University of North Texas
CSCE5290/CSCE4290, Fall 2020
Natural Language Processing
Instructor: Dr. Eduardo Blanco

Homework#5 Perceptron and embeddings
Due date: November 24, 2020
Name:

Question 1 [75pt]
Implement the gradient descent algorithm that we discussed in class to train a sigmoid unit. Your implementation only needs to handle binary classification tasks (each instance will have class 0 or 1). In addition, you may assume that all attributes have binary values (either 0 or 1).

When applying the trained sigmoid unit to a test instance, use 0.5 as the classification threshold (i.e., classify the instance as 1 if the unit outputs a value that is greater or equal than 0.5, otherwise classify the instance as 0). Initialize all the weights to 0.

- A train (`train.data`) and test file (`test.data`) are provided in the course website (`data/` folder).
- Use the starter code in Python.
- Your program must take 4 arguments as input: (1) training file, (2) test file, (3) learning rate and (4) number of epochs. Example:

```
$ python perceptron.py data/train.dat data/test.dat 0.01 10
```

- Follow these steps:
  - Train the sigmoid unit using 0.01 learning rate and 10 epochs.
  - Use the perceptron you just trained to classify the test instances and print the accuracy on the test set.
  - Experiment with different learning rates and epochs, and compare the accuracies obtained. Specifically, try the following learning rates: 0.005, 0.01, 0.025, 0.05, and 0.1, and number of epochs: 1, 2, 3, 4, 5, 10, 15, 25, 50, and 100 (total number of combination: 50). Which combination of learning rate and epochs yields the best results? Briefly justify the accuracies you get. You can run all the experiments using a single command:

```
for lr in 0.005 0.01 0.025 0.05 0.1
do
  for e in 1 2 3 4 5 10 15 25 50 100
  do
    echo "$lr $e "
     python3.7 code/perceptron_sol.py data/train.dat data/test.dat $lr $e
     echo ""
  done
done
```

Question 2 [50pt]
Note: This questions reuses materials from `https://github.com/oxford-cs-deepnlp-2017/practical-1/blob/master/practical1.ipynb`

You are given a Jupyter notebook for this question. Follow these steps to run the notebook (as long as you can run it, you will be fine, even if you do not follow these steps):

1. Download and install Anaconda.[1] In Linux, you have to download a file (choose Python 2.7 or 3.6, whichever you have in your machine) and run a simple command.

[1] `https://conda.io/docs/user-guide/install/index.html#regular-installation`

2. The file `homework5.env` lists all the packages required to run the homework (and some that you don't really need). Create an conda environment with the right packages and versions:

   ```
   conda create --name hw5 --file homework5.env
   ```

3. Activate the environment:

   ```
   source activate hw5
   ```

4. Run the homework (it will open in your browser):

   ```
   jupyter notebook word\_embeddings.ipynb
   ```

5. You can now run each cell using `Shift + Enter`

It should be straightforward to complete the notebook (most of it is already implemented for you, but you do have to write some code). You are asked to complete the notebook and write a short report discussing the following:

1. Regarding the TED and Wiki datasets:
   (a) What are the most frequent words in each dataset? List the top 20 most frequent words and draw the histograms.
   (b) Get the most similar words to *man* using the embeddings you got using each corpus.
      i. Why are they different?
      ii. Can you identity any relationship(s) between *man* and its most similar words beyond "they are similar"? Repeat the same exercise for the word *woman*.
   (c) Using `model_ted.wv.most_similar(positive=['nice', 'man'])` you can get the most similar words to "nice plus man". Does the result make sense to you? What about the result of `model_ted.wv.most_similar(positive=['nice', 'woman'])`?
   (d) Identify at least 5 clusters in the t-SNE visualization that appear to make sense (Copy and paste the cluster, and explain why it makes sense in a couple sentences). Compare the visualizations with both datasets.
   (e) Let's focus on words about times and dates. Can the embeddings from the TED dataset differentiate between months (January, February, etc.) and plural units of time (minutes, hours, days, etc.)? What about the emebeddings from the Wiki dataset?
2. The `leases.zip` file contains a set of oil and gas leases. The actual text was obtained using OCR and contains a lot of errors. Train the embeddings with this dataset and analyze the embeddings of words related to times and dates. You may want to compare the most similar words and examine the t-SNE visualization (which words are near?). Comment on other clusters you identify in the visualization.

Submission. Submit your completed notebook and the report (pdf format) on Canvas. You will be primarily evaluated based on the report. I already know you can write code, the most interesting part is your analysis.