Krannert School of Management

Spring 2021

Individual Assignment 1

Total Points: 10

There are 100 reviews for restaurants and films in a collection under the attached csv file on Brighspace. Each review is treated as a text document. In this assignment, you are required to process those documents such that each of them will be finally numerically represented. In particular, please follow the steps listed below:

Part 1. Text representation (total 5 points)

1. Tokenize each review in the collection.
2. Use the tokenized reviews after step 1, lemmatize all the words, convert in lowercase.
3. Based on the output in step 2, remove all the stop-words and the punctuations.
4. Based on the output in step 3, convert each of the reviews in a TD-IDF vector. The minimal document frequency for each term is 3. Also, include 2-gram.
5. Based on the output in step 1, POS-tag each word and do a TD-IDF vectorization, the minimal document frequency for each term is 4 (please don't do normalization and stop-word removal)

   Tip: you may consider using a "for loop" for step 1 to step 3, so you could process the whole collection at once.

Please submit two files:

1. A word file includes python code with your comment #, and one screenshot on your PyCharm showing that your code has run through successfully for **each of the five steps** (5 screenshots in total). Also, **report** the #dimension of the vectors of step 4 and step 5 at the end of the word file.
2. A CSV file saves your final TF-IDF vectors (step 4). Each review is corresponding to one

row, each column is corresponding to one item in the vectors. (Note: you don't need to submit the intermediate output data in step 1, step 2 and step 3).

3. A CSV file saves your POS-tag TF-IDF vectors (step 5). Each review is corresponding to one row, each column is corresponding to one item in the vectors. (Note: you don't need to submit the intermediate output data in step 1).

Part 2. Word embedding (total 5 points)

1. Choose the first 10 tokenized documents in the data obtained after step 1 in Part 1, use index-based encoding to encode each word and represent each text document in a vector(list) of indices (in integer), save the representation of the whole collection as a 2D array (i.e., a matrix).
2. Based on the output of step 1(in Part 2), use one-hot encoding for each index to further represent each text document as a one-hot 2D array, save the representation of the whole collection as a 3D array (i.e., a cube).
3. Choose the first 10 tokenized documents in the data obtained after step 1 in Part 1, use the pre-trained GloVe model 'glove.6B.50d' (i.e., Wikipedia 2014 + Gigaword 5, 50d output) to embed each word as a 50d vector. Represent each text document as a 2D array, save the representation of the whole collection as a 3D array (i.e., a cube).

Please submit two files:

4. A word file includes python code with your comment #, and one screenshot on your PyCharm showing that your code has run through successfully for **each of the three steps** (3 screenshots in total). Also, **report** the #dimension of the represented collection of step 1, 2, and 3 at the end of the word file.
5. A CSV file saves your final index-encoded 2D array (step 1).
6. A CSV file saves your one-hot encoded 3D array (step 2).
7. A CSV file saves your GloVe encoded 3D array (step 3).