CSI 4107 Group 1: Assignment 2

Assignment description

Group Members

Howard Hao En Tseng, 300108234

Leah Young, 300118869

Shang Lin Hsieh, 300121996

Contribution

Name	Contribution	Completed
Howard Hao En Tseng	Method 1	✓
Leah Young	Method 2	X
Shang Lin Hsieh	Method 3	✓

Getting Started

This section is the complete guide to setting up the project and running the program. Follow the steps below to get started.

Python 3.8 is **required** to install the packages used in this project. If you do not have Python 3.8 installed on your machine, you can download it from here.

Recommended: Setting up a virtual environment

This is an optional, but recommended, step to set up a virtual environment for this project. This is especially important if your default Python version is not 3.8.

Before running the following commands, make sure you have python 3.8 installed on your machine. If you do not have Python 3.8 installed on your machine, you can download it from here.

To set up the virtual environment, run the following commands in the **directory of the method to execute**:

```
pip install virtualenv
python -m virtualenv .venv --python=python3.8
.venv/Scripts/Activate.ps1
```

Installing packages

To install the packages used in this project, run the following command:

```
pip install -r requirements.txt
```

Each completed method has instructions on how to run the program. Please follow the instructions provided in the sections, performing them in the directory of the method to execute (ex. If executing Method 1, ensure your current execution directory is `Assignment 2/Method 1`). The AP_collection is required in all of the completed methods as well.

Method 1: Reranking with new similarity scores

The aim of this experiment is to re-rank the results based on new cosine similarity scores between the query and the selected document. The model generates sentence embeddings for each document in the collection, which are then used as vectors to calculate the cosine similarity score.

The sentence transformers library was utilized throughout this project. This library contains various pre-trained models that could be used for this task, which was preferable as we deemed that training and fine-tuning our own models would be too time-consuming.

Implementation

Modification of code from Assignment 1

The pre-processing code was reused from our implementation in Assignment 1. However, as we were not required to perform the level of preprocessing we performed in Assignment 1, this code was simplified to simply fetch the document ID and document text of each document. The `Document` object of this implementation therefore only stores those two variables.

Neural reranking

For each model, the sentence embedding of each document is calculated by each model. The sentence embedding output is the query vector of each document, which is then able to be used to calculate the cosine similarity. This involves iterating through each of the preprocessed documents and running this code to acquire the embedding for the document:

```
doc_embed = model.encode(doc.doc_text, show_progress_bar=False)
```

The embedding of each document is then stored in a file to allow for easier retrieval, as the process for computing the sentence embeddings is a very time-consuming process. This is done using the `pickle` library, which allows us to easily write python objects into files and easily read them when required.

Storing the embedding of each individual document is particularly important on models that take a significant amount of time to compute, such as `sentence-transformers/gtr-t5-xxl` and `Muennighoff/SGPT-1.3B-weightedmean-msmarco-specb-bitfit`, as these models required the use of Google Colab for the embeddings to be computed in a timely manner, and Google Colab may have a chance to time out and lose all data when away from the computer for too long. This was an unfortunate lesson learned by my peers in Method 3.

Once the embeddings for all of the documents in the preprocessed documents are done computing, we save the output to a file containing the embeddings of all of the documents. The embeddings are then used to

calculate the cosine similarities in a method similar to what was implemented in Assignment 1.

79923 document embeddings are generated for each model.

Testing methodology

A list of models that was deemed to be the most suitable for the task of semantic search was made and tested. This included the top downloaded models from the `sentence-transformers` library, such as the MPNet, distilbert, GTR, and T5 transformers, as well as SGPT for testing GPT for semantic search. Multiple versions of each model were used to compare which variant of the model performed the best.

Some of these models were run on a Jupyter Notebook on Google Colab, as the models took a significant amount of time to compute the embeddings for each document. The embeddings were then saved to a file and downloaded to the local machine to be used for the reranking.

Running the program

Once completing the prerequisite setup steps in the Getting Started section inside the directory of this method, run the following command:

```
python main.py
```

Since all 36 models will be sequentially tested upon the execution of this program, adequate computing resources and disk space are required in order to cache the model and load the model into the computer's GPU memory or RAM. The authors of this report assume no responsibility if this program fails to execute due to limited computer resources.

The document embeddings of all of the models below have been generated. However, all of these files exceed the 100 MB file size limit of GitHub and will exceed the allowed size for the submission. As a result, these files will not be included in the assignment submission.

Sentence transformer models used

The following 36 models were tested and compared for this experiment:

- sentence-transformers/LaBSE
- sentence-transformers/all-MiniLM-L12-v2
- sentence-transformers/all-MiniLM-L6-v2
- sentence-transformers/all-distilroberta-v1
- sentence-transformers/all-mpnet-base-v2
- sentence-transformers/all-roberta-large-v1
- sentence-transformers/bert-base-nli-mean-tokens
- sentence-transformers/distilbert-base-nli-mean-tokens
- sentence-transformers/distilbert-base-nli-stsb-mean-tokens
- sentence-transformers/distiluse-base-multilingual-cased-v2
- sentence-transformers/gtr-t5-base
- sentence-transformers/gtr-t5-large
- sentence-transformers/gtr-t5-xl

- sentence-transformers/gtr-t5-xxl
- sentence-transformers/msmarco-MiniLM-L12-cos-v5
- sentence-transformers/msmarco-MiniLM-L6-cos-v5
- sentence-transformers/msmarco-distilbert-base-tas-b
- sentence-transformers/msmarco-distilbert-cos-v5
- sentence-transformers/multi-qa-MiniLM-L6-cos-v1
- sentence-transformers/multi-qa-distilbert-cos-v1
- sentence-transformers/multi-qa-mpnet-base-cos-v1
- sentence-transformers/multi-qa-mpnet-base-dot-v1
- sentence-transformers/nli-mpnet-base-v2
- sentence-transformers/paraphrase-MiniLM-L3-v2
- sentence-transformers/paraphrase-MiniLM-L6-v2
- sentence-transformers/paraphrase-albert-small-v2
- sentence-transformers/paraphrase-distilroberta-base-v2
- sentence-transformers/paraphrase-mpnet-base-v2
- sentence-transformers/sentence-t5-base
- sentence-transformers/stsb-mpnet-base-v2
- sentence-transformers/xlm-r-distilroberta-base-paraphrase-v1
- Muennighoff/SGPT-125M-lasttoken-msmarco-specb
- Muennighoff/SGPT-125M-weightedmean-msmarco-specb
- Muennighoff/SGPT-125M-weightedmean-msmarco-specb-bitfit
- Muennighoff/SGPT-125M-weightedmean-msmarco-specb-bitfitwte
- Muennighoff/SGPT-1.3B-weightedmean-msmarco-specb-bitfit

Muennighoff/SGPT-2.7B-weightedmean-msmarco-specb-bitfit was also being tested, but it was not completed by the time the report was finished due to the model taking a prohibitively long time to fully calculate the embeddings for all of the documents.

Evaluation and Results

Top 10 models by **MAP**:

Rank	Model	MAP	MAP (w/description)	Difference
1	all-mpnet-base-v2	0.3251	0.3230	-0.65%
2	multi-qa-mpnet-base-cos-v1	0.3081	0.3001	-2.6%
3	gtr-t5-large	0.3057	0.2995	-2.03%
4	gtr-t5-xxl	0.3039	0.2933	-3.49%
5	multi-qa-mpnet-base-dot-v1	0.3010	0.2952	-1.93%
6	paraphrase-mpnet-base-v2	0.2847	0.2820	-0.95%
7	gtr-t5-xl	0.2824	0.2732	-3.26%
8	multi-qa-MiniLM-L6-cos-v1	0.2659	0.2714	2.07%
9	gtr-t5-base	0.2628	0.2559	-2.63%

Rank	Model	MAP	MAP (w/description)	Difference
10	multi-qa-distilbert-cos-v1	0.2569	0.2545	-0.93%

Top 10 models by **P@10**:

Rank	Model	P@10	P@10 (w/description)	Difference
1	gtr-t5-large	0.4540	0.4480	-1.32%
2	multi-qa-mpnet-base-cos-v1	0.4520	0.4400	-2.65%
3	all-mpnet-base-v2	0.4460	0.4440	-0.45%
4	gtr-t5-xxl	0.4420	0.4360	-1.36%
5	multi-qa-mpnet-base-dot-v1	0.4420	0.4400	-0.45%
6	gtr-t5-xl	0.4400	0.4360	-0.91%
7	paraphrase-mpnet-base-v2	0.4320	0.4300	-0.46%
8	multi-qa-MiniLM-L6-cos-v1	0.3980	0.4100	3.02%
9	multi-qa-distilbert-cos-v1	0.3940	0.3960	0.51%
10	all-MiniLM-L6-v2	0.3860	0.3880	0.52%

Of these top 10 models, the best model overall is **all-mpnet-base-v2**, with a MAP score of **0.3251** and P@10 score of **0.4460**.

The MAP score of this model is a **63%** improvement over the class average score of 0.1998, and the P@10 score is a **71%** improvement over the class average score of 0.260. However, this is only a marginal improvement over our Assignment 1 TF-IDF scores, with a 0.4% improvement over the MAP score of 0.3238 and 6% improvement over the P@10 score of 0.42.

It is interesting to note that an average difference of -1.64% for the MAP score and -0.36% for the P@10 score between the models with and without descriptions. This indicates that the descriptions are not very useful in improving the scores of these models.

The results for all of these models can be found in the `results/` folder. The `results.txt` in the root of the `Method 1` directory is the results of all-mpnet-base-v2.

The file `transformers.txt`, `transformers-descriptions.txt`, `sgpt-test.txt`, and `sgpt-descriptions.txt` show the MAP and Precision scores of each model, tested with and without descriptions. The python file `results.py` the results from these files and displays them in a sorted format.

Discussion of answers

Query 3: Insurance Coverage which pays for Long Term Care

```
3 Q0 AP880419-0133 1 0.6684213733318175 runid
3 Q0 AP880920-0017 2 0.6302004190721601 runid
```

```
3 Q0 AP880518-0053 3 0.5883923871349716 runid
3 Q0 AP880329-0114 4 0.5107757647350547 runid
3 Q0 AP880803-0211 5 0.5107499609996304 runid
3 Q0 AP880608-0295 6 0.5061628120068873 runid
3 Q0 AP880523-0108 7 0.501558343091621 runid
3 Q0 AP880609-0016 8 0.48694953690227094 runid
3 Q0 AP880603-0014 9 0.4727823408212035 runid
3 Q0 AP880609-0027 10 0.46170352510225054 runid
```

Of all of the responses, all of them appear to be related to insurance coverage for long term care. Specially, a significant amount of them appear to be related to a healthcare bill passed by congress relating to long term care. Keywords “Medicare” and “Nursing home care” appeared in some of the results, showing the model's ability to identify these keywords as related to “insurance coverage” and “long term care” respectively.

Query 20: The Consequences of Implantation of Silicone Gel Breast Devices

```
20 Q0 AP881122-0171 1 0.8406458392096713 runid
20 Q0 AP881111-0092 2 0.8185698039997388 runid
20 Q0 AP880627-0239 3 0.8175598459326204 runid
20 Q0 AP881110-0035 4 0.7937464412858428 runid
20 Q0 AP881123-0037 5 0.7740555638639333 runid
20 Q0 AP880406-0143 6 0.5692137107252175 runid
20 Q0 AP880705-0255 7 0.45200561999713895 runid
20 Q0 AP880224-0125 8 0.44786202920241625 runid
20 Q0 AP880225-0285 9 0.44757792392244844 runid
20 Q0 AP880224-0220 10 0.42935709760328566 runid
```

Of all the responses,the first 5 were related to silicone breast implants. The first two were about breast implants passing FDA approval despite the consequences, the third were about implants hindering the detection of breast cancer as a consequence, and the last two were about groups seeking either a ban or an in-depth study of implants also detailing consequences of such implants.

The 6th and 7th results were not about gel breast devices specifically, but were about breast cancer surgery and implanting tissue from other parts of the body into the breasts as a method of restoration. These two results are still relatively on topic, as they described surgical procedures done on breasts, of which breast implants are one such procedure.

The rest of the results were about female contraceptive devices. Specifically, these were about initial testing of a female condom.

Method 2: Query vector modification

The aim of this experiment was to implement query vector modification or query expansion based on pretrained word embeddings. Other methods that could be used include adding synonyms to the query if there is similarity with more than one word in the query or with the whole query vector.

This implementation used the word embedding technique word2vec. In particular, it was implemented using the Gensim library. The model was trained in two separate attempts, using the provided collection of

documents and a collection of news headlines posted to Reddit.

The implementation included in this repository is incomplete, but a theoretical approach has been developed.

- The model is trained, either by using the documents provided or by using a corpus comprised of documents provided by Gensim.
- Expansion is done by finding synonyms for each word in a given query and appending the synonymous words to the query.
- Cosine similarities of each document against each query are calculated and ranked using the methods from Assignment 1.

Issues Encountered

The issues encountered when attempting to implement this experiment were primarily related to training the model. Data used to train Word2Vec must be a list of strings and cannot be a dictionary type, which was the implementation used in Assignment 1. Though tokenization was part of the original implementation, it did not properly work with Gensim and its KeyedVectors. A KeyError, specifically an error of a key not being present in the vocabulary, was repeatedly raised. This indicates that there was an issue in how the model was trained, which itself led to an issue with the preparation of the training data. It could also be an issue with

Method 3: Neural models without classical IR systems

The aim of this experiment was to utilize neural models without using a traditional IR system to compute the similarity between the queries and documents.

Setup

After conducting some initial research on the matter, we have decided to utilize the sentence_transformers library to aid us in this task. We found that training and fine-tuning our own models would be too time-consuming. In addition to using the experience gained from assignment 1, I have decided to set up this project using a Python notebook for better organization and convenience throughout the process.

To begin, we have made changes to the Document object, which now saves the vector outputs of the models instead of the tokens obtained from the IR preprocessing process. The functions that were previously used to preprocess documents have been repurposed, and only the portions used to extract documents from the files and convert them into a list of Document objects have been retained. Additionally, the extract_topics() function has been reused to extract topics from the files.

Implementing the models

At the beginning of the assignment, I ran a test on a single model. However, this method proved to be too time-consuming (5 ~ 8 hours), and I needed to find a different strategy. Although the assignment description suggested using a boolean index to restrict calculations to those containing at least 1 query word, due to fear of loss in accuracy, I looked for other ways to reduce the runtime.

After conducting more research, I discovered that the code was only utilizing my CPU to calculate the document vectors, whereas my GPU is generally more suitable for running neural network models. To address this issue, I had to run the pytorch-powered sentence_transformer library with CUDA functionality turned on.

CUDA is a parallel computing platform and programming model developed by NVIDIA that allows for better pure computation performance, such as in machine learning.

To begin, I had to install the CUDA drivers and SDK for my device. I also had to install a specific version of Pytorch that includes CUDA support in my Python environment and enable it in the model.

```
# Example of model with cuda Enabled
model = SentenceTransformer(model_name, device="cuda:0")
```

These results were also shared with my peers in Method one. With this new improvement, most models were able to finish within an hour of running, with most at the 20 ~ 30 minute mark.

The actual running of the code involved entering the text of the documents into the model to create vectors that were then saved into the "extracted_documents" list, which held the Documents object. Similarly, the query vectors were also created using this method, and they were compared using cosine similarity, much like in the first assignment.

Another method utilized to streamline the development process was to save the results from time-consuming runs for future comparison. To achieve this, the list of documents with the vectors created by the model were saved to a CSV file and then compressed using gzip, resulting in a .csv.gz file that saved disk space. However, despite this approach, many files still exceed the 100 MB file size limit of GitHub, and as a result, these files will not be included in the assignment.

Results

- **Sort by Map Scores**

Models	MAP Scores
all-mpnet-base-v2	0.3201
gtr-t5-xl_Results	0.3120
multi-qa-mpnet-base-dot-v1_Results	0.3041
multi-qa-distilbert-cos-v1	0.2733
multi-qa-MiniLM-L6-cos-v1	0.2689
msmarco-distilbert-cos-v5	0.2058
all-distilroberta-v1	0.1800
msmarco-distilbert-base-tas-b	0.1748

- **Sort by P_10 Scores**

Models	P_10 Scores
all-mpnet-base-v2	0.4700
gtr-t5-xl_Results	0.4620
multi-qa-mpnet-base-dot-v1_Results	0.4460
multi-qa-distilbert-cos-v1	0.4000

multi-qa-MiniLM-L6-cos-v1	0.3900
msmarco-distilbert-cos-v5	0.3580
msmarco-distilbert-base-tas-b	0.3260
all-distilroberta-v1	0.2940

- The most successful model was "all-mpnet-base-v2", which achieved a MAP score of 0.3201. Although this is a slight decrease of 1.14% from our A1 score, it represents a significant improvement of 60.22% over the class average MAP score during Assignment 1. In terms of P@10 scoring, it achieved a score of 0.4700, which is 11.9% better than our previous score of 0.4200 and an impressive 80.5% increase over the class average P@10 score in Assignment 1.

- **First 10 answers to Query 3 of "all-mpnet-base-v2"**

```
3 Q0 AP880419-0133 1 0.6621880763567591 runid
3 Q0 AP880920-0017 2 0.6065127139097782 runid
3 Q0 AP880518-0053 3 0.5552188307350332 runid
3 Q0 AP880329-0114 4 0.48668543478621484 runid
3 Q0 AP880803-0211 5 0.4810684422456014 runid
3 Q0 AP880523-0108 6 0.4733738260697369 runid
3 Q0 AP880603-0014 7 0.44558601357060357 runid
3 Q0 AP880512-0164 8 0.4383544679994551 runid
3 Q0 AP880609-0025 9 0.4382644884673459 runid
3 Q0 AP880728-0220 10 0.43508243699817684 runid
```

- In response to Query 3, "Insurance Coverage which pays for Long Term Care", a total of five results were found that were relevant to the topic. One result mentioned long-term but was not related to insurance. Three results mentioned insurance/Medicare but were not related to long-term care. The remaining result was completely unrelated to the topic.

- **First 10 answers to Query 20 of "all-mpnet-base-v2"**

```
20 Q0 AP881122-0171 1 0.7897786050698031 runid
20 Q0 AP881111-0092 2 0.7758698732165796 runid
20 Q0 AP880627-0239 3 0.7364620007639062 runid
20 Q0 AP881110-0035 4 0.689888223337111 runid
20 Q0 AP881123-0037 5 0.6780021726651433 runid
20 Q0 AP880406-0143 6 0.5412892187600634 runid
20 Q0 AP880225-0285 7 0.44065927033821306 runid
20 Q0 AP880224-0220 8 0.4227004785847388 runid
20 Q0 AP880224-0125 9 0.41931252865146984 runid
20 Q0 AP880225-0060 10 0.41577176334216737 runid
```

- In response to Query 20 "The Consequences of Implantation of Silicone Gel Breast Devices", a total of five results were found to be true. The remaining result was completely unrelated to the topic. With four results were related to female contraceptive devices and one result on the birth control device cervical cap.

How to run the Option3.py file

First, follow the instructions from the getting started section.

Prior to running the file, uncomment the model you would like to run from the list:

In this case, the `"multi-qa-MiniLM-L6-cos-v1"` model would be executed.

```
# Variables
# model_name = "all-distilroberta-v1" # will also change the savefile name (csv.gz
files)
# model_name = "all-mpnet-base-v2" # will also change the savefile name (csv.gz
files)
# model_name = "multi-qa-mpnet-base-dot-v1" # will also change the savefile name
(csv.gz files)
# model_name = "msmarco-distilbert-base-tas-b" # will also change the savefile
name (csv.gz files)
# model_name = "multi-qa-distilbert-cos-v1" # will also change the savefile name
(csv.gz files)
# model_name = "msmarco-distilbert-cos-v5" # will also change the savefile name
(csv.gz files)
# model_name = "gtr-t5-xl" # will also change the savefile name (csv.gz files)
model_name = "multi-qa-MiniLM-L6-cos-v1" # will also change the savefile name
(csv.gz files)
extracted_documents = []
```

Finally, in the terminal with the correct python environment with the prerequisite files

To run the program, run the following command:

```
python Option3.py
```

Annex

The scores of the rest of the models from Method 1

Rank	Model	MAP	MAP (w/description)	P@10	P@10 (w/description)
11	sentence-t5-base	0.2468	0.2523	0.3840	0.3880
12	SGPT-1.3B-weightedmean-msmarco-specb-bitfit	0.2369	0.2409	0.3820	0.3920
13	all-MiniLM-L6-v2	0.2357	0.2397	0.3860	0.3880
14	msmarco-distilbert-base-tas-b	0.2067	0.2007	0.3360	0.3360
15	paraphrase-distilroberta-base-v2	0.2003	0.2015	0.3520	0.3580

Rank	Model	MAP	MAP (w/description)	P@10	P@10 (w/description)
16	msmarco-distilbert-cos-v5	0.1872	0.1840	0.3580	0.3500
17	all-distilroberta-v1	0.1803	0.1736	0.3220	0.3060
18	SGPT-125M-weightedmean-msmarco-specb-bitfit	0.1750	0.1750	0.2920	0.2920
19	msmarco-MiniLM-L6-cos-v5	0.1726	0.1657	0.3180	0.3240
20	all-MiniLM-L12-v2	0.1705	0.1721	0.3040	0.3120
21	msmarco-MiniLM-L12-cos-v5	0.1638	0.1583	0.3200	0.3240
22	SGPT-125M-weightedmean-msmarco-specb	0.1613	0.1613	0.3020	0.3020
23	paraphrase-MiniLM-L6-v2	0.1441	0.1469	0.2820	0.2880
24	distiluse-base-multilingual-cased-v2	0.1193	0.1180	0.2420	0.2420
25	paraphrase-MiniLM-L3-v2	0.1132	0.1150	0.2260	0.2220
26	LaBSE	0.1081	0.1038	0.2440	0.2400
27	stsb-mpnet-base-v2	0.0960	0.0934	0.2020	0.2040
28	nli-mpnet-base-v2	0.0950	0.0942	0.2220	0.2260
29	xlm-r-distilroberta-base-paraphrase-v1	0.0937	0.0886	0.2000	0.2040
30	all-roberta-large-v1	0.0823	0.0799	0.1600	0.1600
31	paraphrase-albert-small-v2	0.0698	0.0676	0.1700	0.1720
32	SGPT-125M-weightedmean-msmarco-specb-bitfitwte	0.0640	0.0640	0.1500	0.1500
33	bert-base-nli-mean-tokens	0.0578	0.0620	0.1400	0.1520
34	distilbert-base-nli-stsb-mean-tokens	0.0476	0.0588	0.1520	0.1740
35	distilbert-base-nli-mean-tokens	0.0259	0.0322	0.0760	0.0940
36	SGPT-125M-lasttoken-msmarco-specb	0.0000	0.0000	0.0000	0.0000