Word Vectorization

iNLP | Assignment 3

Nanda Rajiv
2021115002

Singular Value Decomposition

This is a frequency-based method, which is done by building a co-occurrence matrix and performing matrix SVD.

Implementation

After cleaning, adding start end tokens, and tokenizing the data, we get a list of sentences. From this, we build a vocabulary of words that occur above a certain frequency. Post that, we can decide on a context window size, and build a co-occurrence matrix for all the words in the vocabulary. Then, we perform SVD on that matrix, and get the first embedding size number of columns of the U matrix, which are the embeddings for each word.

Why It Works

Intuitively, SVD gives us a matrix (the U matrix) whose columns are sorted by the information about the data that they give us. That is, the first column gives the most information, the second, the second most information and so on. Hence, by selecting some number of embedding size (I have chosen 300), we can 300 rows that give us the rows that gives the most information to be got from that number of rows.

Skip Gram with Negative Sampling

This is a prediction-based method, done by predicted the context words given a target word.

Implementation

Cleaning, tokenizing and building vocabulary is done in a similar way to the SVD method. Following this, we make a set of negative samples and a set of positive samples, depending on the context window. Then, we make a simple feed-forward neural network, which makes an embedding layer with the vocabulary size, and two linear layers (with a non-linearity (ReLU)), and a singmoid at the end, to map to 0 or 1. Basically, this is a binary classifier, which determines whether a word is indeed in the context of a given central word (1) or not (0). We use Binary Cross Entropy Loss and Adam Optimiser to back propogate, and basically learn both the weights of the linear layers and the weights of the embedding layer. The 'weights' of the embedding layer basically gives us the embeddings of the skip-gram method. These are then frozen, and can be used for downstream tasks.

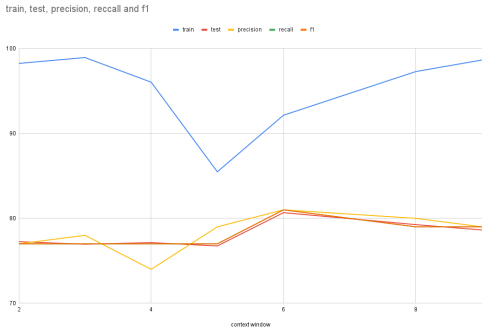
Why It Works

You're basically treating the embeddings as the weights of a trainable layer in a neural net, which is a binary classifier for the skip-gram prediction case. As it gets better at predicting whether a word is in the context or not, it also gets better at representing the words as vectors, and those weights give the embeddings.

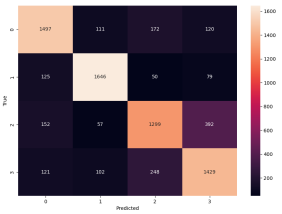
Hyperparameter Tuning

SVD

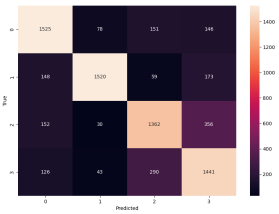
window	train	test	prec	recall	f1
2	98.24	77.25	77	77	77
3	98.93	76.95	78	77	77
4	96.03	77.14	74	77	77
5	85.48	76.75	79	77	77
6	92.14	80.67	81	81	81
8	97.28	79.26	80	79	79
9	98.64	78.63	79	79	79



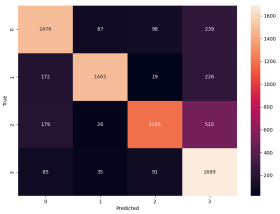
SVD for different context windows



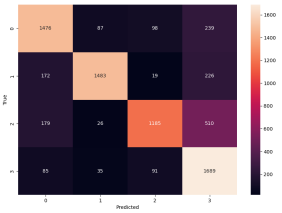
2



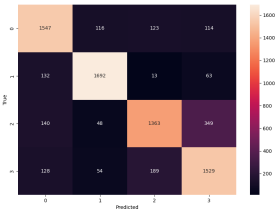
3



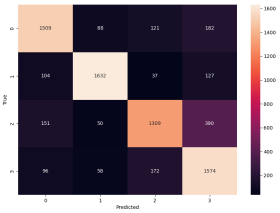
4



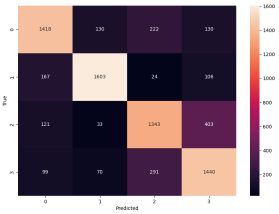
5



6



8



9

Confusion matrices for Different Context Windows for SVD

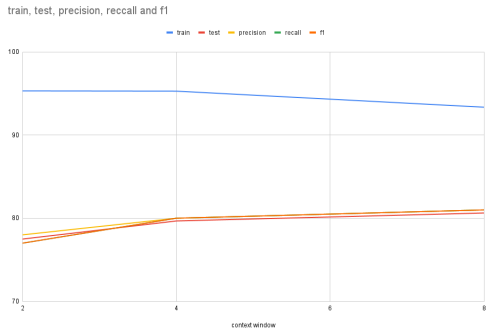
What we observe is that there is not a massive difference between different context window sizes when it comes to SVD, and they perform largely similar. The best test accuracy is achieved that a context window size of 6.

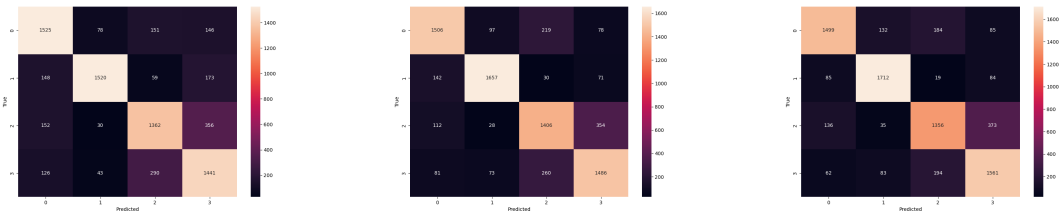
Skip Gram

NOTE: Skip-gram method takes a much longer time to train for different context windows, hence I did not do as many cases as I did for SVD. I restricted to three cases.

window	train	test	prec	recall	f1
2	95.32	77.49	78	77	77
4	95.29	79.67	80	80	80
8	93.36	80.63	81	81	81

SVD for different context windows





2 4 8

Confusion matrices for Different Context Windows for SVD

In the case of Skip Gram, it is clear that for these set of context sizes, as the context size increases, the test accuracies also improve. However, there is a diminishing effect. The difference from 2 to 4 is not much in terms of context window size, but is 2% difference in test accuracy. Meanwhile, a 4 size difference from 4 to 8 shows a 1% increase in test accuracy. This can lead one to conclude that there may be a point after which this improvement will diminish to a negligible point, or perhaps a diminishing accuracy.

Comparison of SVD and Skip Gram

We can three specific context window sizes, 2, 4 and 6.

	skipgram					svd				
window	train	test	precision	recall	f1	train	test	precision	recall	f1
2	95.32	77.49	78	77	77	98.24	77.25	77	77	77
4	95.29	79.67	80	80	80	96.03	77.14	74	77	77
8	93.36	80.63	81	81	81	97.28	79.26	80	79	79

Skip Gram vs. SVD

Why these specific size? I wanted to select a good range of values (low and high), and these are accounted for by 2 and 8. On the other hand, given we observed a larger variance between a pair of lower sizes separated by some value, when compared with a pair of larger sizes separated by the same value, 4 was chosen as an in-between value than can demonstrate this.

Best Performing Configuration? We observe that in both cases, the larger context window performs better for test accuracy, whereas for train accuracy this is flipped. It is

possible that smaller context windows lead to more overfitting when compared to larger context windows.

Analysis

Overall, it is clear that skip gram performs better than SVD, based on all metrics overall.

Skip gram 'learns' the context that words occur in, whereas SVD simply 'memorises' them, in a way that captures the semantic meaning less.

Another thing, is that in theory, a larger context size would add noise to the data a lot more, however, in this case, the data is limited to news headlines. Most words occur in somewhat similar contexts, for example 'stock' would probably occur near 'trade', rather than near 'photograph', and 'banks' would probably occur near 'american' rather than near 'plane'.

If we had more diverse data, it is possible that SVD would perform significantly worse than skip gram in greater context sizes, because of larger noise in the data.

Shortcomings of SVD

One major shortcoming is memory constraints. In order to implement this, we had to limit the number of sentences and have a minimum frequency. In the absence of these, it is not possible with our compute to make a co-occurrence matrix. Also, these matrices are largely very sparse, hence this is not efficient.

It does not capture the semantic role of the context words, but simply memorises them.

No consideration for negative sampling (this is a shortcoming when compared to skip gram), and leads to worse accuracies than are possible.

Also, it is not able to understand homonymy and polysemy and does not represent those at all, which are common in at least the English language.

Shortcomings of Word2Vec

The complexity of training this is very high, and it takes a long time to run, even on GPUs.

Rare or unknown words are not handled well, and generalisation of these to multiple scenarios is challenging.

Also, it doesn't handle polysemy very well.

Conclusion

Overall, the prediction-based model outperforms the frequency-based model for word vectorization.

**