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.

Before tokenization, punctuations were removed and words were split with spaces as delimeters. The 0-index is reserved for unknown words "UNK". For example, passing "Bears are huge sdfsf !!!" to tokenize() produced ['Bears', 'are', 'huge', 'sdfsf'].

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.

Refer to Vocabulary.py.

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.

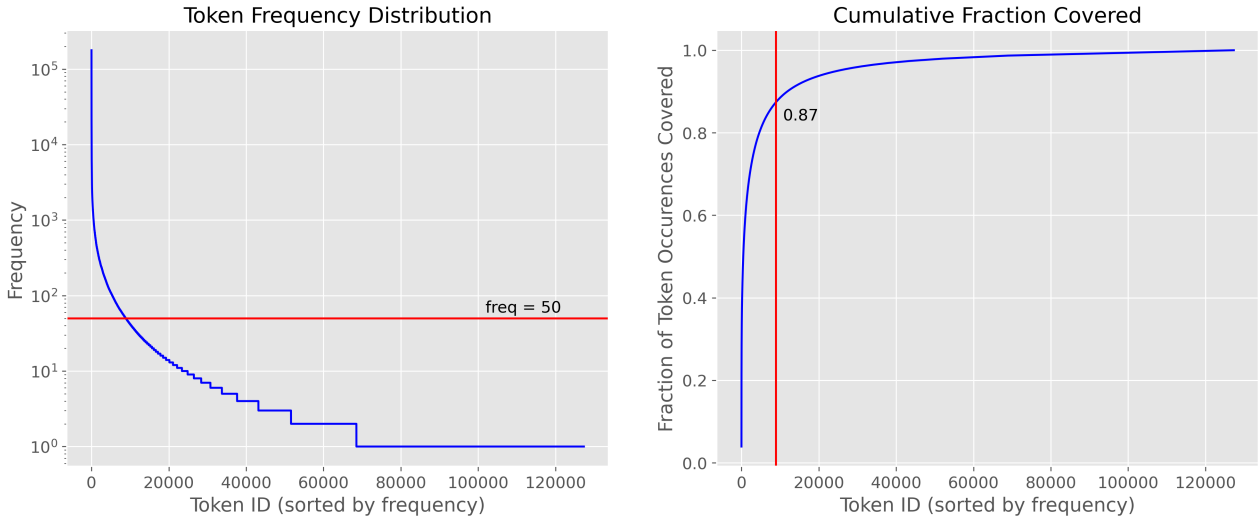

Figure 1: Token Frequency Distribution and Cumulative Fraction Covered

The cutoff heuristic is tokens with frequency less than 50. As shown by Figure 1, the Cumulative Fraction Covered was about to plateau after the cutoff point. Hence, pushing the cutoff line down would significantly increase the computation of downstream tasks but gain little extra information. Furthermore, because representations should be evaluated by assessing the performance of downstream NLP tasks (extrinsic evaluation), it is more practical to start with small vocabulary size to accelerate the other NLP tasks to get feedback faster for further adjustment.

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.

The used context was 2-windows around the word of interest. For example, in "Today is a sunny day, but yesterday was not", if the word of interest is "a", then "Today is" and "sunny day" are the context words.

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).

Refer to build_freq_vectors.py.

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 3 shows a cluster of countries and georaphical locations such "US", "India", "Britain", and "China". Figure 4 shows a cluster of words that are highly related to American football. More interestingly, figure 5 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.

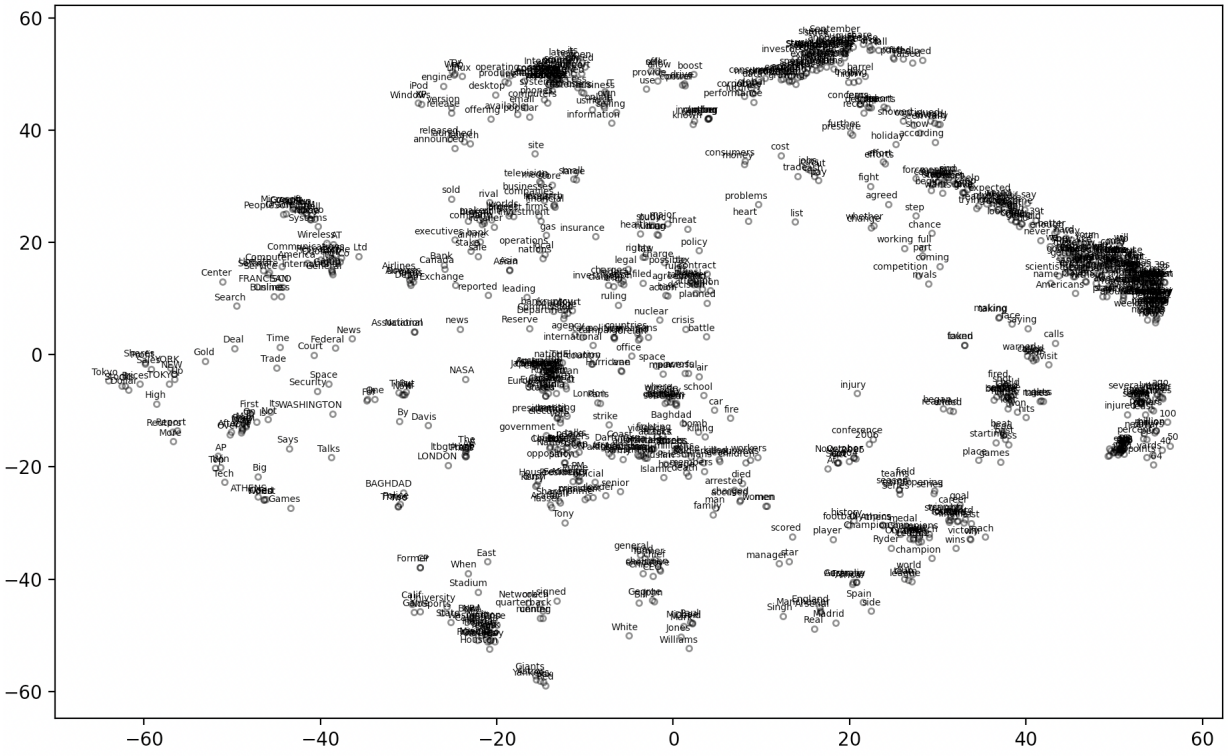

Figure 2: Whole plot

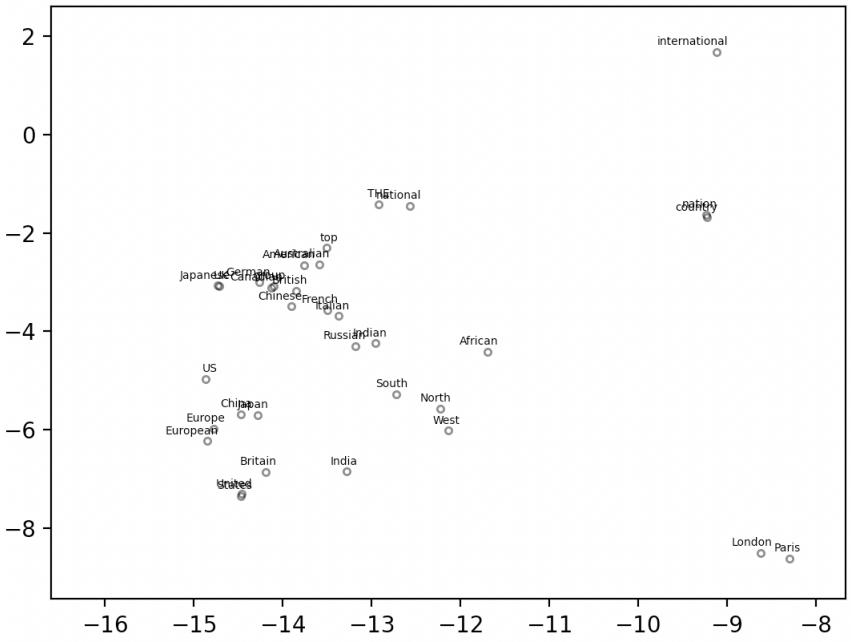

Figure 3: Country cluster

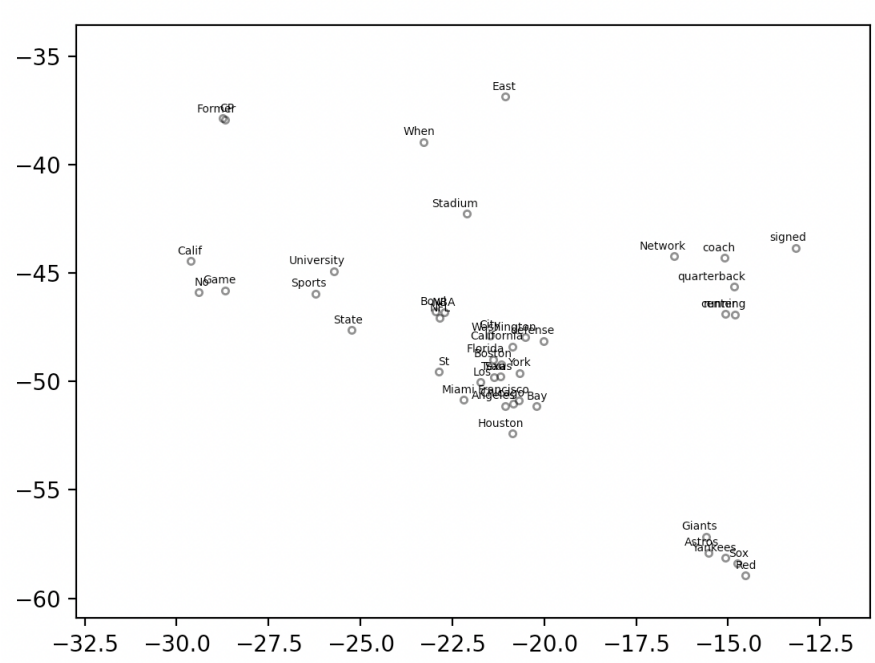

Figure 4: American football cluster

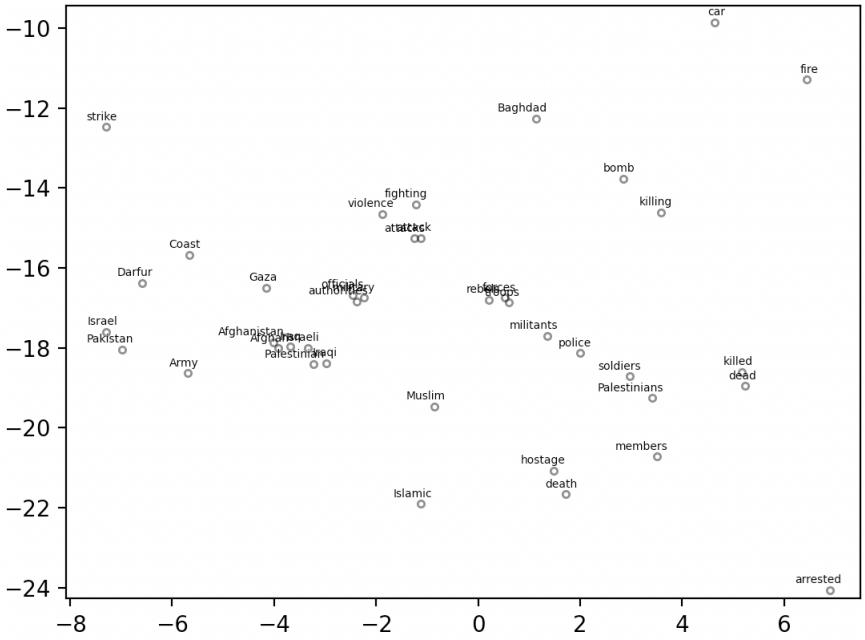

Figure 5: Middle East war cluster

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.

The model was trained with combinations of various values of batch size, dimensionality, and learning rate in 40 epochs. As can be seen in Figure 6 and 7, the average loss monotonically decreased after each epoch for all of the combinations. The losses droped significantly at the first 2 epochs, then gradually decreased until plateauing at around 0.03. Training with batch size of 2048, dimensionality of 64, and learning rate of 0.08 yielded the best performance, reaching the loss of approximately 0.028 after 40 epochs.

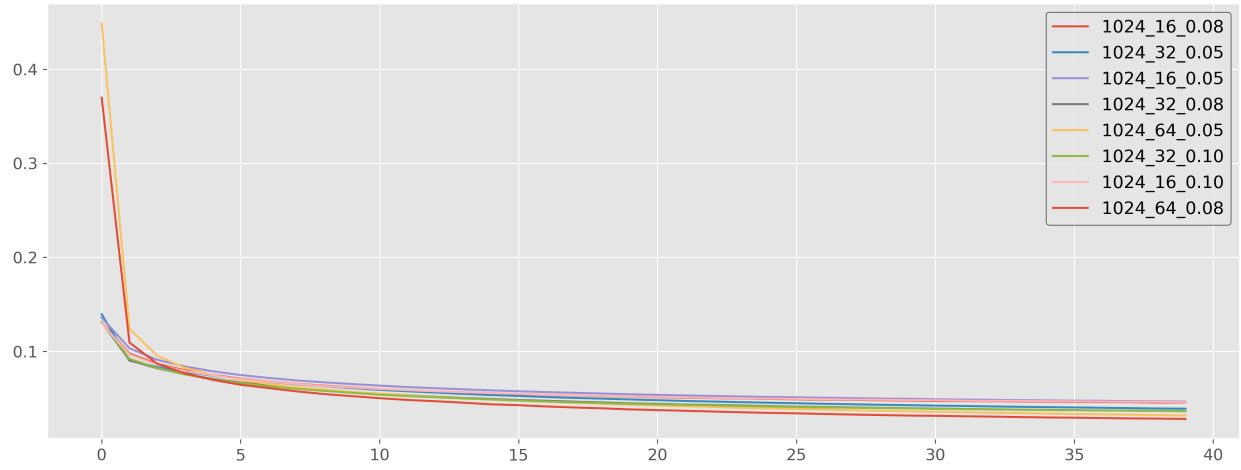

Figure 6: Hyperparameters tuning with batch size of 1024

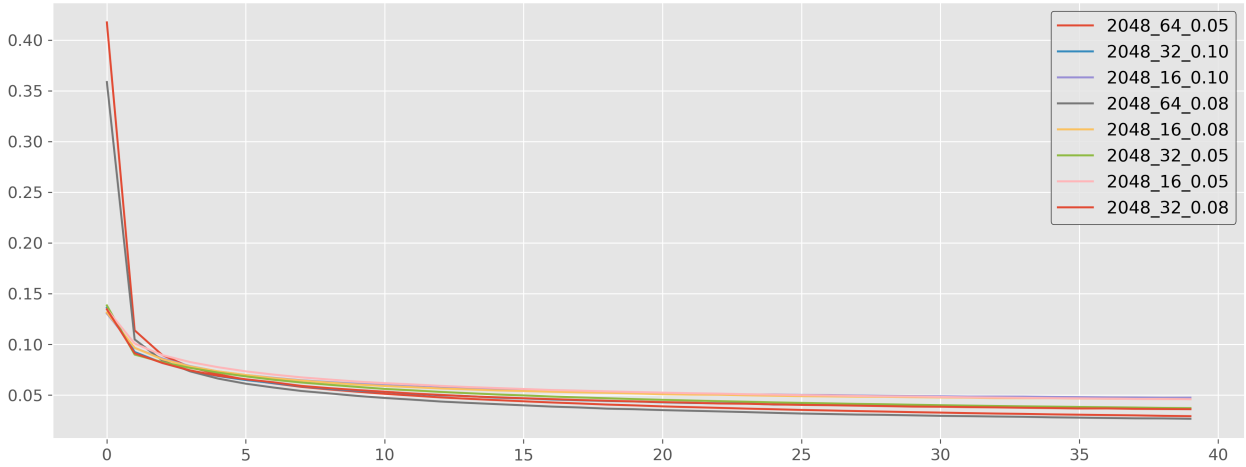

Figure 7: Hyperparameters tuning with batch size of 2048

Figure 8 shows TSNE plot of the best model. Similar clusters to those in task 2.4 can be observed in this plot. In particular, 9 shows a cluster of locations and 10 shows a cluster of quantities.

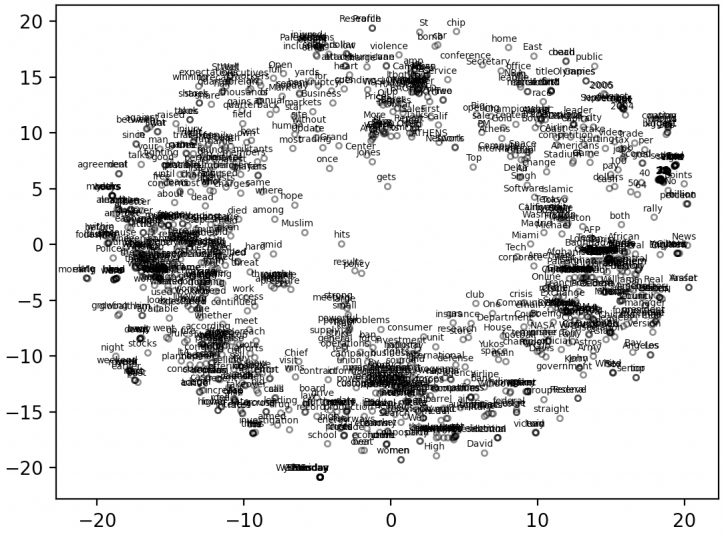

Figure 8: GloVe TSNE bird's-eye view

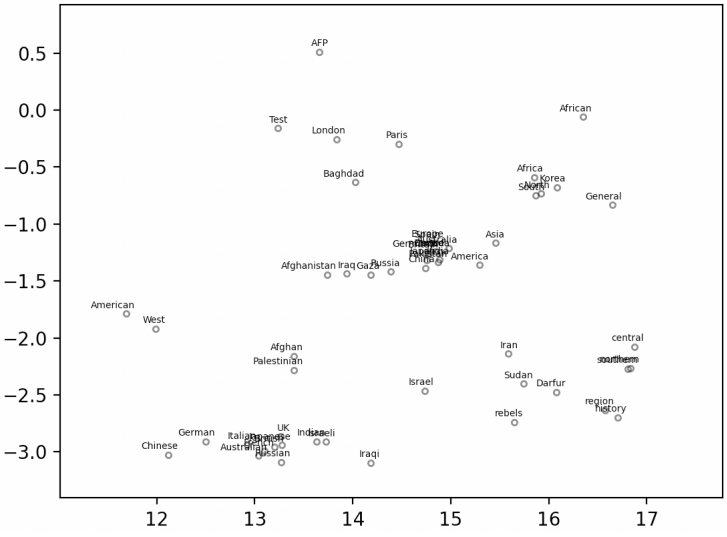

Figure 9: GloVe TSNE location cluster

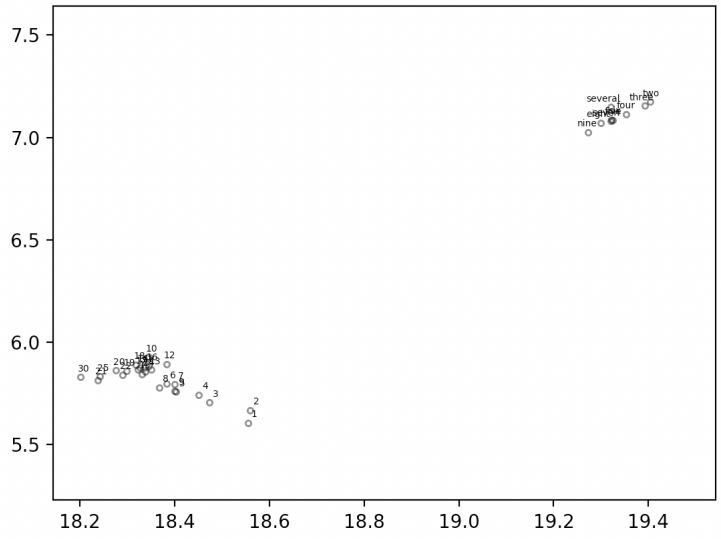

Figure 10: GloVe TSNE quantity cluster

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.

Table 1 shows three reasonable analogies. First, "Messi : Argentina :: Neymar : ?" returned Brazil, which is indeed the country which the famous soccer player Neymar comes from as Messi comes from Argentina. Second, football is used in the United Kingdom as soccer is used in Canada and the US. Finally, Paris is the capical of France as Beijing is that of China.

word	cosine	word	cosine	word	cosine
Brazil	0.622	canada	0.585	Paris	0.721
Uruguay	0.581	usa	0.585	French	0.624
Chile	0.561	australia	0.525	Colombes	0.609
Ecuador	0.56	india	0.511	Marseille	0.597
Argentine	0.555	germany	0.502	Melun	0.589
Argentinean	0.535	malaysia	0.501	Aix_en_Provence	0.578
Paraguay	0.525	amsterdam	0.5	Issy_les_Moulineaux	0.578
Peru	0.521	denmark	0.496	Montpellier	0.574
Argentinian	0.505	thailand	0.488	Toulouse	0.571
striker_Neymar	0.503	bbc	0.483	Nantes	0.566

Table 1: reasonable analogies, Messi:Argentina::Neymar:?, football:uk::soccer:?, China:Beijing::France:?, in order from left to right

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.

Table 2 shows three reasonable analogies. First, although the analogy works out for China and France as shown in Table 1, it does not work for China and the US. Second, the model cannot relate Beyonce to the US as Adele is a famous female singer from England. Lastly, the model cannot relate CIA to the US as FIS, the Foreign Intelligence Service of the Russian Federation, is to Russia.

word	cosine	word	cosine	word	cosine
Unites_States	0.58	stock_symbol_BNK	0.467	Kremlin	0.506
U.S.	0.569	Freddie_Flintoff	0.458	former_Soviet_republics	0.494
United_Sates	0.539	ticker_symbol_BNK	0.455	Moscow	0.482
Untied_States	0.532	DAILY_STAR_Fabio_Capello	0.446	Soviet	0.474
theUnited_States	0.488	Wayne_Rooney	0.446	former_Soviet_Republics	0.473
Washington_DC	0.477	David_Beckham	0.444	CIA_operatives	0.47
Washington	0.465	NatWest_Challenge	0.437	Russian	0.467
UnitedStates	0.453	Beyoncé	0.434	Soviet_Union	0.466
Great_Britain	0.444	Frank_Lampard	0.429	Ukraine	0.465
Athens_Greece	0.439	midfielder_Frank_Lampard	0.427	spy	0.46

Table 2: unreasonable analogies, China:Beijing::United_States:?, Adele:England::Beyonce:?, FIS:Russia::CIA:?, in order from left to right

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.

Table 3 demonstrates gender bias in word2vec. In particular, when the model was asked boy:soccer::girl:?, volleyball and softball are the most similar instead of soccer. On the other hand, girl:soccer::boy:? returned soccer as the most similar word. This implies that boys are more likely to play soccer than girls.

word	cosine
volleyball	0.694
softball	0.668
Soccer	0.649
basketball	0.632
lacrosse	0.618
water_polo	0.596
football	0.596
Volleyball	0.589
tennis	0.585
hockey	0.564

word	cosine
Soccer	0.692
football	0.692
basketball	0.566
fooball	0.545
sports	0.542
baseball	0.542
futbol	0.542
hockey	0.54
SOCCER	0.532
Futsal	0.518

Table 3: bias based on gender in word2vec. The left table shows the most similar to boy:soccer::girl:? and the right table shows the most similar words to girl:soccer::boy:?

Researchers found that there are association between names and races. For example, Gabe has 86% probability of being a White person's name, whereas 70% percent of people named Miguel are Hispanic [SDAI+19]. More interestingly, word2vec shows that Miguel is much more similar to smugglers than Gabe is. Particularly, as shown by Table 4, Miguel:smuggler::Gabe:? returned smugglers with 0.486 cosine similarity, while Gabe:smuggler::Miguel:? produced smugglers with much higher similarity, at 0.596. This implies that a Hispanic is more associated with smugglers than a White person.

word	cosine
smugglers	0.486
smuggling	0.449
Smugglers	0.406
smuggled	0.397
Rabiyah	0.387
trafficker	0.386
smuggle	0.385
drug_smuggler	0.378
Zach	0.375
informant	0.373

word	cosine
smugglers	0.596
migrant_smuggler	0.542
trafficker	0.54
smuggling	0.529
traffickers	0.507
drug_trafficker	0.505
Colombian_traffickers	0.504
Juárez_Cartel	0.501
immigrant_smuggler	0.494
border_crosser	0.483

Table 4: bias based on race in word2vec. The left table shows the most similar to Miguel:smuggler::Gabe:? and the right table shows the most similar words to Gabe:smuggler::Miguel:?

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?

Word2vec contains biases because it captures hidden stereotypes present in the society. In case of "word2vec-google-news-300", it is professional journalists' negative beliefs towards gender and race that were quantified by the model [BCZ+16]. Word embeddings have become the defacto standard method to represent text data in many NLP tasks. Therefore, latent biases contained in these models could be learned by NLP models, which might pose a risk of reinforcing stereotypes in society.

References

[BCZ+16] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. Man is to computer programmer as woman is to homemaker? debiasingword embeddings. *Advances in Neural Information Processing Systems*, 2016.

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

[SDAI+19] Nathaniel Swinger, Maria De-Arteaga, Neil Thomas Hefferman IV, Mark Leiserson, and Adam Tauman Kalai. What are the biases in my word embedding. *In Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 2019.