

# AI539 - Natural Language Processing with Deep Learning

## Homework 1 Report

### Word Vectors: Distributed Representations of Words

Author: Vy Bui

OSUID: 934370552

Instructor: Professor Stefan Lee

The School of Electrical Engineering and Computer Science  
Oregon State University

**Task 1.1**

Implement the `tokenize` function in `Vocabulary.py` that processes a string into an array of strings corresponding to tokens. You are free to implement any tokenization schema that eliminates punctuation. If you want to try out lemmitization, the `nlTK` package may be helpful. In your writeup for this question, include a description of what you implemented and include an example input-output pair.

**Task 1.2**

Implement the `build_vocab` function in `Vocabulary.py` which constructs a finite vocabulary from a string containing all the text from the training corpus. This includes implementing some heuristic for thresholding the number of words and building the `word2indx` and `indx2word` indexes.

**Task 1.3**

Implement the `make_vocab_charts` function in `Vocabulary.py` to produce Token Frequency Distribution and Cumulative Fraction Covered charts like those above for your tokenizer and vocabulary cutoff heuristic. We recommend `matplotlib` for this. Afterwards, running `build_freq_vectors.py` will generate these plots. In your write-up for this question, include these plots and briefly describe the cutoff heuristic you implemented and explain your rationale for setting it.

**Task 2.1**

What are the minimum and maximum values of PMI (Eq. 1)? If two tokens have a positive PMI, what does that imply about their relationship? If they have a PMI of zero? What about if the PMI is negative? Based on your answers, what is an intuition for the use of PPMI?

**Task 2.2**

Implement the `compute_cooccurrence_matrix` function in `build_freq_vectors.py` which takes a list of article overviews and a vocabulary and produces a co-occurrence matrix  $C$ . It is up to the student to define what a context is. Note that looping in Python is quite slow such that unoptimized versions of this function can take quite a long time. Feel free to save the result while you are developing to reduce time in future runs (see `numpy.save/numpy.load`). In your writeup for this task, describe how you defined your context.

**Task 2.3**

Implement the `compute_ppmi_matrix` function in `build_freq_vectors.py` which calls `compute_cooccurrence_matrix` and then computes a PPMI matrix given a list of article overviews and a vocabulary. Hint: Add a small constant to  $C$  to avoid problems with  $\log(0)$ .

**Task 2.4**

It has all led up to this! Run `build_freq_vectors.py` and examine the visualized word vectors. You should have semantically meaningful clusters in your plot. Zoom in and identify three such clusters. In your write-up, include an image of each cluster and what category you think it is representing. Also include the plot as a whole.

**Task 3.1**

Derive the gradient of the objective  $J_B$  with respect to the model parameters  $w_i, \tilde{w}_j, b_i, \tilde{b}_j$ . That is, write the expression for  $\nabla_{w_i} J, \nabla_{\tilde{w}_j} J, \nabla_{b_i} J, \nabla_{\tilde{b}_j} J$ . Note that parameters corresponding to words not in B will have zero gradient.

**Task 3.2**

Implement the gradient computation for a batch in the corresponding Task 3.2 section of `build_glove_vectors.py`.

**Task 3.3**

Run `build_glove_vectors.py` to learn GloVe vectors and visualize them with TSNE! In your write-up, describe how the loss behaved during training (how stably it decreased, what it converged to, etc). Also include the TSNE plot. If everything has been done correctly, you should observe similar clustering behavior as in Task 2.3.

**Task 4.1**

Use the `most_similar` function to find three additional analogies that work. In your response, provide the analogies in the compact `a:b::c:?` form, the model's list of outputs, and why you consider this output to satisfy the analogy.

**Task 4.2**

Use the `most_similar` function to find three analogies that did not work. In your response to this question, provide the analogies in the compact `a:b::c:?` form, the model's list of outputs, and why you consider this output to not satisfy the analogy.

**Task 4.3**

Use the `most_similar` function to find two additional cases of bias based on gender, politics, religion, ethnicity, or nationality. In your response, provide the analogies in the compact `a:b::c:?` form, the model's list of outputs for both `a:b::c:?` and `c:b::a:?`, and why you consider this output to be biased based on the model's two responses.

**Task 4.4**

Why might these biases exist in `word2vec` and what are some potential consequences that might result if `word2vec` were used in a live system?

## References