CSI 4107 Assignment 2: Neural Information Retrieval System Based On BERT

Students: Jiaxun Gao [300063462] & Xiang Li [300056427]

0. Structure tree

IR System

- Document.py: Store a document object, one document is represented by a vector
- IRSystem.py: Our main program
- QueryParser.py: Query iterator, read query file and generate query text
- Tokenizer.py: preprocessing utility
- Models: folder to store the word2vec model
- Output: folder to store output results.
- Files (To run our program, you should create all the folders by your own)
    - StopWords.txt: File that stores stop words
    - topics_MB1-49.txt: File that stores queries
    - Trec_microblog11.txt: File that contains all documents to be indexed

1. Installation and setup

We run all our experiments on a NVIDIA RTX3080ti gpu since it supports CUDA acceleration. Here is the instruction about how to set up the needed environment:

(1) Follow this instruction to install cuda sdk toolkid.

(2) Follow this instruction to Install pytorch. Please choose the cuda 11.3 compute platform.

(3) Type the following instructions in terminal/command line to install dependencies:

```
To install nltk: pip install nltk
To install sentence tranformers: pip install sentence_transformers
To install genism: pip install genism
```

(4) Download nltk stopwords: open a new python window in command line, then type

```
>>> import nltk
>>> nltk.download('stopwords')
```

Alternatively you can also type directly python -m nltk.downloader stopwords in terminal to do this.

(5) Create three folders under the parent folder, namely 'files', 'outputs', 'models'. The first folder stores all the corpus and documents we need, please make sure that 'StopWords.txt', 'topics_MB1-49.txt' and 'Trec_microblog11.txt' are under the 'files' folder subfolder. 'output' folder will store all the generated results; 'models' will store the word2vec model that we will train in our code.

In case you don't have access to a gpu(given that the situation of current gpu market), you can just ignore the first step and follow step 2-5 to complete the installation, however in step2 you will need to choose the compute platform as cpu instead of cuda.

Another option to run our code is using google colab, since it natively supports cuda acceleration. We highly recommend you choose this option since cpu is too slow for serval computation tasks in our project. Google colab is the first option when you do not have a gpu.

2. Group work distribution

Jiaxun Gao implemented Tokenizer, QueryParser, Document classes, he also provided some preprocessing ideas of the data. Xiang Li designed the data structure of inverted index and did some optimization to increase the performance and make the code run faster.

Jiaxun Gao was the programmer of BERT model and Xiang Li was the trainer of the word2vec model.

This report was composed by both two of them.

3. Design & Functionality

We implemented a neural information retrieval system which indexes a collection of Twitter posts, then we tested our system on 49 queries: we take the top 1000 results which match query the best.

Our system takes one document file (i.e. Trec_microblog11.txt) which is the collection of documents that we want to index and a query file (i.e. topics_MB1-49.txt) which contains some query that we are interested in, the system then produce a TREC-format output file, the output file contains top 1000 matches of each query.

To run the program, just enter 'python Neural-IRSystem.py' in the parent directory terminal.

We implemented all 1,2,3 methods in our assignment, in the next section we will describe how the system evolves from the previous IR System which was implemented in Assignment 1 to this brand-new Neural IR System.

4. Implementation and Optimization

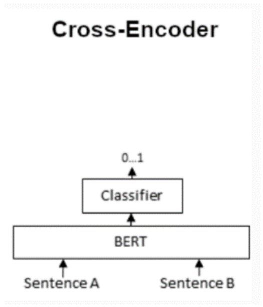
The IR System in our Assignment 1 was implemented using Java. So, in the first step of our Assignmnet2 we reimplemented everything we have in Assignment 1 in python. As there's nothing new, the performance of the system remains the same as in assignment1: map = 0.2089 & p@10 = 0.2163. This is our baseline model, and we did serval successive improvements based on our baseline model.

4.1 use bm-25 similarity instead of tf-idf

In the first variant we implanted, we used the bm-25 to measure similarity instead of df-idf. By replacing df-idf with bm-25, the improvement of the system was drastic: we boost the map to 0.2970 and the p@10 to 0.3857. Consequently, we abandon tf-idf in all our successive steps and use bm-25 instead.

4.2 bm-25 + cross encoder rerank [1st proposed method]

In this new variant we introduce a new procedure in our system: rerank procedure using cross encoder. After the initial retrieval of the documents using the bm-25 similarity, we rank all the retrieved documents using another model named cross encoder. Based on the official document of sentence transformer model, collecting the initial retrieval using traditional IR system(or neural IR system) then reranking the retrieved documents using cross-encoder is the right use of this model. In the current version, the initial retrieval was still performed by the traditional IR system with bm-25 similarity score, then we rerank the retrieved results using a pretrained cross encoder(ms-marco-MiniLM-L-6-v2).



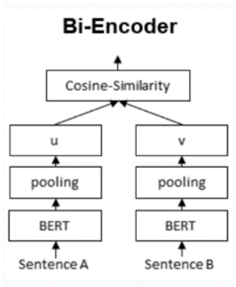
In the cross encoder, we pass both sentences simultaneously to the Transformer network. It produces then an output value between 0 and 1 indicating the similarity of the input sentence pair.

The improvement was HUGE: we boosted the map to 0.3804 and p@10 to 0.4446. All our team are happy to see this improvement. Our system was way more useful compared to the previous versions.

Please note that from this step, if you run our code with only cpu, each run will take approximately 30 minutes.

4.3 bi-encoder + cross encoder rerank [3rd proposed method]

In our previous system, the initial retrieval was collected by a traditional IR system with bm-25 similarity. In the current system, we replace this process by a neural model: a bi encoder model.



In the bi encoder model, we pass to a BERT independently the sentence A (the query) and B(document), which result in the sentence embeddings u and v. These sentence embeddings can then be compared using cosine similarity.

By doing so, the map achieves 0.3933 and p10 achieves 0.4510. We did serval fine tuning in this bi-encoder, such as the max depth of the network etc., only the highest map and p10 are reported here.

4.4 bi-encoder + cross encoder rerank + query expansion [2rd proposed method]

After we implemented the previous system, Jiaxun Gao proposed to implement the query expansion to further improve the system performance.

We did the following: first we trained a word2vec model to find the synonyms of a given word, we implemented this process in train.py. Then in the retrieve process, for a given query q0, the system will first compute the most similar query q1 then concatenate them together as q’ = q0 + q1, then the initial retrieval will be performed on q', next the rerank process will be done based on the original query q0. We noticed that if the rerank process was performed based on q', the system performance will degenerate drastically.

In this step, serval hyper parameters were fine-tined via grid search. For instance, the way we calculate q': in our initial design, the k most similar queries q1, …, qk were computed and were concatenated together to compute the q' = q1 + … + qk. Experiments showed that the best value for k is 1. One reason which might explain this phenomenon is that we only have limited train and test data, if one wants to ask a question to the IR system, she only have very limited way of doing so, thus any larger query expansion will bring chaos to the system and thus will make the system performance degenerate.

The best map and p10 we achieved by using the query expansion are 0.3954 and 0.4551 respectively.

4.5 Summary

Here is a summary of different models we trained:

Model name\performance	map	P@10
Tf-idf	0.2089	0.2163
Bm-25	0.2970	0.3857
Bm-25 + rerank	0.3804	0.4446
Sentence transformer + rerank	0.3933	0.4510
Sentence transformer + rerank + query expansion	**0.3954***	**0.4551***

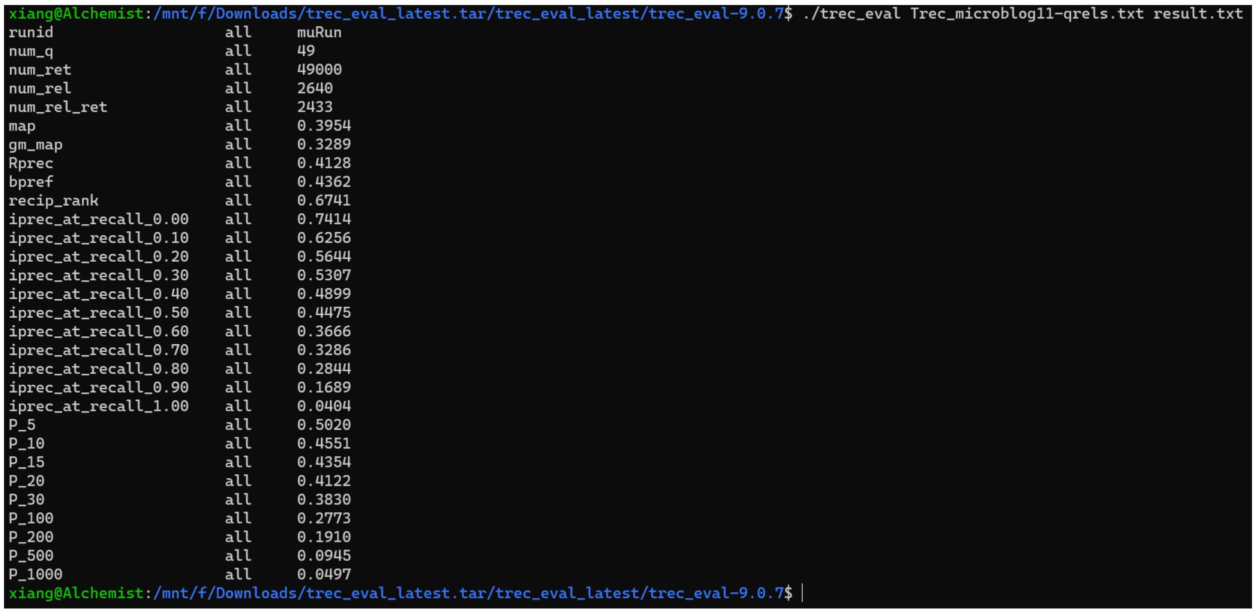
As shown in the above table, based on our experiments, expanding the original query using word net embedding then use bi-encoder (sentence transformer) as initial text retrieval and rerank the initial result by cross encoder achieves the highest performance.

One major optimization we did is that we reimplemented the tokenizer class. In this assignment 2, we used an external tweet tokenizer library. We found that a good tokenizer is crucial to the system. Compared to assignment1, in assignment 2 we remove all the hyper links and mentions (in the format of @somebody) in the raw text. This will improve the map and p@10 for 0.05 and 0.04 respectively in all systems.

There are still other minor optimizations we performed such as the structure of document object etc and the way we computed tf-idf in favor of runtime. These minor changes didn't change much the performance in terms of map and p10, however we believe all of them are very nice practices

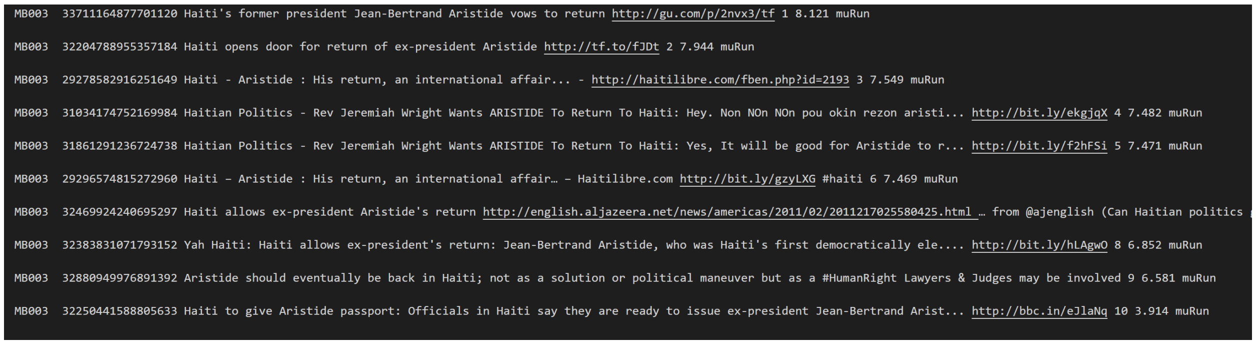
5. Test result based we obtained on our best model

Here is the statistics for our best model:



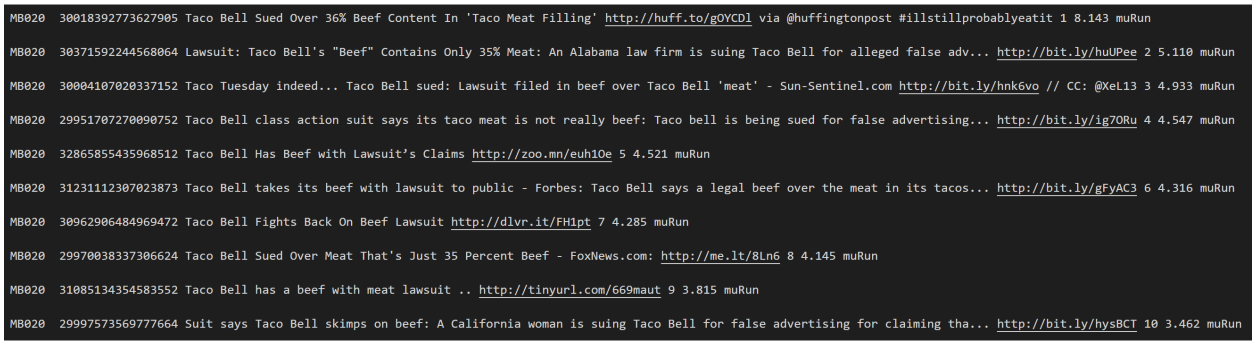
```
xiang@Alchemist:/mnt/f/Downloads/trec_eval_latest.tar/trec_eval_latest/trec_eval-9.0.7$ ./trec_eval Trec_microblog11-qrels.txt result.txt
runid                   all     muRun
num_q                   all     49
num_ret                 all     49000
num_rel                 all     2640
num_rel_ret             all     2433
map                     all     0.3954
gm_map                  all     0.3289
Rprec                   all     0.4128
bpref                   all     0.4362
recip_rank              all     0.6741
iprec_at_recall_0.00    all     0.7414
iprec_at_recall_0.10    all     0.6256
iprec_at_recall_0.20    all     0.5644
iprec_at_recall_0.30    all     0.5307
iprec_at_recall_0.40    all     0.4899
iprec_at_recall_0.50    all     0.4475
iprec_at_recall_0.60    all     0.3666
iprec_at_recall_0.70    all     0.3286
iprec_at_recall_0.80    all     0.2844
iprec_at_recall_0.90    all     0.1689
iprec_at_recall_1.00    all     0.0404
P_5                     all     0.5020
P_10                    all     0.4551
P_15                    all     0.4354
P_20                    all     0.4122
P_30                    all     0.3830
P_100                   all     0.2773
P_200                   all     0.1910
P_500                   all     0.0945
P_1000                  all     0.0497
xiang@Alchemist:/mnt/f/Downloads/trec_eval_latest.tar/trec_eval_latest/trec_eval-9.0.7$
```

Here is the first 10 results of our system for query3 (Haiti Aristide return):



```
MB003  33711164877701120 Haiti's former president Jean-Bertrand Aristide vows to return http://gu.com/p/2nvx3/tf 1 8.121 muRun

MB003  32204788955357184 Haiti opens door for return of ex-president Aristide http://tf.to/fJDt 2 7.944 muRun

MB003  29278582916251649 Haiti - Aristide : His return, an international affair... - http://haitilibre.com/fben.php?id=2193 3 7.549 muRun

MB003  31034174752169984 Haitian Politics - Rev Jeremiah Wright Wants ARISTIDE To Return To Haiti: Hey. Non NOn NOn pou okin rezon aristi... http://bit.ly/ekgjqX 4 7.482 muRun

MB003  31861291236724738 Haitian Politics - Rev Jeremiah Wright Wants ARISTIDE To Return To Haiti: Yes, It will be good for Aristide to r... http://bit.ly/f2hFSi 5 7.471 muRun

MB003  29296574815272960 Haiti – Aristide : His return, an international affair… – Haitilibre.com http://bit.ly/gzyLXG #haiti 6 7.469 muRun

MB003  32469924240695297 Haiti allows ex-president Aristide's return http://english.aljazeera.net/news/americas/2011/02/2011217025580425.html … from @ajenglish (Can Haitian politics

MB003  32383831071793152 Yah Haiti: Haiti allows ex-president's return: Jean-Bertrand Aristide, who was Haiti's first democratically ele.... http://bit.ly/hLAgwO 8 6.852 muRun

MB003  32880949976891392 Aristide should eventually be back in Haiti; not as a solution or political maneuver but as a #HumanRight Lawyers & Judges may be involved 9 6.581 muRun

MB003  32250441588805633 Haiti to give Aristide passport: Officials in Haiti say they are ready to issue ex-president Jean-Bertrand Arist... http://bbc.in/eJlaNq 10 3.914 muRun
```

Here is the first 10 results of our system for query20 (Taco Bell filling lawsuit):



```
MB020  30018392773627905 Taco Bell Sued Over 36% Beef Content In 'Taco Meat Filling' http://huff.to/gOYCDl via @huffingtonpost #illstillprobablyeatit 1 8.143 muRun

MB020  30371592244568064 Lawsuit: Taco Bell's "Beef" Contains Only 35% Meat: An Alabama law firm is suing Taco Bell for alleged false adv... http://bit.ly/huUPee 2 5.110 muRun

MB020  30004107020337152 Taco Tuesday indeed... Taco Bell sued: Lawsuit filed in beef over Taco Bell 'meat' - Sun-Sentinel.com http://bit.ly/hnk6vo // CC: @XeL13 3 4.933 muRun

MB020  29951707270090752 Taco Bell class action suit says its taco meat is not really beef: Taco bell is being sued for false advertising... http://bit.ly/ig7ORu 4 4.547 muRun

MB020  32865855435968512 Taco Bell Has Beef with Lawsuit’s Claims http://zoo.mn/euh1Oe 5 4.521 muRun

MB020  31231112307023873 Taco Bell takes its beef with lawsuit to public - Forbes: Taco Bell says a legal beef over the meat in its tacos... http://bit.ly/gFyAC3 6 4.316 muRun

MB020  30962906484969472 Taco Bell Fights Back On Beef Lawsuit http://dlvr.it/FH1pt 7 4.285 muRun

MB020  29970038337306624 Taco Bell Sued Over Meat That's Just 35 Percent Beef - FoxNews.com: http://me.lt/8Ln6 8 4.145 muRun

MB020  31085134354583552 Taco Bell has a beef with meat lawsuit .. http://tinyurl.com/669maut 9 3.815 muRun

MB020  29997573569777664 Suit says Taco Bell skimps on beef: A California woman is suing Taco Bell for false advertising for claiming tha... http://bit.ly/hysBCT 10 3.462 muRun
```

6. Conclusion

In this assignment, we implemented a neural information retrieval system start from scratch. Finally, we achieved a satisfying performance. Due to the limited time we have, there are still many options that we didn’t try. For instance, in the sentence transformer bi encoder model we just use the pre-trained model, but we didn’t fine tune the word vector embedding.

Another way to improve the performance of the retrieval system is to collect more data. We notice that all the data from TREC2011-2015 are available online. We are planning to retrain our model on the TREC2012-2015 data and test the performance on TREC2011 when we have more free time.

We all enjoyed a lot in this project, thanks!