The University of Adelaide, School of Computer Science

# *Applied Natural Language Processing*

Semester 1, 2022 Assignment 2: Building a text matching system for question matching

DUE: 22 May. 2022, Sunday 11:55 PM

## Submission

Instructions and submission guidelines:

• You must sign an assessment declaration coversheet to submit with your assignment.

• Submit your assignment via the Canvas MyUni.

## Task

You are required to write code for building a text retrieval/matching system to matching similar questions. This system could be used for online question forums. You need to build this system based on both TF-IDF (with inverted file) and word embeddings. You are encouraged to explore different ways of creating sentence representations from the pretrained word embedding. Examples include directly averaging word embedding, downweight frequent words, or using the method in

Sanjeev Arora, Yingyu Liang, Tengyu Ma, A SIMPLE BUT TOUGH-TO-BEAT BASELINE FOR SENTENCE EMBEDDINGS, ICLR 2017

Specifically, the task included:

- 1. Write code for data loading and text processing 5%
- 2. Write code for building inverted index with TF-IDF and performing text matching 25%
- 3. Write code for building text matching with sentence embedding. The sentence embedding is calculated by averaging word embedding. 10%
- 4. Write code for building text matching with sentence embedding. The sentence embedding is calculated by using an alternative strategy other than averaging word embedding, e.g., downweight frequent words, or using the method in Sanjeev Arora, Yingyu Liang, Tengyu Ma, A SIMPLE BUT TOUGH-TO-BEAT BASELINE FOR SENTENCE EMBEDDINGS, ICLR 2017. 10%

Rubric for Coding part

| | | | | |
|---|---|---|---|---|
| Tasks 1-3 | The implementation has major error, e.g., misunderstanding of key concepts: 30% of total marks | The implementation is partially correct, and has minor mistakes. Depending on the type of mistaken, you can earn 30%-90% of the total mark | Correct implementation 100% of the total mark | |
| Task 4 | The implementation has major error, e.g., misunderstanding of key concepts: 30% of total marks | Implement one sentence embedding approach that is different from average embedding. The implementation is partially correct and has minor mistake. you can earn 30%-70% of the total mark | | |
| | | | | |

.

**Programming requirement**

You will use Python to write the program. Third party packages are allowed in the following scenarios: (1) read file (2) text preprocessing (3) get word count statistics (4) SVD calculation or other matrix operations.

You need to write code without using the third party packages for (1) calculating TF-IDF (2) building Inverted index for retrieval and search with inverted index. (3) calculating sentence embedding from word embeddings. (4) calculating the sentence similarity

Please directly use the pretrained word embedding, e.g., https://nlp.stanford.edu/projects/glove/ , instead of doing training by yourself.

You are also required to submit a report (<10 pages in PDF format), which should have the following sections (report contributes 50% to the mark; code 50%):

- A description of how to build sentence representations for both methods (25%)
- Comparing the TF-IDF based and sentence embedding based sentence matching system. 15%
- Comparing different ways of creating sentence embedding. 10%

In summary, you need to submit (1) the code and (2) a report in PDF.

## Data and Evaluation

The dataset is provided at MyUni. The dataset is a tsv file with following columns:

Id, qid1, qid2, question 1, question 2, is_duplicate

Is_duplicate indicates whether question 1 is similar to question 2.

We will use questions in question 1 as queries and find matched questions in question 2. Specifically, we will use the first 100 questions with is_duplicate = 1 to form 100 queries. Each of those queries will be matched against all the questions in question 2. Then we know the ground-truth match for each query is their associated question 2. For example, if you have

1, q1_1, q2_1, is_duplicate = 1

2. q1_2, q2_2, is_duplicate = 0

3, q1_3, q2_3, is_duplicate = 1

4, q1_4, q2_4, is_duplicate = 0

Then you will use q1_1 and q1_3 as queries. Q1_1 will be matched against q2_1, q2_2, q2_3, q2_4,…. The ground-truth match for q1_1 is q2_1.

The system will be tested by using the following protocol: use each query question as the query and run the system to return a list of questions ranked by their relevance to the query. Check whether the ground-truth answer is ranked among the top 5. Using the probability that ground-truth ranked into top 2 and top 5 as two metrics for the retrieval system.

Your code should support the following interface

>> Python SearchQuestion “your question”

Then the top 5 matched questions will be returned.

## Hint

1. You could try the following command to read the tsv file
   Data = pd.read_csv(‘data.tsv’, sep=’\t’,error_bad_lines=False)
2. You may encounter duplicated questions. In that case, you can simply remove the duplicated questions