iNLP Assignment-3 Report

Akshit Sharma
2021101029

SVD

SVD is a matrix factorisation technique widely used for dimensionality reduction and feature extraction. In the context of word vectorisation, SVD operates on a term-document matrix or a word-context matrix. It decomposes the matrix into three matrices: U, Σ, and V^T, where U represents word vectors, Σ is a diagonal matrix containing singular values, and V^T contains context vectors.

Skip-gram with Negative Sampling

Skip-gram with Negative Sampling (SGNS) is a neural network-based model. It learns word embeddings by predicting the context words given a target word. Negative sampling is employed to approximate the Softmax function, enhancing training efficiency.

Analysis

Which performs better?

Based on the performance metrics like accuracy score, F1 score, precision and recall score obtained from evaluation on train and test set in the downstream task, we can conclude that SGNS performs better than SVD. For a chosen window size, the metrics we get for SGNS outperform the ones for SVD.

Comparison and Analysis

Local Context Modelling: SGNS focuses on capturing local context, i.e., the neighbouring words surrounding a target word. This allows it to capture more nuanced and fine-grained semantic relationships between words within a smaller window, leading to more accurate word embeddings. In contrast, SVD considers the entire co-occurrence matrix, which may dilute the significance of local context information, especially in larger corpora.

Performance on Semantic Tasks: SVD tends to perform better on tasks requiring a broader semantic understanding, such as document classification or information retrieval, due to its global context modelling. On the other hand, SGNS excels in tasks requiring finer semantic distinctions, such as word analogy tasks or semantic similarity evaluation, thanks to its focus on local context.

Efficiency: SGNS is generally more computationally efficient than SVD, particularly for large-scale datasets, making it preferable for real time or resource constrained applications.

Robustness: SGNS (with negative sampling) is more robust to sparse or noisy data compared to SVD, as it learns from local context and employs techniques like negative sampling to mitigate the impact of noise.

Non-Linear Relationships: SGNS, being a neural network-based approach, inherently captures non-linear relationships between words. Neural networks, through hidden layers and activation functions, can learn complex patterns and dependencies present in the data. This flexibility allows SGNS to better model the intricate semantic associations between words, including synonyms, antonyms, and analogies, which may be challenging for the linear transformation performed by SVD.

Shortcomings of SVD

- The dimensions of the matrix change very often (new words are added very frequently and corpus changes in size).
- The matrix is extremely sparse since most words do not co-occur.
- The matrix is very high dimensional in general ($\approx 106 \times 106$)
- Quadratic cost to train (i.e. to perform SVD)
- Requires the incorporation of some hacks on co-occurence matrix to account for the drastic imbalance in word frequency.

Shortcomings of SVD

- Difficult to capture global context, which may result in limited representation of global semantic relationships between words.
- **Semantic Ambiguity**: SGNS may struggle with capturing polysemous words, i.e., words with multiple meanings. Since SGNS learns embeddings based on local context alone, it might fail to disambiguate between different senses of a word, leading to less accurate representations, especially in cases where context alone is insufficient to determine the correct meaning.

- **Parameter Sensitivity**: The performance of SGNS is sensitive to hyper-parameters such as the number of negative samples, context window size, and learning rate. Choosing appropriate values for these hyper-parameters can be challenging, and suboptimal choices may result in inferior word embeddings or longer training times.
- **Limited Interpretability**: Neural network-based models like SGNS lack the interpretability of linear models such as SVD. The learned embeddings are distributed representations in a high dimensional space, making it challenging to interpret the individual dimensions or understand the underlying semantic relationships directly from the embeddings.

Hyperparameter Tuning

I have experimented with window size of: **2,5,7 and 10**

Reasons for choosing:

- Smaller window sizes (like 2) tend to capture more syntactic information as they consider only the immediate context of a word. Larger window sizes (like 10) capture more semantic information, as they encompass a broader context, including both syntactic and semantic relationships between words. This allows for a comparison of how the balance between local and global information affects performance.
- With smaller window sizes, there's a risk of missing out on relevant co-occurrences, especially in sparse data. Larger

window sizes mitigate this risk by allowing more words to be considered in the context, potentially capturing more diverse co-occurrence patterns.

- A window size of 5 or 7 allows the model to capture a mix of both syntactic and semantic information. It considers words that are nearby (within a reasonable range) to the target word, enabling the model to understand both the immediate syntactic structure and the broader semantic context in which the word appears.

Results Obtained for SVD

The hyper parameters (other than window size) remain the same, as shown. Also, the order of class index in confusion matrix across row and column is : 1,2,3,4

Window Size: 2

```
Evaluation Metrics for train set :
Accuracy Score: 0.86535
F1_Score (Macro): 0.8652140647703022
F1_Score (Micro): 0.86535
Precision Score: 0.8655695787951423
Recall Score: 0.86535
Confusion Matrix:
 [[25542  1339  1856  1263]
 [  530 28347   522   601]
 [ 1045   491 25356  3108]
 [ 1445   714  3244 24597]]
```

```
UNK_CUTOFF=3
UNKNOWN_TOKEN='<unk>'
WINDOW_SIZE=5
BATCH_SIZE=128
EMBEDDING_SIZE_SVD=300
PAD_TOKEN='<pad>'
NUM_LABELS=4
HIDDEN_SIZE=128
lrate=1e-3
EPOCHS=10
```

```
Evaluation Metrics for test set :
Accuracy Score: 0.8502631578947368
F1_Score (Macro): 0.8499983147998265
F1_Score (Micro): 0.8502631578947368
Precision Score: 0.8501728788959624
Recall Score: 0.8502631578947368
Confusion Matrix:
 [[1602   95  115   88]
 [  42 1786   38   34]
 [  82   38 1550  230]
 [  90   56  230 1524]]
```

Window size: 5

```
Evaluation Metrics for train set :
Accuracy Score: 0.8725833333333334
F1_Score (Macro): 0.8719194626963053
F1_Score (Micro): 0.8725833333333334
Precision Score: 0.8724355129886472
Recall Score: 0.8725833333333334
Confusion Matrix:
 [[25816  1464  1882   838]
 [  594 28832   294   280]
 [ 1188   644 25780  2388]
 [ 1818   919  2981 24282]]
```

```
Evaluation Metrics for test set :
Accuracy Score: 0.8565789473684211
F1_Score (Macro): 0.8559166638550271
F1_Score (Micro): 0.8565789473684211
Precision Score: 0.8565087382914338
Recall Score: 0.8565789473684211
Confusion Matrix:
 [[1618  102  131   49]
 [  52 1803   26   19]
 [  88   52 1582  178]
 [ 107   66  220 1507]]
```

Window size: 7

```
Evaluation Metrics for train set :
Accuracy Score: 0.8759833333333333
F1_Score (Macro): 0.8757692587965115
F1_Score (Micro): 0.8759833333333333
Precision Score: 0.8777963602869505
Recall Score: 0.8759833333333333
Confusion Matrix:
 [[26022  1166  1415  1397]
 [  558 28523   191   728]
 [ 1272   472 23946  4310]
 [ 1361   496  1516 26627]]
```

```
Evaluation Metrics for test set :
Accuracy Score: 0.8602631578947368
F1_Score (Macro): 0.859842356834317
F1_Score (Micro): 0.8602631578947368
Precision Score: 0.8619504569432361
Recall Score: 0.8602631578947368
Confusion Matrix:
 [[1628   75   96  101]
 [  47 1791   21   41]
 [ 105   36 1457  302]
 [  85   34  119 1662]]
```

Window size: 10

```
Evaluation Metrics for train set :
Accuracy Score: 0.8398083333333334
F1_Score (Macro): 0.8421750992232582
F1_Score (Micro): 0.8398083333333334
Precision Score: 0.853340295422289
Recall Score: 0.8398083333333334
Confusion Matrix:
 [[23318   964  4527  1191]
 [  540 26679  1909   872]
 [  517   212 26511  2760]
 [  794   318  4619 24269]]
```

```
Evaluation Metrics for test set :
Accuracy Score: 0.8315789473684211
F1_Score (Macro): 0.8340643484334755
F1_Score (Micro): 0.8315789473684211
Precision Score: 0.8455826725254686
Recall Score: 0.8315789473684211
Confusion Matrix:
 [[1463   68  285   84]
 [  41 1682  122   55]
 [  33   15 1661  191]
 [  47   22  317 1514]]
```

Observations and Reasoning:

We see increasing train and test set accuracy as window size increases from 2 to 7 and the accuracy score is least for window size of 10. With smaller window sizes (like 2), the context window might be too restrictive, leading to a loss of contextual information. As the window size increases to 7, the model has a larger window to consider more contextual information, allowing it to capture a wider range of word co-occurrences and improving accuracy. Intermediate window sizes like 5 or 7 strike a balance between capturing local syntactic information and broader semantic context. This balanced context might lead to better generalisation, resulting in higher accuracy on both train and test sets. With a larger window size (like 10), the model might start capturing too much noise or irrelevant information from the broader context, leading to overfitting on the training data. This overfitting could result in decreased generalisation performance, causing lower accuracy on the test set compared to smaller window sizes.

Results obtained for SGNS

The hyper parameters (other than window size) remain the same, as shown. Also, the order of class index in confusion matrix across row and column is : 1,2,3,4

Window Size: 2

```
Evaluation Metrics for train set :
Accuracy Score: 0.9050833333333334
F1_Score (Macro): 0.904924342921531
F1_Score (Micro): 0.9050833333333334
Precision Score: 0.9072833181769586
Recall Score: 0.9050833333333334
Confusion Matrix:
 [[25664  1241  2408   687]
 [  138 29586   197    79]
 [  406   326 27317  1951]
 [  817   412  2728 26043]]
```

```
UNK_CUTOFF=3
UNKNOWN_TOKEN='<unk>'
WINDOW_SIZE=5
BATCH_SIZE=128
EMBEDDING_SIZE=150
EMBEDDING_SIZE_SGNS=300
PAD_TOKEN='<pad>'
NUM_LABELS=4
HIDDEN_SIZE=128
lrate=1e-3
NEG_SAMPLES=4
EPOCHS=10
THRESHOLD=1e-5
```

```
Evaluation Metrics for test set :
Accuracy Score: 0.8942105263157895
F1_Score (Macro): 0.8941204133282613
F1_Score (Micro): 0.8942105263157893
Precision Score: 0.8969935514892636
Recall Score: 0.8942105263157895
Confusion Matrix:
 [[1613   85  166   36]
 [  10 1864   22    4]
 [  32   27 1707  134]
 [  59   31  198 1612]]
```

Window Size: 5

```
Evaluation Metrics for train set :
Accuracy Score: 0.9244333333333333
F1_Score (Macro) 0.924246920056135
F1_Score (Micro) 0.9244333333333333
Precision Score: 0.9241405261810478
Recall Score: 0.9244333333333333
Confusion Matrix:
 [[27518   802  1008   672]
 [  293 29514    79   114]
 [ 1071   198 26872  1859]
 [ 1011   150  1811 27028]]
```

```
Evaluation Metrics for test set :
Accuracy Score: 0.9025
F1_Score (Macro) 0.9022434220605823
F1_Score (Micro) 0.9025
Precision Score: 0.9021187370108664
Recall Score: 0.9025
Confusion Matrix:
 [[1721   54   73   52]
 [  34 1846   14    6]
 [  91   17 1642  150]
 [  78   21  151 1650]]
```

Window Size: 10

```
Evaluation Metrics for train set :
Accuracy Score: 0.9171
F1_Score (Macro): 0.9170135332237116
F1_Score (Micro): 0.9171000000000001
Precision Score: 0.9179787143821101
Recall Score: 0.9171
Confusion Matrix:
 [[26350   949  1717   984]
 [  155 29558   145   142]
 [  497   258 26828  2417]
 [  592   200  1892 27316]]
```

```
Evaluation Metrics for test set :
Accuracy Score: 0.895921052631579
F1_Score (Macro): 0.8959230756407431
F1_Score (Micro): 0.895921052631579
Precision Score: 0.8973573358817423
Recall Score: 0.895921052631579
Confusion Matrix:
 [[1630   69  132   69]
 [  13 1854   19   14]
 [  38   26 1644  192]
 [  40   21  158 1681]]
```

Window Size: 7

```
Evaluation Metrics for train set :
Accuracy Score: 0.9181
F1_Score (Macro): 0.9179069435733539
F1_Score (Micro): 0.9181
Precision Score: 0.9183180051347483
Recall Score: 0.9181
Confusion Matrix:
 [[26768   986  1309   937]
 [  154 29550   153   143]
 [  733   234 26460  2573]
 [  809   211  1586 27394]]
```

```
Evaluation Metrics for test set :
Accuracy Score: 0.9001315789473684
F1_Score (Macro): 0.8999737045352665
F1_Score (Micro): 0.9001315789473684
Precision Score: 0.900533865790101
Recall Score: 0.9001315789473684
Confusion Matrix:
 [[1665   70   98   67]
 [  13 1852   24   11]
 [  55   18 1627  200]
 [  55   21  127 1697]]
```

Observations and Reasoning:

We see that the accuracy score obtained for train and test set evaluation is highest for window size of 5, lowest for 2 and decreases with increasing window size after 5. For this specific dataset and task, a window size of 5 might provide an optimal balance between capturing local syntactic information and broader semantic context. This balance allows the model to learn meaningful word embeddings that generalise well to unseen data, resulting in higher accuracy compared to a smaller window size like 2. With a window size of 7 and above, the model may start capturing more noise or irrelevant information from the broader context, leading to overfitting on the training data. This overfitting could result in decreased generalisation performance, causing lower accuracy compared to window sizes that strike a better balance between context size and noise reduction.