.1 Baseline

Corpus: `corpus-v1.4.zip`
Outline: `all.test200.cbor.outlines`
Qrel: `all.test200.cbor.article.qrels`
BM25: k1 = 1.2, b = 0.75, *Lucene 4.10.1*
MAP@1000: 0.150
precision at 1000: 0.062
precision at R: 0.202
mrr: 0.504

3.2 Cluster Re-ranking

Summary: In general, clustering re-ranking method return similar result to the baseline. In spite of the nature of dirichlet distribution, the MAP achieve almost the same result as the baseline result with setting($\alpha = 0.2, \beta = 0.2 K = 20, iteration = 20, SampleSize = 1000$), when sample size is double the size, the re-rank result is slightly lower than the baseline. The results with different interpolation setting indicates that (1) the ranking of clusters is consistent with the document ranking, the average number of the clusters using test 200 outline is about 15-20 clusters out of top 1000 documents from the pseudo search result. (2) The clustering achieves homogeneous clusters. Clustering is sensitive to the text occurrences, we would like to explore the cluster-ranking method using tokens within certain range of the TFIDF as the the cluster input in the future.

$$Rank'(Document) = \lambda * Rank_{pseudo}(Document) + (1 - \lambda) * Rank_{Cluster}(Cluster_{Document})$$

Corpus: `corpus-v1.4.zip`
Outline: `all.test200.cbor.outlines`
Qrel: `all.test200.cbor.article.qrels`
dmm. init k = 20,$\alpha = 0.2$,$\beta = 0.2$,Sampling iteration =20, sample size= top 1000.
$\lambda = 0.5$
MAP@1000: 0.150
precision at 1000: 0.062
precision at R: 0.202
mrr: 0.501

$\lambda = 0.2$
MAP@1000: 0.144
precision at 1000: 0.063
precision at R: 0.120
mrr: 0.483

$\lambda = 0.8$
MAP@1000: 0.150
precision at 1000: 0.062
precision at R: 0.204
mrr: 0.504
dmm. init k = 20,$\alpha = 0.2$,$\beta = 0.2$,Sampling iteration =20, sample size= top 2000.

$\lambda = 0.5$

MAP@1000: 0.149
precision at 1000: 0.062
precision at R: 0.203
mrr: 0.500

$\lambda = 0.2$
MAP@1000: 0.148
precision at 1000: 0.062
precision at R: 0.202
mrr: 0.492

$\lambda = 0.8$
MAP@1000: 0.150
precision at 1000: 0.06
precision at R: 0.204
mrr: 0.504

dmm. init k = 20,$\alpha = 0.1$,$\beta = 0.1$,Sampling iteration =20, sample size= top 1000.
$\lambda = 0.5$
MAP@1000: 0.145
precision at 1000: 0.063
precision at R: 0.204
mrr: 0.501

3.3 Query Expansion

An additonal method was implemented using a knowledge base derived from wikipedia. This method links N-grams from the queries to nodes in the knowledge graph. The neighborhood of each KB node is then extracted and used as a query expansion. There is an additional constrait on the inclusion of neighboring entities, which is only neighbors with 5 or less links are included. This was done as a simple method to exclude highly linked nodes, which could introduce noise into the query.

Results are from:
Corpus: `release-v1.4`
Outline: `all.test200.cbor.outlines`
Qrel: `all.test200.cbor.hierarchical.qrels`

```
   Eval(
mrr=0.09885849456829737,
p@5=0.03224669603524244,
r-prec=0.04557057993401605,
map=0.06890922693469953)
```