CSCI 517 (Natural Language Processing)

Real-World Text Representation Learning - Homework 3

<u>Tasks/Questions with (***) Prefix are only for Graduate Students</u>

In this homework, you will train a real-world word2vec representation embedding. While TF-IDF can be used for both word and sentence or document representation learning, *<u>we will need to do a little bit more work to use word2vec for document representation.</u>*

You are provided with a few resources: (1) the template Jupyter notebook code on how to train word2vec from scratch with some # TODO tasks. The template code will also contain the evaluation code for your convenience, (2) training dataset CSV file to be loaded by the template code and (3) validation dataset CSV file to be used to fine-tune your word2vec.

Task 1: Train a word2vec representation embeddings on a large news corpus (>1B tokens) using the template code. This template code follows what we have done in class. You are welcome to use resources presented in class to finish all the #TODO and train your word2vec. *<u>Since the corpus is large, please be patient during training and be smart on your code management (you don't want to run lengthen process twice!) You will need to re-set all the configurations according to your understanding. Those include the desirable size of vocabulary (n_vocab), the desirable size of embeddings (n_embed), the learning rate, how many negative examples to sample per positive example, window size, etc.</u>*

Task 2: Since word2vec is a word-based representation learning, we first need to come up with a **Sentence Aggregation Method** to represent a document of several words. To do this, there are two schemas that we can use. (1) we calculate the average vector of all the representation vectors of words within a sentence as the sentence vector. (2) we concatenate all the representation vectors of words within a sentence as the sentence vector (if a word vector has size 10 and a sentence has 5 words then the sentence vector should have size 5*10=50).

Task 3: Using the sentence representation vector that you came up with from Task 2 for final evaluation. First, you will need to code the transform() function. Like transform() function in Homework 2, this function inputs a query or a document and outputs its representation vector using ONE of the two techniques in Task 2. The evaluation code will utilize your transform() function for evaluation. This code will calculate the similarity between pairs of sentences that have very similar meaning and calculate the average similarity score for all pairs. Report the average similarity score for EACH sentence aggregation methods from Task 2. *<u>(The higher the better)</u>*

(BONUS) Task 4: Your task is to optimize your word2vec performance on the validation set already loaded in the evaluation code by either changing the current codes or to change the training configurations in Task 1. Once the homework due is over, we will test your code on a public test set (similar with Homework 2-Bonus). Students with the best result on this test set will get bonus points and a small prize.