LING570 Hw10: Word analogy task
Due: 12/7

In this assignment, you implement the algorithm for solving the word analogy task; that is, given A, B, and C, find D such that A to B is like C to D. The algorithm is on slides 12-13 of day19-word-representation.pdf. You can also read the paper (Mikolov et al., 2013) which provides more detail. The example files are under /dropbox/17-18/570/hw10/examples/.

Q1 (55 points): Write a script **word_analogy.sh** that finds D given A, B, and D.

- The command line is: word_analogy.sh vector_file input_dir output_dir flag1 flag2
- vector_file is an input file with the format "w v1 v2 ... vn" (e.g., **vectors.txt**). Here, $< v1, v2, ..., vn >$ is word embedding of the word w.
- input_dir (e.g., **question-data**) is a directory that contains a list of test files. The lines in the test file have the format "A B C D", the four words as in the word analogy task.
- output_dir is a directory to store the output:
  - For each file under input_dir, your script should create a file with the same name under output_dir.
  - The two files should have exactly the same number of lines and the same content, except that the word D in the files under output_dir is the word selected by the algorithm; that is, you will go over all the words in vector_file and find one what is most similar to y = $x_B - x_A + x_C$.
- flag1 is an interger indicating whether the word embeddings should be normalized first.
  - If flag1 is non-zero, you need to normalize the word embedding vectors first. That is, if the vector is $< v_1, v_2, ..., v_n >$, you normalize that to $< v_1/Z, v_2/Z, ..., v_n/Z >$, where $Z = \sqrt{v_1^2 + v_2^2 + ... + v_n^2}$.
  - If flag1 is 0, just use the original vectors.
- flag2 is an integer indicating which similarity function to use for calculating sim(x,y):
  - If flag2 is non-zero, use cosine similarity (https://en.wikipedia.org/wiki/Cosine_similarity).
  - If flag2 is 0, use Euclidean distance (https://en.wikipedia.org/wiki/Euclidean_distance).
  - Note that when Euclidean distance is used, the smaller the distance is, the more similar the two words are.

In addition to output_dir, your script should print out to stdout (1) accuracy for each file under input_dir and (2) *total accuracy*. The stdout can then be redirected to a file (see **eval_result**).

- You should print out the following to stdout:

```
fileName1
ACCURACY TOP1: acc% (cor/num)
```

```
fileName2
ACCURACY TOP1: acc% (cor/num)
...
Total accuracy: accTotal% (corSum/numSum)
```

- $fileName_i$ is i_{th} file in the input_dir.
- *num* is the number of examples in the file.
- *cor* is the number of examples in the file that your system output is correct (i.e., the D in *output_dir/filename* is the same as the D in *input_dir/filename*)
- $acc\% = \frac{cor}{num}$.
- For total accuracy line, *corSum* is the sum of the *cor*, and *numSum* is the sum of *num* in the previous lines.
- accTotal% = $\frac{corSum}{numSum}$.

Q2 (20 points): Run the following commands and submit output dirs:

- mkdir exp00 exp01 exp10 exp11
- word_analogy.sh vectors.txt question-data exp00 0 0 > exp00/eval_res
- word_analogy.sh vectors.txt question-data exp00 0 1 > exp01/eval_res
- word_analogy.sh vectors.txt question-data exp00 1 0 > exp10/eval_res
- word_analogy.sh vectors.txt question-data exp00 1 1 > exp11/eval_res

Here, vectors.txt and question-data are the ones under /dropbox/17-18/570/hw10/examples/.

Submission: Your submission should include the following:

1. readme.[txt|pdf] with any note that you want the grader to read.
2. hw.tar.gz that includes word_analogy.sh and the output directories created in Q2 (see the complete file list in submit-file-list).