Statistical NLP - Assignment 3

due Wednesday October 16 - at 11:59pm by email to aparikh@cs.nyu.edu and urvish@nyu.edu

In this assignment, you will explore word embeddings and do some problem set style questions.

Deliverables:

- The output of the evaluation code and the word embeddings you trained.
- Writeup emailed to instructors and problem set questions. The recommended writeup length for this assignment is 1.5 pages. To give you some freedom to not worry about space saving, you may submit a report up to 4 pages (including any figures / charts). Reports (not including problem set questions) above 4 pages will be penalized.
- Answers to problem set questions. If you use good handwriting, you are welcome to handwrite them and email us a scanned copy.

Submission Format: Please title your submission email **Assignment 3 submission**. You should submit a zip file (firstname_lastname_hw5.zip) containing the report in pdf format (firstname_lastname_hw3.pdf) and the zipped code directory. Write your netID and your collaborators' netID (if any) in the report.

Resources: Please look at the Piazza post to download the resources for this assignment.

1 Part 1 (Word Embeddings)

Given a large corpus of text, your goal is to build vector representations of each of the words in that text in such a way that the cosine similarity of these vectors accurately represents the semantic similarity of the words that they represent.

Your representations will be evaluated on the wordSim353 dataset. The evaluation code calculates the similarity of a word pair by taking the cosine similarity of the two words' vectors. The entire dataset is then scored using the Spearman's rank correlation coefficient between these cosine similarities and the human annotations in the dataset.

Training Word Embeddings (10 points): You can use an already implemented algorithm for this assignment such as FastText:

```
https://fasttext.cc/
```

For training data you can use the data provided in the `data5/training-data` subdirectory of the provided code folder which comes from the first 1 million lines of:

```
http://www.statmt.org/lm-benchmark/
    1-billion-word-language-modeling-benchmark-r13output.tar.gz
```

Once you have trained your word embeddings you will need to convert them to a format suitable for the evaluation code. The embedding file should contain one word vector per line, with the word and each embedding dimension sep-

arated by a single whitespace. The first line should contain the number of words in the file, and the vector dimension, again separated by whitespace. For example, for three words and 2D embeddings, we can have a file whose contents may look like:

```
3 2
cat 0.8 0.0
dog 0.7 0.1
bat 0.5 0.5
```

If the embeddings are not all of the stated dimension, the cosine similarity score will not work.

Evaluation (20 points): The provided evaluation code is written in Java. You are not required to modify it, you simply need to run it with the following command (from the **classes** subdirectory).

```
java -classpath ../lib/commons-math3-3.5.jar::nlp/util/.*.java \
          nlp.assignments.WordSimTester -embeddings <path_to_vectors> \
          -wordsim ../../../data5/wordsim353/combined.csv
```

If for some reason the precompiled code doesn't run, you may need to download JDK from the Java website and run the below command to compile the code (from the **NLP_class** directory)

```
javac -d classes -cp lib/commons-math3-3.5.jar  */*/*/*.java
```

(If this does not work please consult with the instructors)

If you are curious the evaluation data is from here:
`http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/`

You are not allowed to optimize the vector representation learning algorithm on this dataset, but are welcome to take a look at the data. It is included for your convenience in the /data5/wordsim353/ directory.

For full credit your embeddings should get a score of at least 0.55.

Experiments/Writeup (25 points):

Regardless of the settings that obtain the desired score above, please experiment with **two** of the axes below and report your results in a writeup (with graphs/tables):

- amounts of training data
- dimensions for the embeddings
- contexts (linear, syntactic)
- objective functions

The recommended length is 1.5 pages. You will be scored on presenting the results of your experiments thoroughly (even if not all of them achieved positive results). Please use graphs/tables.

2 Part 2: Problem Set Questions

Repeated Features in Naive Bayes / Logistic Regression (25 points): Consider a classification problem with 3 binary features x_1, x_2, x_3 where $x_2 = x_3$ (i.e. they are identical features) and a binary class label y. Ideally, the addition of repeated (identical) features should not affect the classification decision. We will study the effect of these repeated features on Naive Bayes and Logistic Regression.

Assume all models below are trained with Maximum Likelihood to convergence.

- Mathematically characterize the effect (or non-effect) of repeated features on Naive Bayes e.g. how A Naive Bayes model trained only on (x_1, x_2) compares to one trained on (x_1, x_2, x_3).
- Mathematically characterize the effect (or non-effect) of repeated features on Logistic Regression (without any regularization) e.g. how a Logistic Regression model trained only on (x_1, x_2) compares to one trained on (x_1, x_2, x_3).
- Does addition of ℓ_2 regularization to logistic regression affect its sensitivity to repeated features? (If so, how?)

HMMs and CRFs (20 points):

Please answer True/False to the following questions about Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) **and give a justification**.

1. Training an HMM is faster (in big-O notation) than training a CRF.
2. You can use HMMs in unsupervised POS tagging but not CRFs.
3. Assume training is complete. Given a set of observations, computing the probability of a sequence of hidden states in HMMs is faster (in big-O notation) than it is in CRFs.
4. Assume training is complete. Given a set of observations, computing the most likely sequence of hidden states in HMMs is faster (in big-O notation) than it is in CRFs.