information-retrieval-final-project

Team

Team members: Ryan Pyatetsky, Daniel Hefel, Nicole Zamora Team member submitting code:

==========================

TREC topic

Number: 439

TREC Information:

Title: inventions, scientific discoveries

Description: What new inventions or scientific discoveries have been made?

Narrative: The word "new" in the description is defined as occurring in the 1990s. Documents that indicate a "recent" invention or scientific discovery are considered relevant. Discoveries made in astronomy or any scientific discoveries that are not patentable are not relevant.

Queries:

From TREC XML

- Title
- Description
- Narrative

User Queries:

- Scientific Discovery

- Scientific Discovery patent
- Scientific Discovery not astronomy

=========================

Summary:

We took several approaches to the document retrieval task:

SBERT doesn't process the full article, but rather takes the first 500 or so words. This means that if relevant information is at the end of the document, it is truncated and not included. To see if processing the full article would make a difference, the article is segmented into portions, vectors created for each portion, and then the embeddings are averaged among portions.

We also trained a doc2vec model on the corpus to create document vectors that could be used for retrieval.

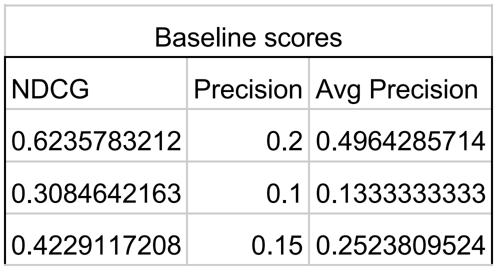
Additionally, original articles are pre-processed by deleting the terms that go below a certain threshold, and then tested against the bm25 custom analyzer.

=========================

Output:

Initial Queries

Baseline Scores



Baseline scores		
NDCG	Precision	Avg Precision
0.6235783212	0.2	0.4964285714
0.3084642163	0.1	0.1333333333
0.4229117208	0.15	0.2523809524

Our Scores

Query Text			NDCG		Precision		Ave Precision
Title	tf-idf	-0.01	0.279	0	0.05	-0.01	0.091
	2-tier	0.14	0.430676	0	0.05	0.15	0.25
	d2v	-0.29	0	-0.05	0	-0.10	0
	sbert	0.20	0.4865	0.1	0.15	0.27	0.365079
Description	tf-idf	0.09	0.35		0.1	0.06	0.139
	2-tier	0.05	0.308464	0.05	0.1	0.06	0.133333
	d2v	-0.26	0		0	-0.08	0
	sbert	0.12	0.380203		0.2	0.11	0.189474
Narrative	tf-idf	-0.02	0.235	0	0.05	-0.01	0.056
	2-tier	0.25	0.504138	0.15	0.2	0.25	0.315476
	d2v	0.04	0.293377	0.05	0.1	0.03	0.094298
	sbert	0.48	0.73255	0.15	0.2	0.37	0.433001

User-made queries

Baseline scores		
NDCG	Precision	Avg Precision
0.2891	0.05	0.1
0.2626	0.05	0.07692307692
0.25	0.05	0.06666666667

Our Results:

Query Text			NDCG		Precision		Ave Precision
Science Discovery	tf-idf	0.01	0.634	0	0.2	0.03	0.527
	2-tier	-0.24	0.386853	0	0.2	0.00	0.496429
	d2v	-0.33	0.292174	-0.1	0.1	-0.39	0.108824
	sbert	-0.02	0.603252	0.05	0.25	-0.15	0.347029
Science Discovery patent	tf-idf	0.02	0.333		0.05		0.5
	2-tier	0.02	0.325995	0	0.1	0.02	0.155556
	d2v	-0.06	0.244651	-0.05	0.05	-0.07	0.0625
	sbert	-0.02	0.285727	0	0.1	-0.04	0.094538
Science discovery not astronomy	tf-idf	0.01	0.436		0.15		0.12
	2-tier	-0.10	0.327246	-0.05	0.1	-0.14	0.1125
	d2v	0.21	0.630929	-0.1	0.05	0.25	0.5
	sbert	-0.01	0.417355	-0.05	0.1	-0.06	0.191667

=========================

Results:

Note: all of our results are compared against the baseline provided by the bm25 custom analyzer.

The 2-tier search system ignored all of the relevant results that were also found in a query including the word astronomy. This approach performed better in all of the TREC query texts, especially on the narrative (almost doubled the ndcg).*Provide a brief overview of the results, which will be described in more detail in the slide presentation.*

=========================

Dependencies:

Describe all resources used for your system, including version numbers and download locations for code obtained from the web.

pip install gensim

pip install -U sentence-transformers

Data

Download from https://drive.google.com/u/0/uc?id=1Y03Cgf-84lua5cmBEt8-4JQKbdgvu3vG&export=download

=========================

Build instructions:

Describe how to build your system and run it. A commented script or scripts including all steps would suffice.

The Word2Vec, retraining of SBERT and 2-tier search system required the use of some code from hw5. So, we have attached zip files to the projects (pa5_official_nicolezamora.zip and ------), and the files that we modified can be found on the main directory of this repository.

doc2Vec

-run the doc2vec.py file and then use the resulting jl file to build the elastic search index. See the query.py file for how to run queries with elastic search

SBERT

-run the sbert.py file and then use the results jl file to build the elastic search index by adjusting the basedoc to take in the new vector

2-Tier Search System

- Base code references hw5:
- Run the elasticsearch server on the terminal
- Build the wapo_50k_index by reading the 50k json file
- Call the evaluate.py file, which applies the bm25 custom analyzer against the user query, and ignores all of the results found in a query composed of the original * text plus the word "astronomy"

tfidf

Create a dictionary for tf per doc and another one for corpus_tf by running the tfidf.py file. This took about 17 minutes to build locally.

========================

Team member contributions:

Ryan:

Wrote the code that will calculate the tfidf score for every document and then remove the terms from all the documents that dont meet the specified threshold. Then created a new jsonline file with the new edited documents so that the elasticsearch could build a new index with the edited documents, which allowed us to use BM25+Custom on the edited documents to compare precision, average precision, and ndcg scores with our original results.

Daniel:

Wrote and trained the doc2vec and sbert approaches and their evaluation file. List of .py files written by Daniel:

- utils.py
- sbert.py
- doc2vec.py
- query.py
- One version of doc_template.py and index.py All files located in the eponymous zip file

Nicole:

Processing of documents: Tried loading the 50k documents into a shelf that would contain the tf scores of each document, to later calculate the tfidf per token. After multiple tries with building the index, the last attempt resulting in a 36 hour code-run that was stopped by the kernel, I decided to change the approach to load the tf scores into a dictionary and then into a pickle, following the TA's recommendation. Developed the 2-tier search system.