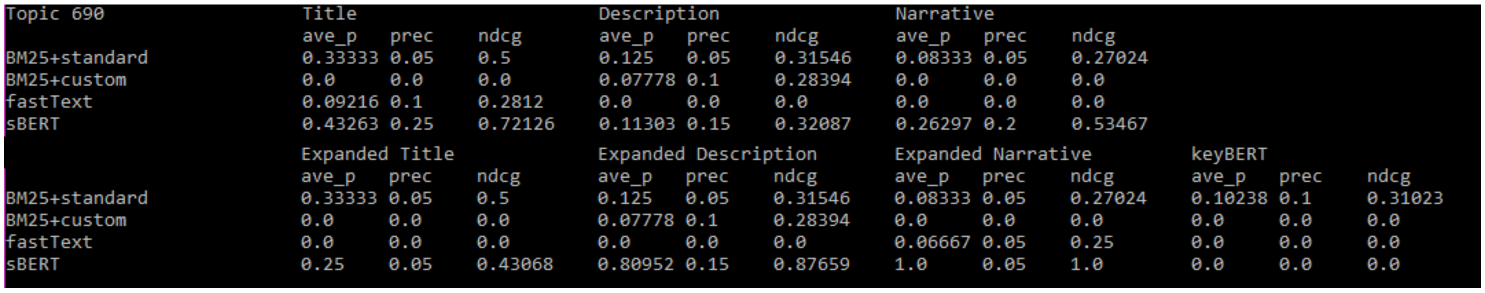
Team members: Daeja Showers, Benjamin Siege, Michael Gardner
Team member submitting code: Michael Gardner
TREC topic number: TREC 690

Original TREC title, description, and narrative:
TITLE: college education advantage
DESC: Find documents which describe an advantage in hiring potential or increased income for graduates of U.S. colleges.
NARRATIVE: Relevant documents cite some advantage of a college education for job opportunities. Documents citing better opportunities for non-college vocational-training is not relevant.

Queries:
Expanded Title, Expanded Description, and Expanded Narrative will be how they are referred to, but they were not made as "user queries". They are runtime wordnet synonym extractions. We normalized out stopwords, found that Adverbs and Verbs were less informative to our BASE topic queries and dropped them, then concatenated in synonym sets from wordnet. In this version the Title becomes: "college college education instruction teaching pedagogy didactics educational activity education advantage vantage advantage" (An attempt to make disjoint and conjoin synonym queries without concatenation can we explored in the files extractALT and evaluateALT, customized for BERT use, but none achieved better retrieval. In that version, Descritpion becomes ['advantage potential increased income alumnus United States government college', 'vantage possible increased income alumna United States college', 'advantage potential increased income alum U.S. government college', 'advantage likely increased income graduate United States college', 'advantage potential increased income graduate United States of America college', 'advantage likely increased income graduate America college', 'reward possible increased income alumna United States college']).

Summary:
Our team improved our search with queries made of concatenated synonyms of the most informative query terms, against matches on Title AND Content. Bert returned the best results, when querying the Expanded Description, though improvement was seen in different categories using different techniques, like the ALT methods improved scores on the Expanded Narrative, especially FastText.

Output:
Our output is a user-visible runtime table of all the query types against all the search types.
With the evaluate.py functionality use the --produce_metrics argument.



```
Topic 690                 Title                    Description              Narrative
                          ave_p   prec    ndcg     ave_p   prec    ndcg     ave_p   prec    ndcg
BM25+standard             0.33333 0.05    0.5      0.125   0.05    0.31546  0.08333 0.05    0.27024
BM25+custom               0.0     0.0     0.0      0.07778 0.1     0.28394  0.0     0.0     0.0
fastText                  0.09216 0.1     0.2812   0.0     0.0     0.0      0.0     0.0     0.0
sBERT                     0.43263 0.25    0.72126  0.11303 0.15    0.32087  0.26297 0.2     0.53467
                          Expanded Title           Expanded Description     Expanded Narrative       keyBERT
                          ave_p   prec    ndcg     ave_p   prec    ndcg     ave_p   prec    ndcg     ave_p   prec    ndcg
BM25+standard             0.33333 0.05    0.5      0.125   0.05    0.31546  0.08333 0.05    0.27024  0.10238 0.1     0.31023
BM25+custom               0.0     0.0     0.0      0.07778 0.1     0.28394  0.0     0.0     0.0      0.0     0.0     0.0
fastText                  0.0     0.0     0.0      0.0     0.0     0.0      0.06667 0.05    0.25     0.0     0.0     0.0
sBERT                     0.25    0.05    0.43068  0.80952 0.15    0.87659  1.0     0.05    1.0      0.0     0.0     0.0
```

figure 1. Example results table at runtime, showing the best achieved results

It is also possible to display the associated search results. Either through the evaluate command line, or by running the flask app.

Results:
We were able to show marked improvement, as we chose the lowest scoring TREC topic. Some example improvements are the sBert results on the basic Title query, and the sBert results on the Expanded Description. At the beginning the best search method's top 20 results had relevance judgments of 690-0 for all 20. Now, several of these methods get 1-3 "very relevant" and 2-4 "relevant" results.

Dependencies:
All of the following can be pip installed, except elastic search, which can be gotten from www.elasticsearch.co elasticsearch~=7.12.0
elasticsearch-dsl, sentence-transformers, flask~=1.1.2, numpy~=1.20.2, zmq~=0.0.0, tqdm~=4.59.0, pyzmq~=22.0.3, nltk~=3.5, keybert~=0.3.0

Build instructions:
First elasticsearch must be running: from its directory ./bin/elasticsearch(.bat)
Then Bert and FastText need to be engaged:

```
python -m embedding_service.server --embedding sbert  --model msmarco-distilbert-base-v3
python -m embedding_service.server --embedding keybert  --model stsb-roberta-base-v2
```

Next, we haven't included the massive data .jl, but you'll need to index that with command:

```
python load_es_index.py --index_name wapo_docs_50k --wapo_path data/subset_wapo_50k_sbert_ft_filtered.jl
```

If you want to initiate the flask app and interface with a browser:

```
python hw5.py
```

proceed to a browser and go to the localhost

And lastly, for command line interface:

```
python evaluate.py --index_name wapo_docs_50k --topic_id 690 --produce_metrics
```

And you can specify which results you want to see with different arguments:

```
python evaluate.py --index_name wapo_docs_50k --topic_id 690 --query_type expanded_description --vector_name sbert_vector --top-k 20
```

Team member contributions:
The work was very mutual, and often processed in parallel, but the final code modules broke down like this:
Daeja Showers: bulk of evaluate.py, keybert
Benjamin Siege: extract.py, hw5.py, evaluate.py improvements
Michael Gardner: improvements to extract.py and evaluate.py and doctemplate.py

Links:
https://github.com/digiberri/cosi132a_elasticsearch_exploration
https://docs.google.com/presentation/d/1o3fiFeaHv2uC37psFtGQ-VMCPlX9xsjotapMtPVevnI/edit?usp=sharing