Generating a PCA Graph Using the Skip-gram Model

ECE-472 Deep Learning Midterm Project

Ayden Shankman and Gavri Kepets

October 2022

1 Introduction

The Skip-gram model is a method for learning vector representations for words, and is typically effective for learning implicit relationships between them. For example, England and China might not be necessarily used together in a dataset too often, but Skip-gram has the capability of representing these words similarly, because they are both countries. In the paper “Distributed Representations of Words and Phrases and their Compositionality” by Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado and Jeffrey Dean, new methods are introduced as an extension of the Skip-gram model to improve performance and break through certain limitations that the model has [1].

In the paper, a PCA graph is introduced to show the relationships that are observed from the vectorizations of the words.

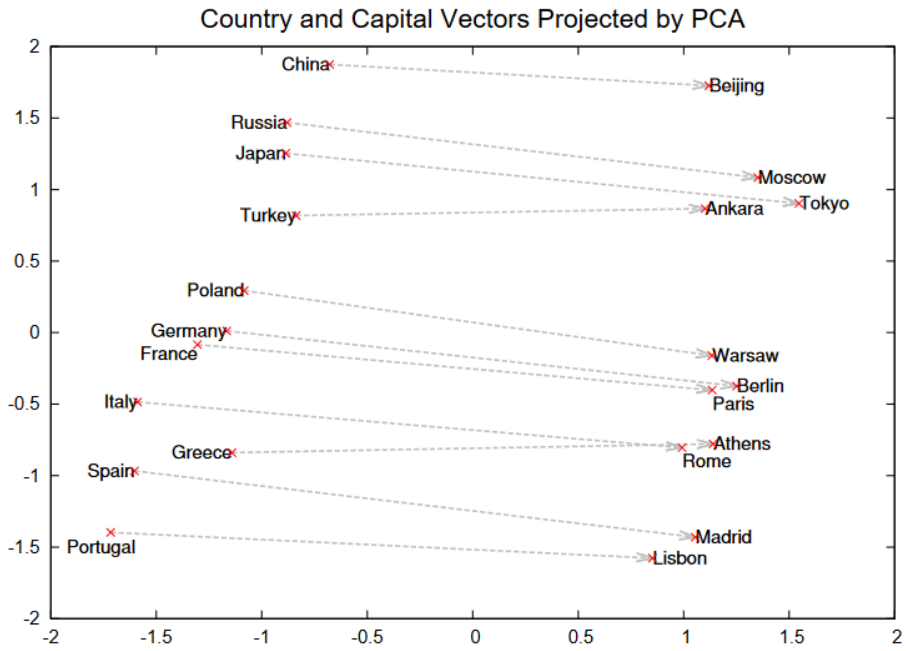
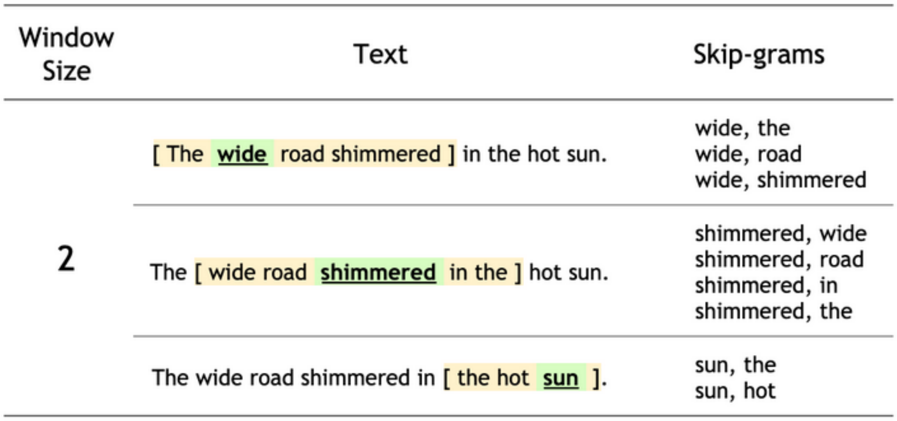


Figure 1: The PCA graph generated in the paper showing the relationships that are observed from the vectorizations of various countries and their capitals.[1]

This PCA shows that the Skip-gram has generated vectorizations that identify that the countries belong together and that the cities belong together. However, there is an additional relationship that is drawn, which is each country to their respective capitals. The goal of this project was to train a Skip-gram model and generate a similar PCA graph.

2 How It Works

The Skip-gram model has a unique way of creating word vectorizations. Instead of using One Hot Encoding, which assigns a unique value to each word, the Skip-gram model labels words based on the surrounding words. “Context” words are selected for given “target” words, where the target word is one word in the sentence and the context words are the words that surround the target word.



Window Size	Text	Skip-grams
2	[The wide road shimmered] in the hot sun.	wide, the wide, road wide, shimmered
	The [wide road shimmered in the] hot sun.	shimmered, wide shimmered, road shimmered, in shimmered, the
	The wide road shimmered in [the hot sun].	sun, the sun, hot

Figure 2: Word2Vec's example for target and context words. ”Skip-grams” are pairings between the target word and its respective context word [2].

In the “Distributed Representations of Words and Phrases and their Compositionality” paper, a method called negative sampling is used. While context words are used as labels for the target words, negative sampling looks at words that are too far away from the target word to be considered a context word, and assigns them to be a negative label. This creates positive and negative associations between the target word and the other words that are part of the sequence.

3 Implementation

In order to generate the PCA graph, we needed to create word embeddings with a Skip-gram model. The model consists of two embedding layers, one for the target words and the other for the embedding words. We used various different datasets and models to test this. Our first model had an embedding size of 128 and was trained on the Brown corpus from NLTK which consists of one million words of American English texts printed in 1961. It has has over 44,700 unique words within 57,000 sentences.

Corpus	Brown Corpus
Embedding Size	128
Epochs	20
Context Window Size	5
Negative Samples	4

To test the embedding, we plotted the PCA and looked for if words that belonged to the same category were close together. The topics we used were emotions, energy resources, and types of areas. As seen in Figure 3, this model did not produce good results.

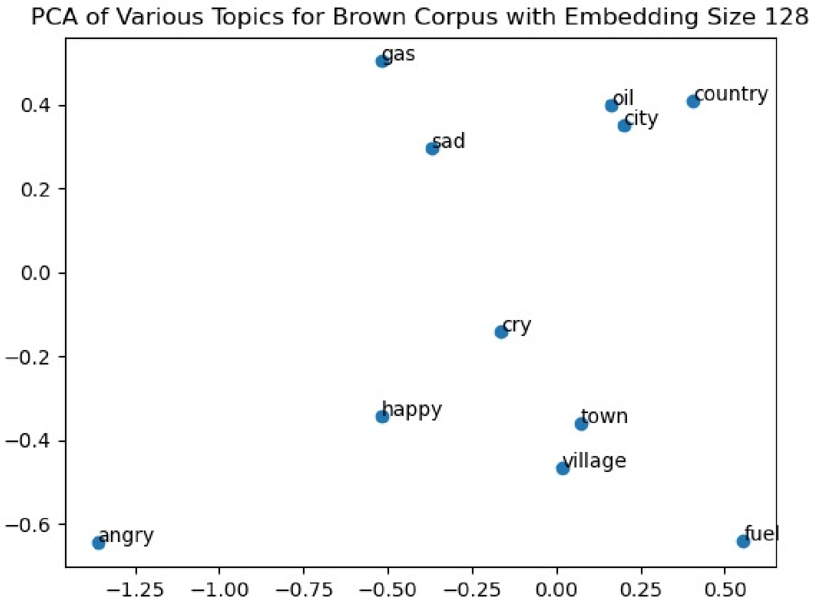


Figure 3: Initial results of PCA projection of various topics from the Brown Corpus.

Our next step was to find the ideal embedding size. We generated seven models, with embedding sizes of 8, 16, 32, 64, 128, 256, and 512. Each embedding size was trained with less and less epochs, as the model with a size 8 embedding layer was about 15 seconds per epoch, while the model with a 512 layer was about 4 minutes per epoch. To confirm that the loss of the models were converging, the loss was plotted. After generating PCA graphs for each model, we found that an embedding size of 32 seemed to work best as seen in Figure 4.

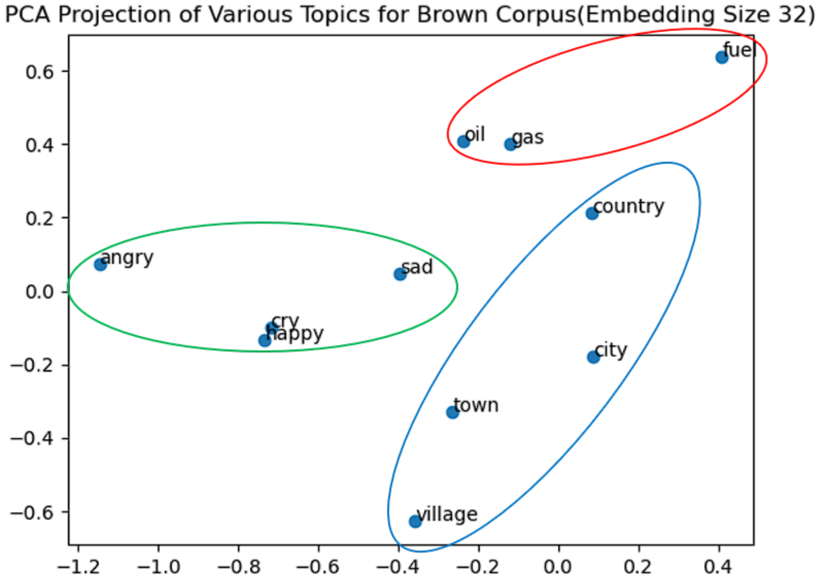


Figure 4: PCA projection graph of various topics for the Brown Corpus with an embedding size of 32. Emotions are seen in green, energy resources are red, and types of areas are blue.

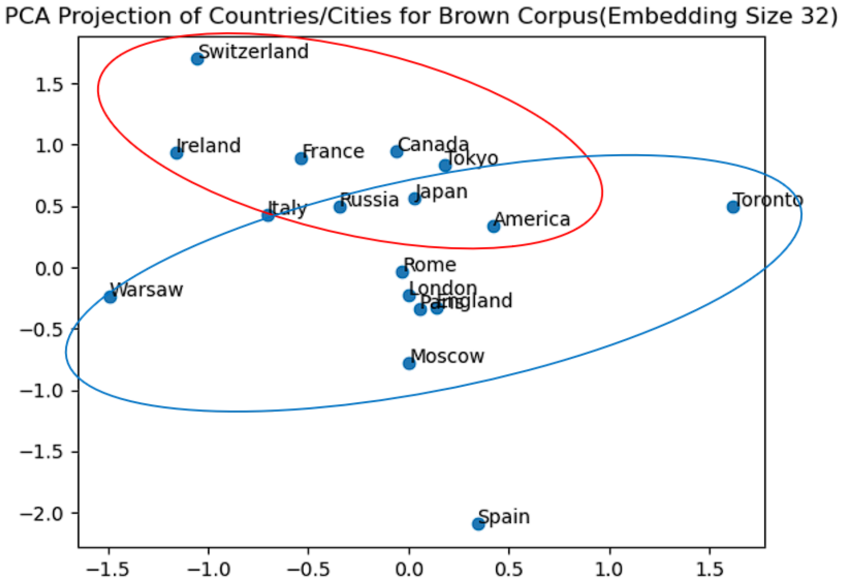


Figure 5: PCA projection graph of Countries and Cities for the Brown Corpus with an embedding size of 32. The Countries are mostly grouped together in the upper left in red and the cities are mostly grouped together in the center in blue.

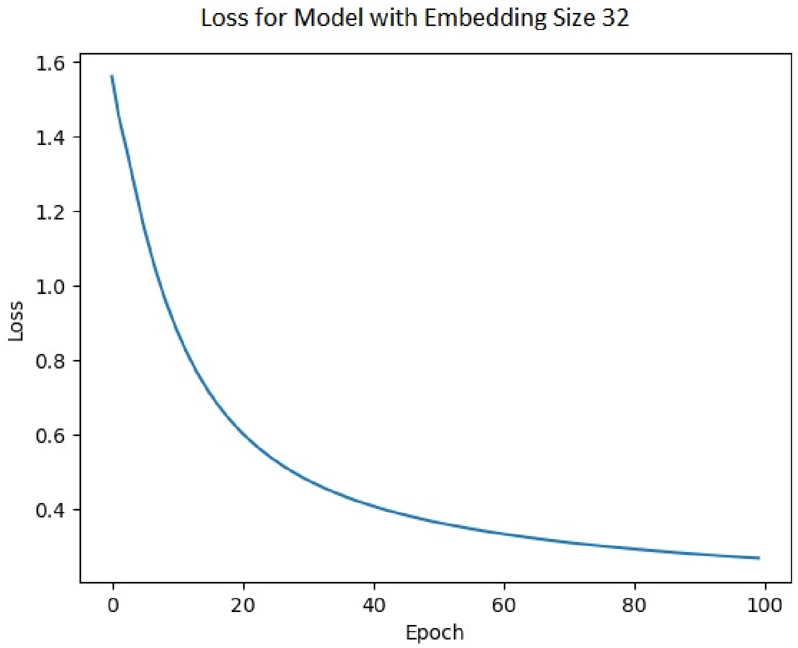


Figure 6: Loss for model trained on the Brown Corpus with embedding size of 32.

We also experimented with the amount of negative samples and the size of the context window. The paper claimed that for smaller training datasets, they found that larger negative samples (5-20) produced better results, while for larger ones 2-5 worked well [1]. Therefore, we raised the negative samples from 4 to 10. The model with the specifications below produced the best results for the various topics in the Brown Corpus. The words that are relevant to each other are vaguely clustered together as seen in Figure 7. However, the PCA for the countries and cities seemed to be worse when we increased the negative sampling as seen in Figure 8.

Corpus	Brown Corpus
Embedding Size	32
Epochs	100
Context Window Size	5
Negative Samples	10

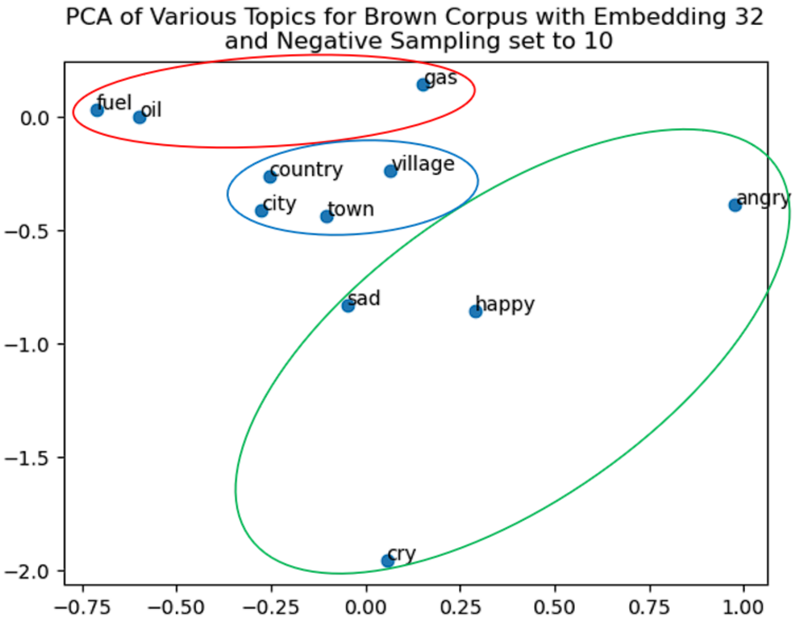


Figure 7: PCA Projection graph of various topics for Brown Corpus with embedding size 32 and negative sampling set to 10. Emotions are seen in green, energy resources are red, and types of areas are blue.

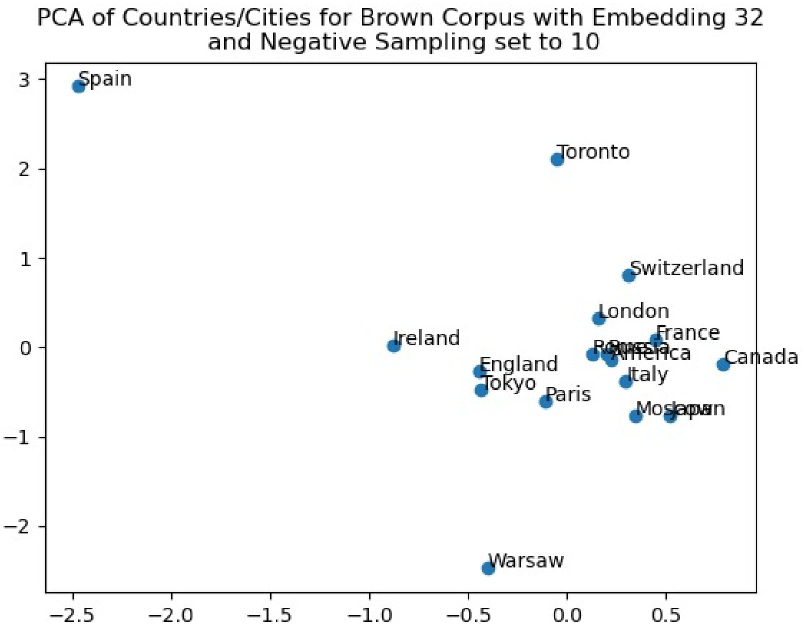


Figure 8: PCA Projection graph of countries and cities for Brown Corpus with embedding size 32 and negative sampling set to 10.

The results of this experiment were fine, but not nearly as evident as what the paper had. The paper used a significantly larger dataset, so we decided to test on a larger dataset. The c4 dataset is Google's cleaned up version of the Web Crawl corpus. It is significantly larger than the Brown corpus, with over 300 million samples. We do not have the computing power or memory to work with the entire dataset, so we decided to start with one file, which contained over 300 thousand samples. Tokenizing the entire file caused our computers to run out of memory (it was using over 25 GB of RAM), so we trained it on half of the first file. This was still about seven times larger than the Brown corpus in terms of the vocabulary size. Training on the larger model took about seven hours to converge over 30 epochs, but resulted in a decent PCA graph for the various topics as seen in Figure 9. This model was also tested on countries and cities, which produced unremarkable results as seen in Figure 10.

Corpus	c4 Corpus
Embedding Size	32
Epochs	30
Context Window Size	5
Negative Samples	4

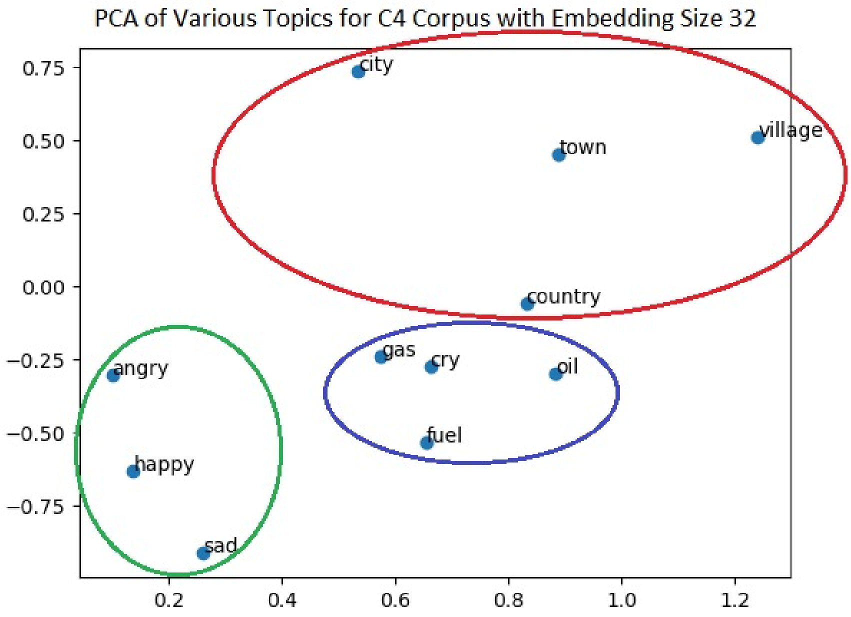


Figure 9: PCA graph of various topics for the c4 corpus. Emotions are seen in green, energy resources are blue, and types of areas are red.

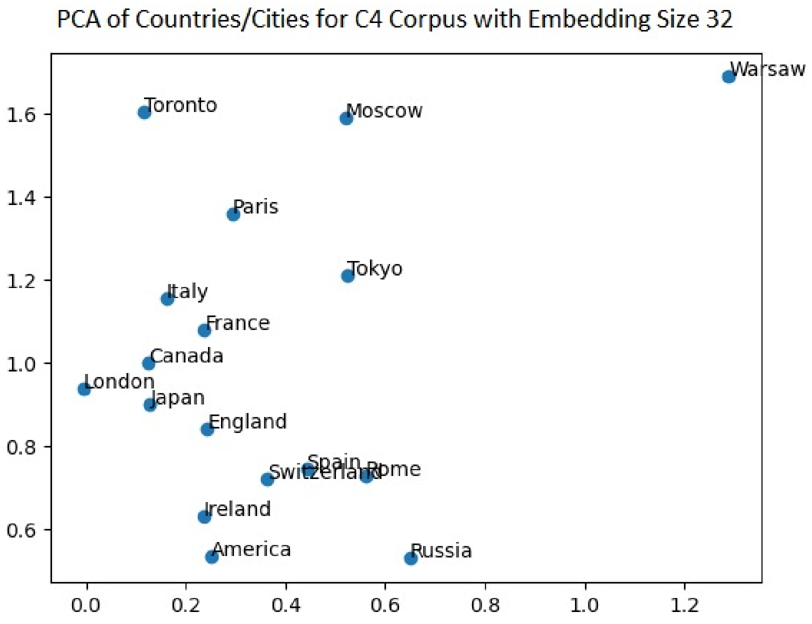


Figure 10: PCA graph of cities and countries for the c4 corpus.

4 Conclusion

After working with the Skip-gram model and using it to generate word embeddings for a generalized dataset, we concluded that in order to strongly learn the implicit relationships between words, a large dataset and model are needed along with extensive training time. Furthermore, it is not necessarily beneficial to have larger embeddings. We found that the model trained with an embedding size of 32 produced better results than models with larger embedding sizes. However, while we did get some desirable results, they were not nearly as clear as the paper's results due to our limited computing power.

5 Bibliography

References

[1] Mikolov, Tomas Sutskever, Ilya Chen, Kai Corrado, G.s Dean, Jeffrey. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems. 26.

[2] Word2vec:Tensorflow Core. TensorFlow. (n.d.). Retrieved from https://www.tensorflow.org/tutorials/text/word2vec