CS 5740, Spring 2016: Assignment 5

Please enter your groups in CMS by April 20. All students must work in pairs.

Due: May 7, 11:59pm

This assignment focuses on building vector representations (or embeddings) of words. The goal is to learn vector representations of words that point in similar directions for similar words, and different directions for different words.

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 Spearmans rank correlation coefficient between these cosine similarities and the human annotations in the dataset.

Resources You will want to use a large corpus of text. A good one to start with is this one, which is typically used for language modeling:

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

The data release (`https://www.dropbox.com/s/wkv3eyhalbofmmh/data5.zip?dl=0`) for this assignment is in the usual location. It contains the first 1M sentences in raw text and also analyzed with a part-of-speech tagger and dependency parser. The parsed data is in CONLL format (you can find a definition of the format online if it is unclear). If you want to use more data, you can download the entire corpus. Keep in mind that depending on your approach, you might have to parse it. Note that the data files can get quite big. Similarly, the embedding files will get big. Please submit only the embeddings for the words in the development and test data. The evaluation dataset is (also provided in the data release):

`http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/`

You are not allowed to optimize the vector representation learning algorithm on this dataset. It is included for your convenience in the `data5/wordsim353/` directory. The data is divided into development (`dev.csv`) and test (`test.csv`). The test file is for generating the final results only, while the development file is to report ablations and use during development. Other resources that might be useful are (a copy of Wordnet is in the data release):

WordNet: `https://wordnet.princeton.edu/wordnet/download/`
PPDB: `http://paraphrase.org/`

Task There are many papers on the topic of learning word embeddings. You might want to read a selection of these before settling upon an approach. A simple one that uses syntax is here:

`http://www.cs.bgu.ac.il/~yoavg/publications/acl2014syntemb.pdf`

All models of distributional similarity have a notion of the context in which a word occurs. In the simple model that is included, the context is all of the content words in a sentence. The context can also be chosen to capture syntactic and semantically relevant information. The simple model that included maps each word into a raw vocabulary space. Embeddings find lower dimensional representations that capture most of the information in the raw feature space with respect to some loss function. The literature describes multiple ways of doing this. You will have to experiment with different choices for at least three of:

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

Code Please download `code5.zip`. The starting point for this assignment is the class:

```
edu.cornell.cs.nlp.assignments.WordSimTester
```

A very basic vector space model of words is included in the code, but you will want to implement something more interesting, like word2vec (please dont just download the open-source version of word2vec!). The code to generate the embeddings should be submitted alongside the embeddings. You are free to use other NLP tools, such as POS taggers, parsers, and so on. You do not have to build your code within the example package. If you use Python, you are free to use NumPy and SciPy. If there is another scientific computing package in Java or Python, which you wish to use, please ask. To generate the final result, you must use the evaluation script that is part of the Java package. It is written to allow loading of saved embeddings, which you are to save in a standard format (see below). You can build the jar file using `ant dist` and run the baseline as follows (make sure your classpath includes the math library needed by the evaluation code):

```
java -classpath lib/commons-math3-3.5.jar:dist/code5.jar
edu.cornell.cs.nlp.assignments.WordSimTester
-embeddings <path_to_vectors>
-wordsim <data_dir>/wordsim353/dev.csv
-trainingdata <data_dir>/training-data/training-data.1m
-trainandeval
-wordnetdata <data_dir>/wordnet/core-wordnet.txt
```

Output Format The output embedding file should contain one word vector per line, with the word and each embedding dimension separated 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! Please submit only embeddings for the words that appear in the test and development data. Otherwise the submission file will get too big. The provided code contains functionality to prune the embeddings to only the ones that are needed.

Submission Instructions Your submission on CMS must include both your writeup **in PDF format**, your code in **in a ZIP/TAR.GZ/TGZ file** and the embeddings requested above. The names of all members of the group must be indicated clearly at the top of the first page. The report page limit is **four pages**. Do not submit the data provided with the assignment – this will cause CMS to reject your submission.

The assignment was adapted from Slav Petrov's Course at NYU.