ASSESSMENT COVER SHEET

Assignment Details

Course: Applied Natural Language Processing

Semester/Academic Year: Semester 1/ 2022

Assignment title: Assignment 2: Building a text matching algorithm for question retrieval

Assessment Criteria

Assessment Criteria are included in the Assignment Descriptions that are published on each course's web site.

Plagiarism and Collusion

Plagiarism: using another person's ideas, designs, words or works without appropriate acknowledgement.

Collusion: another person assisting in the production of an assessment submission without the express requirement, or consent or knowledge of the assessor.

Consequences of Plagiarism and Collusion

The penalties associated with plagiarism and collusion are designed to impose sanctions on offenders that reflect the seriousness of the University's commitment to academic integrity. Penalties may include: the requirement to revise and resubmit assessment work, receiving a result of zero for the assessment work, failing the course, expulsion and/or receiving a financial penalty.

DECLARATION

I declare that all material in this assessment is my own work except where there is clear acknowledgement and reference to the work of others. I have read the University Policy Statement on Plagiarism, Collusion and Related Forms of Cheating:

http://www.adelaide.edu.au/policies/?230

I give permission for my assessment work to be reproduced and submitted to academic staff for the purposes of assessment and to be copied, submitted and retained in a form suitable for electronic checking of plagiarism.

HN 21-05-2022

SIGNATURE AND DATE

Applied Natural Language Processing (COMP SCI 7417)
Assignment 2 Report

Cheuk Hang Ng (a1821087)

University of Adelaide, a1821087@student.adelaide.edu.au

1 INTRODUCTION

This assignment aims to write a code in Python for building a text retrieval and matching system to match similar questions based on both TF-IDF (with inverted file) and word embeddings. The code was written and run on Jupyter Notebook Python environment. In addition, a user interface was also implemented which accepts a query question as user input and returns 5 most similar questions in the dataset. The dataset is a tsv file provided by the course, which consists of columns “id”, “qid1”, “qid2”, “question 1”, “question 2”, “is_duplicate”. The dataset indicates whether each question 1 has a similar question by denoting “is_duplicate” as 1, and the corresponding similar question is the question 2 in the same row, which we denoted as the ground truth throughout the assignment. The dataset was first read as a dataframe and investigated. Some manipulations were done before using as the input of different algorithms. For example, duplicated and null rows were removed, and the columns’ type were modified. All questions under the column “question2” were used as the base for building the sentence representations with TF-IDF and word embeddings. The performances of three methods, namely TF-IDF based similarity search, average weighted sentence embedding and IDF score weighted sentence embedding were investigated by the probability that the ground truth was ranked in the top 2 or top 5 most similar questions for the first 100 question 1 which marked as “is_duplicate” as the queries.

2 DESCRIPTIONS OF DETAILS

2.1 Text Pre-processing

Numbers of text pre-processing techniques were used in the implementation, which include a general text pre-processing, stopwords removal and text normalization. In this assignment, lemmatization with part-of-speech (POS) tagging was applied in the text normalization process.

The dataset was first read as a dataframe object, duplicated or null rows were dropped. The “index” column after resetting index was also dropped. The columns’ type of “qid1” and “qid2” were set as “int32” for convenience, and the columns’ type of “question1” and “question2” were set as “string” for easy access. The modified dataframe were copied and text pre-processing were performed on the columns “question1” and “question2”. The texts were first lowercased, and numbers and punctuations were also removed. Tokenization and stopwords removal were done immediately after the above text pre-processes. The texts were then tagged with part-of-speech (POS) before being lemmatized. The POS tagging process marks each word with a particular part of speech, which based on the definition or the context of the word. We used a function to slightly modify the outputs from the nltk.pos_tag, where the tags were reduced to five types of tags (wordnet.ADJ/NOUN/VERB/ADV or None) that the lemmatizer would accept. The POS tagged words were then

lemmatized where the lemmatizer reduced words to root words. WordNetLemmatizer fom nltk (nltk.stem.WordNetLemmatizer) were used for lemmatization.

2.2 Building Sentence Representations

2.2.1 TF-IDF based sentence representation

TF-IDF is a weighting scheme where it increases the weight of important words and decreases the weight of irrelevant words in a document. The idea behind is that a word is important to a document if it occurs frequently in that document but is rare in other documents. The TF-IDF score of a word is the product of the TF score and the IDF score of that word. The definition of TF-IDF score we used is as follows:

$$TF - IDF_{w,d} = TF_{w,d} \times IDF_w$$

, where

$$TF_{w,d} = \begin{cases} log_{10}\big(1 + count(w,d)\big), if\ count(w,d) > 0 \\ 0 \quad , \quad otherwise \end{cases}$$

with $count(w,d)$ is the number of occurrence of a word w in a document d, and

$$IDF_w = \frac{N}{df_w}$$

, where N is the total number of document in the dataset and df_w is the number of document that contains word w.

An inverted file was introduced before we built the sentence representations with the TF-IDF score. The inverted file was a nested dictionary object where the keys were the words in all questions 2, and the values for each key stored the pair of question ID and the corresponding TF-IDF of that word. Each query input was then represented by some accumulated scores corresponding to each document retrieved from the inverted file . In particular, the system looked up the inverted file for every word in each query and recorded the accumulated TF-IDF scores for each document in descending order, simply due to the fact that a document is the most similar to a specific sentence if it has the highest accumulated TF-IDF score for that sentence.

2.2.2 Average Weighted Sentence embedding with pre-trained GloVe model

The model used for the sentence embedding method was a pre-trained word embedding model, downloaded from the link provided in the assignment instructions. The file used in this project was the model that trained with 6 billion tokens, 400 thousands vocabulary, uncased, 50 dimensions one, which was first read as a dictionary object. The sentence representation of each question and query was built with the follow logic: Every word in the sentence was looked up in the model dictionary, and the vector representation for that word was returned if the word could be found in the dictionary, otherwise a zero vector was returned. Then we took the average of these vectors to represent the sentence. A pairwise comparison between a query and all questions in the dataset was conducted and cosine similarity was used to evaluate how similar the query and a question

was. The cosine similarity between two vectors v_1 and v_2 is computed by dividing the dot product of the two vectors by the product of their norm, which is:

$$\frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}$$

The questions were then ranked by the cosine similarity in descending order for each query.

2.2.3 IDF Score Weighted Sentence embedding with pre-trained GloVe model

A sentence embedding calculated by an alternative strategy was also introduced. The sentence representation of a query/ question was deduced in a similar way. Every word in the sentence was looked up in the model dictionary, and the vector representation for that word was returned if the word could be found in the dictionary, however a vector of all ones was returned if the word does not exist in the model dictionary. Then instead of taking the average of these vectors to represent the sentence, we sum the products of each word vector and the logarithm of its IDF score or $log_{10}(N)$, depends on whether the word was in any of the questions 2 or not, and then divide the sum with the total IDF score to represent that sentence, which is:

$$v_{sent} = \frac{\Sigma_{w \in sent} w \times IDF'_w}{\Sigma_{w \in sent} IDF'_w}, \qquad where\ IDF'_w = \begin{cases} log_{10}\left(\frac{N}{df_w}\right), if\ w\ in\ any\ of\ the\ questions\ 2 \\ log_{10}(N), \qquad otherwise \end{cases}$$

A pairwise comparison between a query and all questions in the dataset was also conducted and cosine similarity was used again to evaluate how similar the query and a question was, where the computation is the same to last section, and the questions were also ranked by the cosine similarity in descending order for each query.

3 DESCRIPTION OF EVALUATION METHOD

The evaluation for all methods was the same. We calculate the probabilities of the ground truth being ranked in the top 2 and top 5 similar questions by the three methods and use the probability to evaluate each method.

4 RESULTS AND COMPARISON

The runtimes and metrics of the methods are compared, and a user interface was implemented to test the sentence matching system with average weighted sentence embedding. In addition, the vector representations from the average weighted and the IDF score weighted sentence embedding models are also compared.

The average weighted sentence embedding method outperformed the TF/IDF based sentence matching system. The probabilities of obtaining the ground truth in the top 2 and top 5 similar questions by the average weighted sentence embedding method were 0.4 and 0.5, while the probabilities from the TF/IDF sentence matching system were 0.23 and 0.35, which is roughly 1.5 times better. However, as the average weighted sentence embedding method used the pre-trained model, the runtime was near 7 minutes, which was significantly longer than the 6 seconds runtime from the TF/IDF based method. While the IDF score weighted sentence embedding method did not remarkably improve the performance of the average weighted method,

having the probabilities of 0.36 and 0.51 for the ground truth being ranked in top 2 and top 5 respectively, and a runtime of slightly more than 6 minutes.

TF/IDF based

```
start = timeit.default_timer()

#calculate the probability of the ground truth being in the top 2 or 5 most similar
tf_idf_top_2_probi, tf_idf_top_5_probi = metrics(find_tf_idf(question_2, queries_dict), ground_truth)
```

```
print(tf_idf_top_2_probi, tf_idf_top_5_probi)

stop = timeit.default_timer()
print('Time: ', stop - start)
```

```
0.23 0.35
Time:  6.358554899998126
```

Average weighted sentence embedding

```
start = timeit.default_timer()

#calculate the probability of the ground truth being in the top 2 or 5 most similar
avgweight_top_2_probi, avgweight_top_5_probi = metrics(avgweight_sent_embedding(question_2, queries_dict), ground_truth)
```

```
print(avgweight_top_2_probi, avgweight_top_5_probi)

stop = timeit.default_timer()
print('Time: ', stop - start)
```

```
0.4 0.5
Time:  405.42531570000574
```

Weighted with term IDF sentence embedding

```
start = timeit.default_timer()

idfweight_top_2_probi, idfweight_top_5_probi = metrics(idfweight_sent_embedding(question_2, queries_dict), ground_truth)
```

```
print(idfweight_top_2_probi, idfweight_top_5_probi)

stop = timeit.default_timer()
print('Time: ', stop - start)
```

```
0.36 0.51
Time:  382.4497138000006
```

We have obtained the vector representations of first 20 questions 2 in the dataset and the first 5 queries from the two weighted sentence embedding methods. Some of the vectors are being for side-to-side compared below. The first two images are the vector representations of the first 2 questions 2 from the average weighted method and the IDF score weighted method respectively, and the last two images are the vector representations of the last 2 queries. We observed that the vector representations from the IDF score weighted sentence embedding are closer in each dimension than the average weighted one.

```
[-0.12911     0.248468    0.129958   -0.36434     0.070104   -0.1778524
  0.186436    0.13053     0.1536316   0.0998572   0.0351416  -0.235438
  0.113802    0.003397    0.1260558   0.102996    0.031054    0.0126578
 -0.178254   -0.38702296  0.0337446   0.110548    0.24546    -0.068448
  0.076268   -0.74305    -0.09686     0.027428   -0.136308   -0.019124
  1.8604      0.302708   -0.138314   -0.485212    0.378984    0.174252
  0.311538    0.43445     0.266024   -0.456542   -0.1284708  -0.203598
  0.09763     0.500448   -0.222418   -0.20583     0.490518    0.381082
  0.208438    0.0678318 ]
[ 0.349345  -0.55255    0.289645   0.10384   -0.149485   0.013528
  0.16477   -0.094675   0.23866   -0.259645   0.319285   0.006904
 -0.26953   -0.0143445 -0.42391    0.408185   0.039643   0.0237745
 -0.0422155 -0.09584   -0.098895  -0.10199   -0.11886   -0.179825
 -0.14502   -0.9998    -0.044797  -0.22443    0.27383    0.2669
  1.8182     0.07165   -0.443585  -0.31221   -0.155905   0.14926
 -0.11482    0.006406   0.045145   0.177615  -0.0310535  0.192105
  0.045593   0.19074   -0.30006    0.0158115 -0.042553  -0.114195
 -0.018606  -0.094325 ]
```

```
[0.54263584 0.76361309 0.70943241 0.37395454 0.67995835 0.49009904
 0.7241952  0.69453419 0.69532611 0.68698935 0.62818441 0.47379579
 0.6707334  0.60516113 0.6915199  0.6869696  0.66502858 0.62550773
 0.49074163 0.33152332 0.62071195 0.67286348 0.77044719 0.55525883
 0.65874442 0.11131941 0.55340583 0.64407735 0.56344138 0.61059932
 1.85741057 0.81553811 0.53753367 0.29749513 0.83963319 0.71804356
 0.77909737 0.88530405 0.76504132 0.30589103 0.52606798 0.48782299
 0.67962786 0.90943654 0.48080611 0.4986744  0.89368005 0.84338339
 0.72366842 0.65921291]
[0.91629302 0.41518182 0.88312249 0.77988545 0.63913293 0.72970627
 0.8137394  0.66958648 0.85479418 0.5779258  0.89959107 0.72602584
 0.57243349 0.71421974 0.48665682 0.94898572 0.74421629 0.73539943
 0.69873405 0.66893918 0.66724176 0.66552212 0.6561488  0.62227541
 0.64161378 0.16668063 0.69729971 0.59749197 0.87433535 0.8704849
 1.73241872 0.76200003 0.47572499 0.54871961 0.63556585 0.80512173
 0.65839351 0.72574914 0.74727332 0.82087634 0.70493588 0.82892728
 0.74752223 0.82816886 0.5554704  0.73097503 0.69854652 0.65874077
 0.71185196 0.66978095]
```

```
[ 0.00691429  0.30003614 -0.01740143 -0.17283957  0.24237529  0.17781286
 -0.09236886 -0.15327214 -0.18024729 -0.33288    -0.49912057  0.07104
 -0.287199   -0.13505314 -0.10166614  0.29982571  0.07314857 -0.02829644
  0.28270143  0.0276713   0.18272857  0.25627714  0.03367729 -0.30586486
  0.60507743 -2.06774286  0.26769143  0.12122414 -0.28547143  0.36545146
  2.44776857  0.28676543 -0.62934857 -0.52032571 -0.36338143 -0.60215214
 -0.49576571  0.22459143 -0.704864   -0.471984   -0.53256     0.21811429
 -0.19614071 -0.35049671 -0.123183    0.15662143 -0.32992814  0.15146857
  0.077027    0.153929  ]
[-0.05109813  0.28877829 -0.44918286  0.09448143  0.68577886  0.11567559
  0.00610714 -0.90452286  0.16258    -0.01012857  0.59606286  0.48682857
 -0.43783571 -0.08822     0.41779457 -0.08457071 -0.00362143 -0.12539286
 -0.16863557 -0.57780429  0.08591843 -0.17156714  0.13429741 -0.27777
  0.10865    -0.64660857 -0.93248786 -0.41155    -0.02887429  0.32501286
  2.30769286 -0.32954286 -0.50994665 -0.76942571 -0.02233627 -0.16788714
 -0.02335057  0.44641657  0.252614    0.52696429  0.24949786 -0.02599286
 -0.19607207  0.34663929 -0.37060057  0.05995257  0.50633143  0.455331
 -0.14231457  0.3698307 ]
[ 3.80046526e-02  3.64602643e-01 -5.69580117e-02 -2.99793359e-01
  2.89275003e-01  2.13654833e-01 -5.95035879e-02 -1.21632264e-01
 -1.95296811e-01 -2.67213386e-01 -4.43249676e-01  1.08173379e-01
 -2.80931448e-01 -5.66585243e-02 -1.23608087e-01  2.79954152e-01
  8.44030084e-02 -5.47409836e-02  3.40357240e-01 -9.76105158e-04
  5.05085560e-02  1.96679948e-01 -3.48506196e-02 -2.30259846e-01
  5.80569522e-01 -2.01588105e+00  2.76364570e-01  2.72571214e-01
 -5.95208231e-02  4.00497570e-01  2.44570355e+00  1.98774585e-01
 -5.76001714e-01 -6.47273986e-01 -4.27979696e-01 -5.89194211e-01
 -4.89997065e-01  2.36989008e-01 -6.37417087e-01 -5.34195190e-01
 -4.72462084e-01  1.49878621e-01 -1.94830229e-01 -2.88732866e-01
 -1.76592398e-01  1.83807953e-02 -2.10802034e-01  5.17337997e-02
  1.21598503e-01  3.17622193e-01]
[ 6.58139439e-02  1.03531583e-01 -4.60687274e-01  2.88465972e-01
  8.32472804e-01  1.02691229e-01  4.96979489e-01 -9.98957218e-01
  1.49546689e-01 -1.18954085e-01  1.11250066e+00  5.10265481e-01
 -5.38750541e-01 -1.77958001e-01  3.09468016e-01  1.17706160e-01
 -5.95598219e-02 -1.46543713e-01 -1.31397614e-01 -8.37460460e-01
 -4.56965988e-02 -4.90118652e-01  5.38608485e-02 -4.41978001e-01
 -9.41094890e-02 -2.12107175e-01 -8.28907377e-01 -2.45607061e-01
  1.48099275e-01  5.86889994e-01  1.75529634e+00 -4.28281273e-01
 -6.67116192e-01 -8.50609247e-01 -1.33293535e-01 -2.88678511e-01
 -1.14626725e-03  3.76630040e-01  4.17261044e-01  1.06891081e+00
  4.27595111e-01 -7.21877746e-04 -4.06379547e-01  5.31736774e-01
 -5.19845385e-01 -1.02291827e-01  6.96039073e-01  5.19572603e-01
 -1.84251740e-01  2.60435041e-01]
```