

# CSE40201 Assignment 1

## Distributional Word Representations

Due: Mar 29, 2022, 11:59:00pm KST

**Submission Instructions:** Submit your report and code (but not the data) via BlackBoard. It is your responsibility to make sure your code is runnable and if needed, please include `readme.txt` to describe how to run your code. If your code is not runnable, you will get no point.

**Collaboration Policy:** You are welcome to discuss assignments with others in the course, but solutions and code must be written individually.

You will implement and experiment with ways of creating word vectors. Let's begin by defining a **vocabulary**  $V$  and a context vocabulary  $V_C$ . The steps below will ask you to implement different ways of building word vectors for the words in  $V$ , using context words from  $V_C$ . I will provide you a corpus of sentences from English Wikipedia, along with files that you can use to fill  $V$  and  $V_C$ . Before describing the ways of computing word vectors, we will discuss evaluation and the provided data files.

**EVALWS(M):** You will quantitatively evaluate your word vectors by comparing the word similarities they produce to human-annotated word similarities. That is, you will be provided with files containing pairs of words along with their human-annotated similarity scores. You should use your word vectors (represented in the matrix  $M$ ) to compute the cosine similarity of each word pair in the word similarity files. Use Spearman's rank correlation coefficient (also called Spearman's  $\rho$ ; see `scipy.stats.spearmanr`) as the evaluation metric. That is, for each word pair in a word similarity dataset, you will compute its cosine similarity using your word vectors, which forms a list of similarities. Then, form another list of similarities for those same word pairs, this time containing the manually annotated similarities from the word similarity datasets we will provide. Compute Spearman's  $\rho$  between the two lists of similarities. Do this for two word similarity datasets: MEN and SimLex-999. The files are provided for you. (If any word pairs contain a word that does not appear in the training corpus, simply use a cosine similarity of 0.0 for that pair.)

## Provided Data

The following data files are provided to you for this assignment:

- `wiki-{0.1,1}percent.txt` - samples of English Wikipedia downloaded May 15, 2015, with punctuation separated from words, all characters converted to lowercase, and with one sentence per line. For any results and analysis reported in your submission, you should use `1percent`. The smaller file is provided for debugging and preliminary testing. It should still produce reasonable nearest neighbors for most common words.
- `men.txt` - MEN word similarity dataset. The first line is a header, then each line has the format “word1[tab]word2[tab]similarity score”.
- `simlex-999.txt` - SimLex-999 word similarity dataset. Same format as MEN.
- `vocab-wordsim.txt` - vocabulary file containing the words from the word similarity evaluations (one word per line). Use `vocab-wordsim.txt` for  $V$  when you are computing word similarity correlations, but not when you are finding nearest neighbors.
- `vocab-25k.txt` - vocabulary file containing the 25000 most common words in Wikipedia. When computing word similarity correlations, use `vocab-wordsim.txt` for  $V$  and `vocab-25k.txt` for  $V_C$ . When finding nearest neighbors, use `vocab-25k.txt` for both  $V$  and  $V_C$ .

## 1 Required Tasks (50 points total)

### 1.1 Distributional Counting (20 points)

Implement distributional counting for word vectors. Build a word-context matrix  $C$  of counts, with size  $|V| \times |V_C|$ . Each row of  $C$  corresponds to a word in  $V$ . Each column of  $C$  corresponds to a word in  $V_C$ . A particular entry  $C_{ij}$  in  $C$  should equal the number of times that context word  $j$  appears within  $w$  words of word  $i$  in the Wikipedia corpus. A context window contains  $w$  words to either side of the center word, so it contains  $2w + 1$  words in total (including the center word). Use  $w = 3$ . Use `vocab-wordsim.txt` to populate  $V$  and use `vocab-25k.txt` to populate  $V_C$ . When computing counts, be sure to pad the sentences with the start-of-sentence symbol `<s>` and end-of-sentence symbol `</s>`. (These are both included as words in the beginning of the `vocab-25k.txt` file.) Call `EVALWS(C)` and report your results on both MEN and SimLex-999. As a sanity check, your Spearman correlation for MEN should be close to 0.22. Submit your code.

### 1.2 Computing PMIs (10 points)

After building  $C$  as above, build a new matrix  $C_{pmi}$  that contains pointwise mutual information (PMI) values for each entry (i.e., each word pair). See Section 4.2 of Turney and Pantel (2010) for more details. Call `EVALWS( $C_{pmi}$ )` and report your results. You should see improvements in Spearman correlation for both MEN and SimLex-999. Submit your code.

### 1.3 Experimentation (5 points)

Report the results of  $\text{EVALWS}(C)$  and  $\text{EVALWS}(C_{pmi})$  for three values of  $w$ : 1, 3, and 6. You should notice systematic trends in terms of correlation as window size changes. Look at the manually-annotated similarities in the MEN and SimLex-999 datasets and describe why you think the two datasets show the trends they do.

### 1.4 Analysis (15 points)

In this component, you will analyze your word vectors qualitatively by computing nearest neighbors for particular query words and comparing them across window sizes. You may also find it helpful to manually inspect instances of certain words in the original corpus to find typical contexts for each word.

For this section, use vocab-25k.txt to populate both  $V$  and  $V_C$ . Compute the PMI matrix  $C_{pmi}$  with window sizes  $w = 1$  and  $w = 6$ . To find the  $n$  nearest neighbors for a query word, compute cosine similarity between the query word and all words in  $V$  and then keep the  $n$  words with highest cosine similarity. When doing so, omit the query word from the list of nearest neighbors since it will always have cosine similarity of 1.

#### 1.4.1 Warm-up: Printing nearest neighbors (3 points)

For the two window sizes  $w = 1$  and  $w = 6$ , compute and print the 10 nearest neighbors for the query word monster. (For example, the nearest neighbor of monster is dragon with  $w = 1$  and evil with  $w = 6$ .)

#### 1.4.2 Part-of-speech tag similarity in nearest neighbors (7 points)

Do nearest neighbors tend to have the same part-of-speech tag as the query word, or do they differ? Does the pattern differ across different part-of-speech tags for the query word? How does window size affect this?

Explore these questions by choosing query words with different parts of speech and computing their nearest neighbors. When choosing query words, consider nouns, verbs, adjectives, and prepositions. (Hint: when considering verbs, use inflected forms like *transported*.) Try a few query words from each part of speech category and see if you can find any systematic patterns when comparing their nearest neighbors across window sizes  $w = 1$  and 6. When the neighbors differ between window sizes, how do they differ? Can you find any query words that have almost exactly the same nearest neighbors with the two window sizes?

Discuss your findings, showing examples of nearest neighbors for particular words to support your claims.

### **1.4.3 Words with multiple senses (5 points)**

Now try choosing words with multiple senses (e.g., *bank*, *cell*, *apple*, *apples*, *axes*, *frame*, *light*, *well*, etc.) as query words. What appears to be happening with these words? What happens when you compare the neighbors with different window sizes ( $w = 1$  vs.  $w = 6$ )? Discuss your findings, showing examples of nearest neighbors for particular words to support your claims.