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.

remove punctuation and separate words by spaces.

There are still many other ways to try, like lemmatization or stemming,

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 idx2word 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 rational 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?

PMI values can go from negative infinity to positive infinity. If two tokens have a positive PMI, they co-occur more often than we expect by chance. Intuitively speaking, they tend to appear together in some predefined context. On the other hand, a negative PMI implies that the two tokens co-occur less often than expected by chance. A zero PMI implies that the probability that two tokens co-occur is just about the same as they occur together by chance. PPMI is favored because negative PMI is only reliable for immense corpora. Imagine two tokens each of which has probability of one millionth. To determine if two tokens co-occur less often than expected, the probability of the two appearing with one another has to have a magnitude of one thousand billionth, which requires a huge corpus [JM08].

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.

1. Context?
2.

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.

PPMI seems to effectively draw the association between words. Figure 2 shows a cluster of countries and georaphical locations such "US", "India", "Britain", and "China". Figure 3 shows a cluster of words that are highly related to American football. More interestingly, figure 4 shows that the Middle East countries and cities often co-occur with war-related words such as "killing", "bomb", "hostage", and "violence". This seems to be the public image of the Middle East represented by the media.

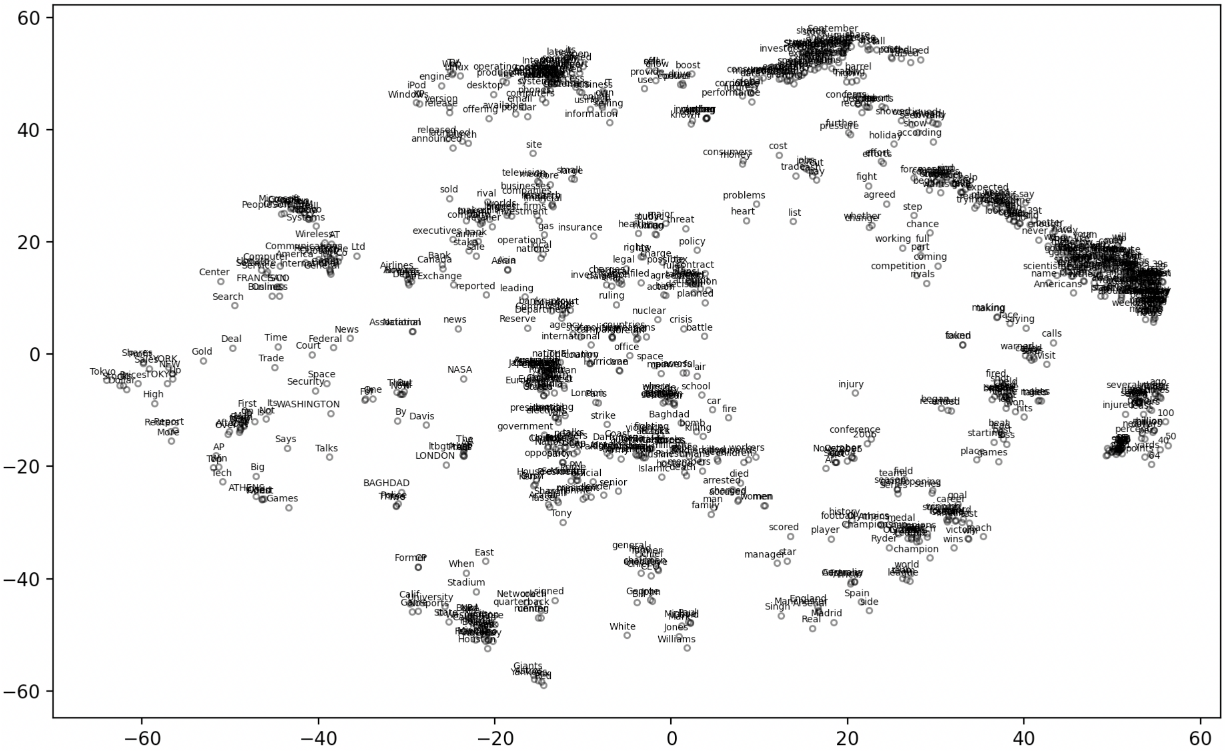

Figure 1: Whole plot

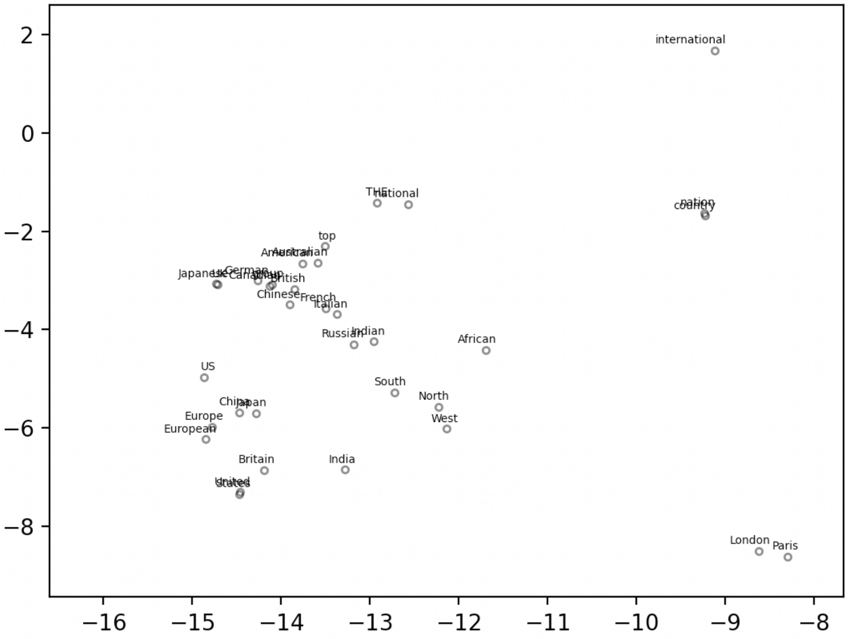

Figure 2: Country cluster

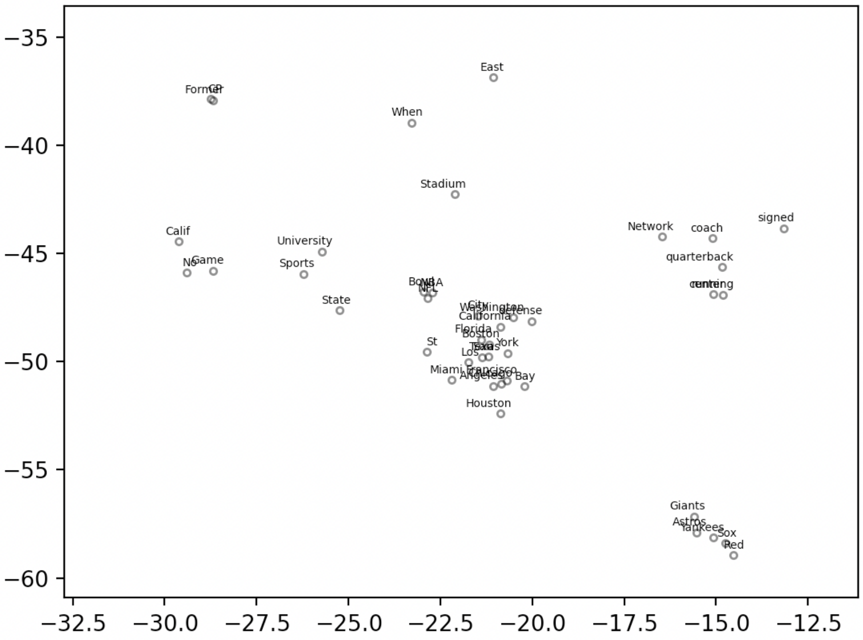

Figure 3: American football cluster

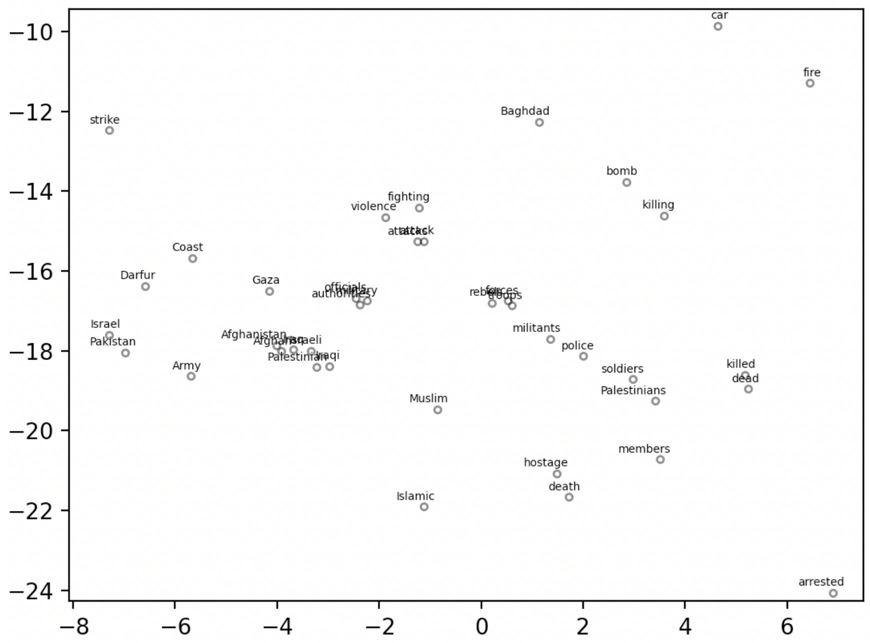

Figure 4: Middle East war cluster

Task 3.1
Derive the gradient of the objective J_B with respect to the model parameters $w_i, \widetilde{w}_j, b_i, \widetilde{b}_j$. That is, write the expression for $\nabla_{w_i} J, \nabla_{\widetilde{w}_j} J, \nabla_{b_i} J, \nabla_{\widetilde{b}_j} J$. Note that parameters corresponding to words not in B will have zero gradient.

$$\nabla_{w_i} J = \sum_{(i_m, j_m) \in B} 2f(C_{i_m j_m})(w_{i_m}^T \widetilde{w}_{j_m} + b_{i_m} + \widetilde{b}_{j_m} - logC_{i_m j_m})\widetilde{w}_{j_m}$$

$$\nabla_{\widetilde{w}_j} J = \sum_{(i_m, j_m) \in B} 2f(C_{i_m j_m})(w_{i_m}^T \widetilde{w}_{j_m} + b_{i_m} + \widetilde{b}_{j_m} - logC_{i_m j_m})w_{i_m}$$

$$\nabla_{b_i} J = \sum_{(i_m, j_m) \in B} 2f(C_{i_m j_m})(w_{i_m}^T \widetilde{w}_{j_m} + b_{i_m} + \widetilde{b}_{j_m} - logC_{i_m j_m})$$

$$\nabla_{\widetilde{b}_j} J = \sum_{(i_m, j_m) \in B} 2f(C_{i_m j_m})(w_{i_m}^T \widetilde{w}_{j_m} + b_{i_m} + \widetilde{b}_{j_m} - logC_{i_m j_m})$$

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

```
c_biases_grad = w_biases_grad = np.multiply(np.multiply(fval, 2), error)
wordvecs_grad = np.multiply(wordbiases_grad, c_batch)
contextvecs_grad = np.multiply(contextbiases_grad, w_batch)
```

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.

Tried tuning with combinations of three hyperparameters, B,d,lr.

64_0.08 seems to be the best combination, converged to 0.028 after 40 epochs.

losses monotonically decrease after each epoch, but they oscillate between iterations within one epoch, which is expected for stochastic gradient descent?.

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

[JM08] Daniel Jurafsky and James Martin. *Speech and Language Processing.* Prentice Hall, second edition, 2008.