CSI4107 Assignment 2 report

Group number 6

Group members and tasks

Brandon Lagacé 300088150

- Implementation of experimental method 2 (reranking of document results using word embeddings from word2vec/ GloVe)
- Provided explanation of experimental method 2

Vidulash Rajaratnam 8193098

- Evaluation of results using trec_eval for method 1 and method 2
- Modifications to method1 to output txt file in correct format
- Analysis of trec_eval scores

Matthew Tran 300028206

- Implementation of experimental method 1 (reranking of document results using BERT and sent2vec library)
- Analysis of results and discussion

Running the program

- All appropriate resources (stopwords, test queries, tweet library) should be included in the .zip. To run the program simply navigate in your *cli* command line to the appropriate directory containing the program and type the command *python assignment1.py*
- There will be a short delay while the preprocessing and ranking is done, then there will be a user prompt for query results run name. Type in the identifier you wish for the run. **Note: if you want to conserve runs you should save the file *results_file.txt* in a separate directory as it will be overwritten**
- **There are 2 files for both experimental neural network methods (explained below)** that can be run separately to generate results. *Please note that running over all 49 queries in the first method can take a long time so a quick test can be done by modifying the second for loops to only run 1 or 2 queries.*
- The formatted query results should be available in the generated *results_file_method1/2.txt* files

Explanation of experiments and Results

Method 1: Reranking using sent2vec

The first part of this method ran similarly to assignment 1 with the preprocessing, indexing and cosine similarity ranking steps being identical. However, instead of writing these results to a file the top 1000 results were stored in a list to be used for the next step. This list was passed into a function along with the indexed Documents to be passed through the vectorizer provided by the sent2vec module. This function encoded sentences, in this case the tweets, using the BERT language model and returned the ‘berted’ tweet and the associated doc id.

The next step was to compute the similarity measure distances between the encoded tweets and the query (which was similarly encoded). These could be added to a list and easily sorted using a built-in python function from low (good) to high (bad). This concluded the reranking step using the NLP method to try and extract greater accuracy in our analysis. The rest of the method 1 file is trivial as it was a matter of repeating the steps to write the results to a file. It should be noted that for a full 49 queries using this method was extremely time consuming during the BERT encoding phase combined with the need to recalculate and resort all the results.

Method 2: Reranking using word2vec/GloVe (word embeddings)

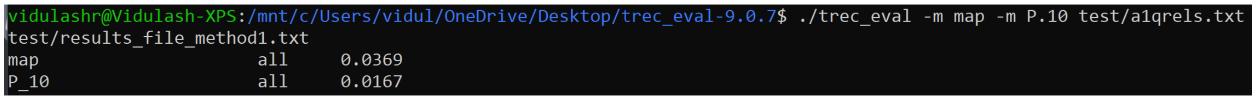
______The first part of this method ran similarly to assignment 1 with the preprocessing, indexing and cosine similarity ranking steps being identical. However, instead of writing these results to a file, the top 1000 results were stored in a list to be used for the next step. This list was used to access the document IDs, the document’s tweet and the cosine score. We would then run through each document's tweet and calculate the cosine similarity between the tweet and our query using the word embeddings from gensensim ***glove-twitter-25***.

The next step was to rerank the top 1000 results with the previously obtained score and the newly calculated one. This was done by taking the average of both previous and newly calculated scores. This allows for the previous score to have weight on the final score. Similarly, this can be said for the score determined by word embeddings. In result, there was an increase in both P10 and MAP. Using word embeddings enriched our results by expanding our queries to allow similar words to be added to our query.

Evaluation of results

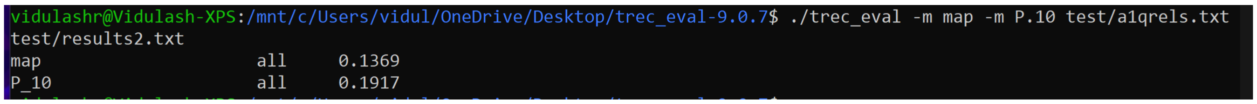
For the evaluation of our results, we once again utilized the trec_eval script, which is a tool used to evaluate rankings that is sorted by relevance. Using our results file from our models with method 1 using sent2vec and method 2 using word2vec and a standard set of judged results, also known as the query relevance list, we were able to calculate the MAP (mean average precision) and P10 (precision in the first 10 documents retrieved). We were able to get the evaluation measure of 0.0369 for the MAP score and 0.0167 for the P10 (Figure 1.0) for the method using sent2vec. However, our evaluation of method 2 was much more promising with a score of 0.1369 for the MAP score and 0.1917 for the P10 (Figure 2.0).

The P10 score indicates that 1.7% of the documents in the top-10 results are relevant documents in method 1 while 19.2 % of the documents in the top-10 results are relevant in method 2. However, we mention once again that this specific metric has its flaws, since even in a perfect system, it may give a low score if some queries don't have multiple relevant documents. (Ex. Only 2 relevant documents out of top-10 results in a score of 0.2). While our MAP scores for both methods indicate that overall, the mean of the average precisions for each query are better than our precision scores for method 1 while lower than the precision scores in method 2. Method 1 had a score of 0.0369 or 3.7% while method 2 had a greater score of 0.1369 or 13.7%. This is a useful metric to indicate how well our models are at retrieving the relevant documents overall. As seen by the trec_eval, our score is only satisfactory for method 2 since it indicates our model only works correctly about 1/7th of the time on average. Evidently, method 1 did not perform as well since it only works correctly 1/30th of the time on average.



```
vidulashr@Vidulash-XPS:/mnt/c/Users/vidul/OneDrive/Desktop/trec_eval-9.0.7$ ./trec_eval -m map -m P.10 test/a1qrels.txt
test/results_file_method1.txt
map                     all     0.0369
P_10                    all     0.0167
```

Figure 1.0: Trec_eval evaluation scores for MAP and P10 for method 1



```
vidulashr@Vidulash-XPS:/mnt/c/Users/vidul/OneDrive/Desktop/trec_eval-9.0.7$ ./trec_eval -m map -m P.10 test/a1qrels.txt
test/results2.txt
map                     all     0.1369
P_10                    all     0.1917
```

Figure 2.0: Trec_eval evaluation scores for MAP and P10 for method 2

First 10 results for method 1

Query 3

```
3 Q0 30218019980447744 0 0.14803290305284766 myRun
3 Q0 32684585548517376 1 0.17055058564226533 myRun
3 Q0 34905563364065280 2 0.17415912265271483 myRun
3 Q0 31169178027163648 3 0.1753558185239703 myRun
3 Q0 34640348076572673 4 0.17734779790433453 myRun
3 Q0 33914951038926848 5 0.17746264511494592 myRun
3 Q0 28984571475271680 6 0.17938425571720074 myRun
3 Q0 34036814926843904 7 0.1818956166876562 myRun
3 Q0 29869233252868096 8 0.18227363940307706 myRun
3 Q0 30018052766572544 9 0.18297911277851486 myRun
3 Q0 32738733216243712 10 0.18911519063418714 myRun
```

Query 10

```
10 Q0 31131202018611200 0 0.1653749170362946 myRun
10 Q0 31178885412429825 1 0.18453988092419826 myRun
10 Q0 31396671703220224 2 0.1915903839548312 myRun
10 Q0 31157310583734272 3 0.19238833929223498 myRun
10 Q0 32634716012154881 4 0.1974449485144183 myRun
10 Q0 34000533693931520 5 0.19757906013087911 myRun
10 Q0 30959570411065344 6 0.1985838990637423 myRun
10 Q0 30637812046897152 7 0.20070745845380877 myRun
10 Q0 30964380627640321 8 0.20486495021132112 myRun
10 Q0 29265765374885888 9 0.2057819761648222 myRun
10 Q0 31335062007980032 10 0.21060781618549285 myRun
```

First 10 results for method 2

Query 3

```
3 Q0 32204788955357184 0 0.6902456371731827 40
3 Q0 33711164877701120 1 0.6625195213675849 40
3 Q0 29613127372898304 2 0.6323114259668283 40
3 Q0 32383831071793152 3 0.6223046544354627 40
3 Q0 32411439918489600 4 0.6212603124896311 40
```

3 Q0 32387196078006272 5 0.6212603124896311 40

3 Q0 32384227123134464 6 0.6212603124896311 40

3 Q0 32878302150529024 7 0.6021067133099864 40

3 Q0 32443364628500480 8 0.6014276452826367 40

3 Q0 32211683082502144 9 0.5918538130078066 40

Query 10

10 Q0 31396671703220224 0 0.6438296327559396 40

10 Q0 31086675086016513 1 0.5881784829905405 40

10 Q0 31426139767443456 2 0.5860043837703695 40

10 Q0 31333924969914369 3 0.5837766869267202 40

10 Q0 29906946052063232 4 0.5820596448346316 40

10 Q0 31073993943416832 5 0.577887944488721 40

10 Q0 30806804950683648 6 0.5759734089042853 40

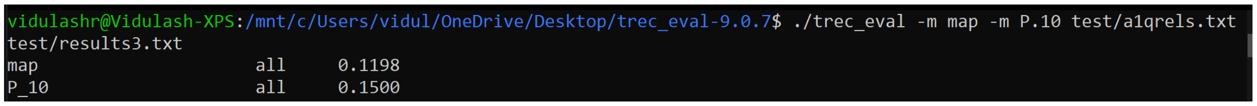
10 Q0 31416724976836609 7 0.5745998131552097 40

10 Q0 31425120488660992 8 0.5744173235278157 40

10 Q0 31265266654515200 9 0.5723321924004736 40

Discussion

Evidently we can see from *Figure 3.0* that our results for MAP and P10 scores were higher without using any additional experimental methods in Assignment 1 than in method 1. It's also clear the second method, which used a library with an algorithm that was pre-trained on an existing twitter corpus (glove-25-twitter) yielded better results because it was more tuned to our specific task, rather than the general BERT encoding of the first method.



```
vidulashr@Vidulash-XPS:/mnt/c/Users/vidul/OneDrive/Desktop/trec_eval-9.0.7$ ./trec_eval -m map -m P.10 test/a1qrels.txt
test/results3.txt
map                     all     0.1198
P_10                    all     0.1500
```

Figure 3.0: Trec_eval evaluation scores for MAP and P10 for Assignment 1 (no neural networks)

Still, the fact the our scores were lower must indicate that there is still some discrepancy between theoretical model predictions and real world performance, as these libraries can't just be added off the shelf but need some more fine-tuning to deliver accurate results. It is worth noting that for this assignment pre-trained libraries were used and they were quite common ones too, not necessarily tailor-made for the task. In method 1, the reranking step was also essentially implemented on a text index, instead of a boolean retrieval index which may have influenced the results.

A more advanced implementation could have also included a probabilistic model such as the Latent Semantic Indexing seen in class, although this would have also been more expensive in terms of computation, even if we could have bypassed the stemming step.