A text retrieval/matching system to matching similar questions

In this assignment, I constructed a text retrieval/matching system to matching similar questions. The system is based on two different algorithms: 1 Based on TF-IDF algorithm 2. Based on sentence embedding algorithm. The system based on the two algorithms uses the same data preprocessing method, including removing null values, removing punctuation, converting to lowercase, extracting stem and removing stop words. In TF-IDF algorithm, I calculate the word frequency and inverse document rate of each word, so as to generate the sentence vector in each corpus and the sentence vector of query sentences. In the sentence embedding algorithm, the sentence embedding model is directly used to calculate the sentence vector. Next, the closest result is obtained by comparing the sentence vector of the query sentence with the cosine value of the sentence vector of all sentences in the corpus. Through observation, it is found that the systems based on TF-IDF algorithm and sentence embedding algorithm have good performance.

1. data pre-processing

	id	qid1	qid2	question1	question2	is_duplicate
0	133273	213221	213222.0	How is the life of a math student? Could you d...	Which level of prepration is enough for the ex...	0.0
1	402555	536040	536041.0	How do I control my horny emotions?	How do you control your horniness?	1.0
2	360472	364011	490273.0	What causes stool color to change to yellow?	What can cause stool to come out as little balls?	0.0
3	150662	155721	7256.0	What can one do after MBBS?	What do i do after my MBBS ?	1.0
4	183004	279958	279959.0	Where can I find a power outlet for my laptop ...	Would a second airport in Sydney, Australia be...	0.0
...	...	...	...	...	...	...
363187	57443	100941	83372.0	How do I make money flying my drone?	How can I use a dji phantom to make money	1.0
363188	278260	62873	34460.0	What can you do with an economics degree?	What jobs can you get with an economics degree?	1.0
363189	136211	217377	217378.0	What type of current does a battery produce?	How does a generator work and produce current?	0.0
363190	302720	425744	285638.0	Grammar: What is difference between schedule a...	How do I understand the difference between the...	0.0
363191	126322	39774	20105.0	What is the easiest way to earn money using in...	How can I earn money online easily?	1.0

363192 rows × 6 columns

Firstly, by observing the data and topic requirements, we find that the corpus to be calculated comes from the question1 and question2 columns in the data set. Because it is mentioned in the requirements that only the data listed as 1 in "is_duplicate" will be used as the test, we only keep part of the data as the calculation object.

	level_0	index	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	402555	536040	536041.0	How do I control my horny emotions?	How do you control your horniness?	1.0
1	1	3	150662	155721	7256.0	What can one do after MBBS?	What do i do after my MBBS ?	1.0
2	2	7	106969	147570	787.0	What is the best self help book you have read?...	What are the top self help books I should read?	1.0
3	3	11	233239	71243	177376.0	What will be Hillary Clinton's policy towards ...	What will be Hilary Clinton's policy towards I...	1.0
4	4	13	11568	22332	22333.0	Which is the best book to study TENSOR for gen...	Which is the best book for tensor calculus?	1.0
...	...	...	...	...	...	...	...	...
37021	37021	99994	251230	62891	29406.0	What exactly is open source software?	What is open source software?	1.0
37022	37022	99995	281278	67738	77112.0	Ow can the drive from Edmonton to Auckland be ...	How can the drive from Edmonton to Auckland be...	1.0
37023	37023	99996	315903	211398	121844.0	Why do guys stare a lot at girls?	Why do guys like to stare at girls?	1.0
37024	37024	99997	273368	13889	1183.0	Is backward time travel possible?	Do you think time travel is possible?	1.0
37025	37025	99999	64894	112697	112698.0	Is the human life more important than the life...	Do human lives have any more value than other ...	1.0

Next, I turn all the sentences into lowercase to avoid that the same word may be double counted due to different writing methods. At the same time, I also added the function of removing punctuation to remove some unnecessary interference in the calculation. Finally, I use the stop word removal and stem extraction function in nltk library. The purpose is also to remove some words that have little impact on the result to improve the accuracy of calculation.

```
['can',
 'overcome',
 'depression',
 'homesickness',
 'anxiety',
 'culture',
 'shock'],
['would',
 'happen',
 'every',
 'human',
 'can',
 'read',
 'thoughts',
 'all',
 'human',
 'beings'],
['are',
 'advantages',
```

The final processed data is a list composed of all the data set question2 sentences divided into words.

2. TF-IDF algorithm

In building TF-IDF algorithm, we must first understand the meaning of TF and IDF. TF-IDF is a statistical method used to evaluate the importance of a word to a document set or one of the documents in a corpus. The importance of a word increases in proportion to the number of times it appears in the document, but decreases in inverse proportion to the frequency of its appearance in the corpus. In conclusion, the more a word appears in an article and the less it appears in all documents, the more it can represent the article. TF-IDF is divided into two concepts: TF and IDF. TF represents word frequency, which means the number of times a query keyword appears in the target document. This number is usually normalized (usually the word frequency divided by the total number of words in the article) to prevent it from being biased towards long documents. IDF indicates the prevalence of keywords. If there are fewer documents containing a word, the larger the IDF, indicating that the word has a good ability to distinguish between categories.

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

$$IDF_i = \log \frac{|D|}{1 + |j : t_i \in d_j|}$$

$$TF\text{-}IDF = TF \cdot IDF$$

In the specific implementation process, I use Python language. Build a TF-IDF sparse matrix of word bank length and corpus width through numpy library, count the words and number in each sentence, and finally calculate the TF-IDF value of each word in each sentence. This process will take a lot of time due to the huge size of the database, so it needs to be calculated in advance. In the query, the query statement is also preprocessed to obtain the vector of a query statement, calculate the cosine distance between it and the TF-IDF sparse matrix, and finally get the result of the most relevant query statement after sorting.

3. Sentence embedding algorithm

In the construction of sentence embedding algorithm, I also use Python language. Because the sentence embedding algorithm is an algorithm that can directly transform the sentences in the corpus into a unified scale vector, and there are models that can

be used directly. I use the 'Roberta large NLI stsb mean tokens' model in the nltk library, because the query proves that the model is the best performing and most accurate model of the same type. The disadvantage is that it also requires the largest amount of calculation. Therefore, like TF-IDF algorithm, the construction of sentence embedding algorithm also needs to spend a lot of time in advance to calculate the vector of each sentence. After calculating the vectors of all sentences, we will get a sentence vector matrix of corpus. After transforming the query sentence into vector, we can calculate the distance between each sentence and the query sentence and get the closest sentence.

4. Compare the results of TF-IDF algorithm and sentence embedding algorithm

The system test results based on TF-IDF algorithm are as follows:

```
# test for TF-IDF algorithm
q = "What are the best books on cosmology"
scores = docs_score(q)
d_ids = scores.argsort()[-5:][::-1]
# print("\ntop 5 docs for '{}':\n{}".format(q, [data['question2'][i] for i in d_ids]))
print("top 5 closet sentence for '{}':\n{}".format(q, [Data['question2'][i] for i in d_ids]))
```

```
top 5 closet sentence for 'What are the best books on cosmology':
['Which is the best book for cosmology?', 'Is cosmology a science?', 'Which are the best books to learn C++?', 'Which are the best books to
learn C++?', 'What are best c++ books in 2016?']
```

```
# test for TF-IDF algorithm
q = "What are the coolest Android hacks and tricks you know?"
scores = docs_score(q)
d_ids = scores.argsort()[-5:][::-1]
# print("\ntop 5 docs for '{}':\n{}".format(q, [data['question2'][i] for i in d_ids]))
print("top 5 closet sentence for '{}':\n{}".format(q, [Data['question2'][i] for i in d_ids]))
```

```
top 5 closet sentence for 'What are the coolest Android hacks and tricks you know?':
['What are the best Android hacks?', 'What are the best travel hacks you know?', 'What are the best travel hacks you know?', 'What are some
cool hacks for Android phones?', 'What are some best study hacks?']
```

```
Data['question1'][5]
```

```
'What are the coolest Android hacks and tricks you know?'
```

```
Data['question2'][5]
```

```
'What are some cool hacks for Android phones?'
```

```
# test for TF-IDF algorithm
q = "What are some of the most beautiful houses in the world?"
scores = docs_score(q)
d_ids = scores.argsort()[-5:][::-1]
# print("\ntop 5 docs for '{}':\n{}".format(q, [data['question2'][i] for i in d_ids]))
print("top 5 closet sentence for '{}':\n{}".format(q, [Data['question2'][i] for i in d_ids]))
```

```
top 5 closet sentence for 'What are some of the most beautiful houses in the world?':
['Which are some of the most beautiful houses around the world?', 'Who are some of the most beautiful women in the world?', 'Who are some of
most beautiful women in the world?', 'Who are some of most beautiful women in the world?', 'Who are some of most beautiful women in the worl
d?']
```

```
Data['question1'][15]
```

```
'What are some of the most beautiful houses in the world?'
```

```
Data['question2'][15]
```

```
'Which are some of the most beautiful houses around the world?'
```

```
# test for TF-IDF algorithm
q = "What are the best websites for entrepreneurs?"
scores = docs_score(q)
d_ids = scores.argsort()[-5:][::-1]
# print("\ntop 5 docs for '{}':\n{}".format(q, [data['question2'][i] for i in d_ids]))
print("top 5 closet sentence for '{}':\n{}".format(q,[Data['question2'][i] for i in d_ids]))
```

```
top 5 closet sentence for 'What are the best websites for entrepreneurs?':
['What do entrepreneurs do?', 'What are some good movies for entrepreneurs?', 'Which books should entrepreneurs read?', 'What are some books
all entrepreneurs must read?', 'What are some books all entrepreneurs must read?']
```

```
Data['question1'][20]
```

```
'What are the best websites for entrepreneurs?'
```

```
Data['question2'][20]
```

```
'What are the best websites for entrepreneur?'
```

```
# test for TF-IDF algorithm
q = "What the best science documentaries?"
scores = docs_score(q)
d_ids = scores.argsort()[-5:][::-1]
# print("\ntop 5 docs for '{}':\n{}".format(q, [data['question2'][i] for i in d_ids]))
print("top 5 closet sentence for '{}':\n{}".format(q,[Data['question2'][i] for i in d_ids]))
```

```
top 5 closet sentence for 'What the best science documentaries?':
['What are some good science documentaries?', 'What are the best documentaries about World War 2?', 'What are your favourite documentaries?
Why?', 'What are your favourite documentaries? Why?', 'What are the best World War II documentaries?']
```

```
Data['question1'][25]
```

```
'What the best science documentaries?'
```

```
Data['question2'][25]
```

```
'What are some good science documentaries?'
```

The system test results based on sentence embedding algorithm are as follows:

```
queries=['What exactly is open source software?']
```

```
query_embeddings = model.encode(queries)
```

```
import scipy
#test for sentence embedding algorithm
print("Search Results")
number_top_matches = 5
for query, query_embedding in zip(queries, query_embeddings):
    distances = scipy.spatial.distance.cdist([query_embedding], sentence_embeddings, "cosine")[0]
    results = zip(range(len(distances)), distances)
    results = sorted(results, key=lambda x: x[1])
    print("Query:", query)
    print("\nTop {} most similar sentences in corpus:".format(number_top_matches))

    for idx, distance in results[0:number_top_matches]:
        print(dataset[idx])
```

```
Search Results
Query: What exactly is open source software?

Top 5 most similar sentences in corpus:
what is open source software
what are some examples of open source application software
how can one create software
what is the scope of software engineering
what are some examples of software
```

```
queries=['What are the best books on cosmology']
```

```
query_embeddings = model.encode(queries)
```

```
import scipy
#test for sentence embedding algorithm
print("Search Results")
number_top_matches = 5
for query, query_embedding in zip(queries, query_embeddings):
    distances = scipy.spatial.distance.cdist([query_embedding], sentence_embeddings, "cosine")[0]
    results = zip(range(len(distances)), distances)
    results = sorted(results, key=lambda x: x[1])
    print("Query:", query)
    print("\nTop {} most similar sentences in corpus:".format(number_top_matches))

    for idx, distance in results[0:number_top_matches]:
        print(dataset[idx])
```

```
Search Results
Query: What are the best books on cosmology

Top 5 most similar sentences in corpus:
which is the best book for cosmology
which book is best book for physics
which book is best book for physics
what are the best books to learn astronomy for beginners
why do people like science fiction
```

```
queries=['What the best science documentaries?']
```

```
query_embeddings = model.encode(queries)
```

```
import scipy
#test for sentence embedding algorithm
print("Search Results")
number_top_matches = 5
for query, query_embedding in zip(queries, query_embeddings):
    distances = scipy.spatial.distance.cdist([query_embedding], sentence_embeddings, "cosine")[0]
    results = zip(range(len(distances)), distances)
    results = sorted(results, key=lambda x: x[1])
    print("Query:", query)
    print("\nTop {} most similar sentences in corpus:".format(number_top_matches))

    for idx, distance in results[0:number_top_matches]:
        print(dataset[idx])
```

```
Search Results
Query: What the best science documentaries?

Top 5 most similar sentences in corpus:
what are some good science documentaries
what are the top science fiction movies ever
what are some really cool science facts
what are the best scifi tv shows
what are your favourite documentaries why
```

```
queries=['What are the best websites for entrepreneurs?']
```

```
query_embeddings = model.encode(queries)
```

```
import scipy
#test for sentence embedding algorithm
print("Search Results")
number_top_matches = 5
for query, query_embedding in zip(queries, query_embeddings):
    distances = scipy.spatial.distance.cdist([query_embedding], sentence_embeddings, "cosine")[0]
    results = zip(range(len(distances)), distances)
    results = sorted(results, key=lambda x: x[1])
    print("Query:", query)
    print("\nTop {} most similar sentences in corpus:".format(number_top_matches))

    for idx, distance in results[0:number_top_matches]:
        print(dataset[idx])
```

```
Search Results
Query: What are the best websites for entrepreneurs?

Top 5 most similar sentences in corpus:
what are the best websites for entrepreneur
what are the best websites in online jobs
what is the best online website builder to create a new web page for a business
what is the best online website builder to create a new web page for a business
who is the best website broker
```

By comparing the five groups of results, it is found that the system based on TF-IDF algorithm has four groups of correct answers, while the system based on sentence embedding algorithm has all the correct answers.

A1808830 Zhengjie He
2022/4/17