This assignment is due **Wednesday, November 24 at 11:59PM**.

1. Goals

Through this assignment you will:

- Investigate issues and design of distributional semantic models.
- Analyze the effects of different context sizes as well as association measures in distributional similarity models.
- Evaluate distributional models relative to human assessments.

[Back to Top]

2. Background

Please review the class slides and readings in the textbook on distributional semantics and models. The count-based and word2vec models are to be implemented separately, so that you may do the more extensive coding required for the count-based distributional model in your preferred programming language and then use the Python-based **`gensim`** package for the **`word2vec`** implementation.

[Back to Top]

3. Creating and Evaluating Count-based Models of Distributional Semantic Similarity

Implement a program to create and evaluate a distributional model of word similarity based on local context term cooccurrence. Your program should:

1. Read in a corpus that will form the basis of the distributional model and perform basic preprocessing.
    - All words should be lowercase.
    - Punctuation should removed, both from individual words and from the corpus (e.g. "," should not be a word).
      Only alphanumeric (a-z, 0-9) characters should remain.
      Note: in many regex packages, including Python's, `\w` matches a single alphanumeric character, while `\W` denotes a single non-alphanumeric character.
2. For each word in the corpus:
    - Create a vector representation based on word cooccurrence in a specified window around the word.
    - Each element in the vector should receive weight according to a specified weighting
3. Read in a file of human judgments of similarity between pairs of words. (See Files Section)
4. For each word pair in the file:
    - For each word in the word pair:
        - Print the word and its ten (10) highest weighted features (words) and their weights, in the form:

        - **`word feature1:weight1 feature2:weight2 ….`**
    - Compute the similarity between the two words, based on **cosine similarity** (e.g. using `scipy.spatial.distance.cosine`).
    - Print out the similarity as: `wd1,wd2:similarity`
5. Lastly, compute and print the Spearman correlation between the similarity scores you have computed and the human-generated similarity scores in the provided file as: `correlation:computed_correlation`.
   You may use any available software for computing the correlation. In Python, you can use `spearmanr` from `scipy.stats.stats`.

[Back to Top]

4. Programming

Create a program **`hw7_dist_similarity.sh`** that implements the creation and evaluation of the distributional similarity model as described above and invoked as:
`hw7_dist_similarity.sh <window> <weighting> <judgment_filename> <output_filename>`, where:

- **`<window>`**
    - An integer specifying the size of the context window for your model. For a window value of **2**, the window should span the two words before and the two words after the current word.
- **`<weighting>`**
    - A string specifying the weighting scheme to apply: “FREQ” or “PMI”, where:
        - **FREQ**: Refers to “term frequency”, the number of times the word appeared in the context of the target
        - **PMI**: (Positive) Point-wise Mutual Information: A variant of PMI where negative association scores are removed.
- **`<judgment_filename>`**
    - The name of the input file holding human judgments of the pairs of words and their similarity to evaluate against, mc_similarity.txt.
    - Each line is of the form:
        - `wd1,wd2,similarity_score`
- **`<output_filename>`**:
    - The name of the output file with the results of computing similarities and correlations over the word pairs.
    - The file name should identify the configuration under which it was run, as in:
        - **`hw7_sim_<window>_<weighting>_output.txt`**
        - e.g. **`hw7_sim_30_FREQ_output.txt`** would hold the results of running the bag of words model with context window of 30 and term frequency weights.

In this assignment, you should use the Brown corpus provided with NLTK in **`/corpora/nltk/nltk-data/corpora/brown/`** as the source of cooccurrence information. The file is white-space tokenized, but all tokens are of the form **`“word/POS”`**.

If you choose to use NLTK, you may use the Brown corpus reader as in:
`brown_words = nltk.corpus.brown.words()`

[Back to Top]

5. Comparison to Continuous Bag of Words (CBOW) using Word2Vec

Implement a program to evaluate a predictive CBOW distributional model of word similarity using Word2Vec. Your program should:

1. Read in a corpus that will form the basis of the predictive CBOW distributional model and perform basic preprocessing.
    - All words should be lowercase.
    - Punctuation should removed.
2. Build a continuous bag of words model using a standard implementation package, such as gensim's word2vec
3. Read in a file of human judgments of similarity between pairs of words.
4. For each word pair in the file:
    - Compute the similarity between the two words, using the **`word2vec`** model
    - Print out the similarity as: **`wd1,wd2:similarity`**
5. Lastly, compute and print the Spearman correlation between the similarity scores you have computed and the human-generated similarity scores in the provided file as:
    - **`correlation:computed_correlation`**.
    - You may use any available software for computing the correlation. In Python, you can use **`spearmanr`** from **`scipy.stats.stats`**.

NB: If you want to play with pre-trained embeddings before doing your own training, try:

```
import gensim.downloader as api
wv = api.load('word2vec-google-news-300')
```

and have a look at their tutorial.

[Back to Top]

Programming #2

Create a program **`hw7_cbow_similarity.sh`** that implements the creation and evaluation of the Continuous Bag-of-Words similarity model as described above and invoked as:

`hw7_cbow_similarity.sh <window> <judgment_filename> <output_filename>`, where:

- **`<window>`**
    - An integer specifying the size of the context window for your model.
    - For a window value of **2**, the window should span the two words before and the two words after the current word.
- **`<judgment_filename>`**
    - The name of the input file holding human judgments of the pairs of words and their similarity to evaluate against, **`mc_similarity.txt`**.
        - Each line is of the form: `wd1,wd2,similarity_score`
- **`<output_filename>`**
    - The name of the output file with the results of computing similarities and correlations over the word pairs. The file name should identify the configuration under which it was run, as in:
    - `hw7_sim_<window>_CBOW_output.txt`
        - e.g. `hw7_sim_30_CBOW_output.txt` would hold the results of running the Continuous Bag of Words model with context window of `30`.

[Back to Top]

6. Files

Test and Example Data Files

Aside from the Brown corpus, all files related to this assignment may be found on patas in **`/dropbox/21-22/571/hw7/`**, as below:

- **`mc_similarity.txt`**
  - These are the pairs of words whose similarity is to be evaluated under each of your models, along with human similarity judgments from [Miller and Charles, 1991].
  - Each line is of the form:
    - `wd1,wd2,similarity_score`
- **`example_similarity_output.txt`**
  - This file holds an example output file with term frequency weights and no pre-processing.

Submission Files

- **`hw7.tar.gz`**: Tarball including the following:
  - **`hw7_dist_similarity.sh`**: Program which implements and evaluates your count-based distributional similarity model.
  - **`hw7_cbow_similarity.sh`**: Program which implements and evaluates the word2vec similarity model.
  - **`hw7_sim_2_FREQ_output.txt`**: Output of running your program with `window=2` and `weighting=FREQ`.
  - **`hw7_sim_2_PMI_output.txt`**: Output of running your program with `window=2` and `weighting=PMI`.
  - **`hw7_sim_10_PMI_output.txt`**: Output of running your program with `window=10` and `weighting=PMI`.
  - **`hw7_sim_2_CBOW_output.txt`**: Output of running your `hw7_cbow_similarity.sh` program with `window=2` using word2vec.
- **`readme.{txt|pdf}`**:
  - This file should describe and discuss your work on this assignment. Include problems you came across and how (or if) you were able to solve them, any insights, special features, and what you learned. Give examples if possible. If you were not able to complete parts of the project, discuss what you tried and/or what did not work. This will allow you to receive maximum credit for partial work.
    In particular, you should discuss the effects of window size, weighting, and model on the quality of the similarity model, as captured by the correlation between your automatically calculated similarity and human judgments.

[Back to Top]