INLP ASSIGNMENT 3

- Used first 20,000 sentences from training data to get word embeddings using SVD and whole training corpus for skip-gram.
- Tokens with frequency less than 3 are replaced by “<unk>”.
- Embedding dimension is set to 300.
- For downstream task, the LSTM is trained on the entire train corpus.
- Used bi-directional LSTM for downstream task with following hyperparameters:

```
input_dim = 300
hidden_dim = 128
n_layers = 2
bidirectional = True
n_epochs = 10
lr = 0.001
output_dim = n_classes
```

- 90% of the sentence lengths fall under 42, so max_len is set to 42. Sentences with length less than max_len are padded using <pad> tokens whereas sentences with length greater than max_len are truncated.
- No.of negative samples per positive sample K is set to 3 in SGNS.

Hyperparameter Tuning

Experimented with context window sizes 1,2,3,4 and 5 for both SVD and SGNS, while keeping other parameters constant.

SVD:

1. window_size = 1

```
Epoch 1/10, Train Loss: 0.9238011894424757, Val Loss: 0.6881705433924993
Epoch 2/10, Train Loss: 0.5933364555289348, Val Loss: 0.6124802518884341
Epoch 3/10, Train Loss: 0.5544873491674661, Val Loss: 0.5783113085230192
Epoch 4/10, Train Loss: 0.5197744686653216, Val Loss: 0.5658115209142367
Epoch 5/10, Train Loss: 0.492266666414837, Val Loss: 0.5426506902575493
Epoch 6/10, Train Loss: 0.46843461809804043, Val Loss: 0.5371860260168712
Epoch 7/10, Train Loss: 0.4437082229355971, Val Loss: 0.5391194501717885
Epoch 8/10, Train Loss: 0.41817417736848195, Val Loss: 0.49680688554048535
Epoch 9/10, Train Loss: 0.39727895410358904, Val Loss: 0.489883018275102
Epoch 10/10, Train Loss: 0.37587050294627744, Val Loss: 0.4792905680735906
```

Metrics on train, val and test data:

```
Train Accuracy: 0.8711, F1: 0.8712, Precision: 0.8765, Recall: 0.8711
Val Accuracy: 0.8190, F1: 0.8192, Precision: 0.8320, Recall: 0.8190
Test Accuracy: 0.8346, F1: 0.8343, Precision: 0.8405, Recall: 0.8346
Train Confusion Matrix:
[[20891   902   761  1620]
 [  499 22329   123   753]
 [  883   362 18453  4062]
 [  897   499  1017 21949]]
Val Confusion Matrix:
[[4796  245  219  566]
 [ 197 5707   65  327]
 [ 360  226 4225 1429]
 [ 232  173  306 4927]]
Test Confusion Matrix:
[[1594   99   76  131]
 [  65 1750   13   72]
 [  96   43 1357  404]
 [  70   60  128 1642]]
```

2. window_size = 2

```
Epoch 1/10, Train Loss: 1.3865375507672628, Val Loss: 1.3867773461341857
Epoch 2/10, Train Loss: 1.3857686748107274, Val Loss: 1.387363123099009
Epoch 3/10, Train Loss: 1.326431392788887, Val Loss: 0.9755481197039286
Epoch 4/10, Train Loss: 0.7737012387166421, Val Loss: 0.6159244533379873
Epoch 5/10, Train Loss: 0.5377315277457237, Val Loss: 0.629646807650725
Epoch 6/10, Train Loss: 0.4909296332498391, Val Loss: 0.5591899106105168
Epoch 7/10, Train Loss: 0.4687520682240526, Val Loss: 0.5298455594579379
Epoch 8/10, Train Loss: 0.43976243899265927, Val Loss: 0.5038758487304051
Epoch 9/10, Train Loss: 0.4183095578402281, Val Loss: 0.47615756301085155
Epoch 10/10, Train Loss: 0.3961295291533073, Val Loss: 0.4813737645347913
```

Metrics on train, val and test data:

```
Train Accuracy: 0.8630, F1: 0.8632, Precision: 0.8655, Recall: 0.8630
Val Accuracy: 0.8222, F1: 0.8228, Precision: 0.8266, Recall: 0.8222
Test Accuracy: 0.8424, F1: 0.8427, Precision: 0.8449, Recall: 0.8424
Train Confusion Matrix:
[[19800   893  2151  1330]
 [  448 22442   384   430]
 [  526   330 20534  2370]
 [  847   618  2826 20071]]
Val Confusion Matrix:
[[4474  244  648  460]
 [ 153 5744  201  198]
 [ 188  176 4965  911]
 [ 195  209  684 4550]]
Test Confusion Matrix:
[[1523   81  190  106]
 [  58 1760   40   42]
 [  60   37 1580  223]
 [  62   58  241 1539]]
```

3. window_size = 3

```
Epoch 1/10, Train Loss: 0.8367647318392992, Val Loss: 0.6439334202607473
Epoch 2/10, Train Loss: 0.5467411652157704, Val Loss: 0.5530718187888464
Epoch 3/10, Train Loss: 0.49865740072230497, Val Loss: 0.5175601089795431
Epoch 4/10, Train Loss: 0.4678936263124148, Val Loss: 0.5238705696662267
Epoch 5/10, Train Loss: 0.44854912721614043, Val Loss: 0.49330101521809894
Epoch 6/10, Train Loss: 0.423995362897714, Val Loss: 0.4980183209280173
Epoch 7/10, Train Loss: 0.40415130964666607, Val Loss: 0.4467369435330232
Epoch 8/10, Train Loss: 0.3839994620680809, Val Loss: 0.44511997828880945
Epoch 9/10, Train Loss: 0.36377523505066833, Val Loss: 0.44215321079889935
Epoch 10/10, Train Loss: 0.3478716725918154, Val Loss: 0.4335529350042343
```

Metrics on train,val and test data:

```
Train Accuracy: 0.8831, F1: 0.8831, Precision: 0.8841, Recall: 0.8831
Val Accuracy: 0.8357, F1: 0.8354, Precision: 0.8389, Recall: 0.8357
Test Accuracy: 0.8530, F1: 0.8526, Precision: 0.8544, Recall: 0.8530
Train Confusion Matrix:
[[22033   523   754   864]
 [  967 22207   156   374]
 [ 1208   276 19574  2702]
 [ 1472   432  1490 20968]]
Val Confusion Matrix:
[[5137  148  237  304]
 [ 331 5689  103  173]
 [ 477  169 4578 1016]
 [ 373  168  444 4653]]
Test Confusion Matrix:
[[1714   47   62   77]
 [ 107 1743   21   29]
 [ 139   37 1451  273]
 [ 120   47  158 1575]]
```

4. window_size = 4

```
Epoch 1/10, Train Loss: 0.8448175813257695, Val Loss: 0.6622831749121348
Epoch 2/10, Train Loss: 0.5538933908989032, Val Loss: 0.5657155467073123
Epoch 3/10, Train Loss: 0.49001892496148747, Val Loss: 0.4901344377597173
Epoch 4/10, Train Loss: 0.46167230762292943, Val Loss: 0.501315916677316
Epoch 5/10, Train Loss: 0.43681480149676405, Val Loss: 0.4689821786880493
Epoch 6/10, Train Loss: 0.41066658062736194, Val Loss: 0.4748291460474332
Epoch 7/10, Train Loss: 0.38888982249299686, Val Loss: 0.4505798563261827
Epoch 8/10, Train Loss: 0.3678824068556229, Val Loss: 0.576694535712401
Epoch 9/10, Train Loss: 0.3524625619351864, Val Loss: 0.42317419447501503
Epoch 10/10, Train Loss: 0.3303551925184826, Val Loss: 0.4267711200912793
```

Metrics on train,val and test data:

```
Train Accuracy: 0.8900, F1: 0.8901, Precision: 0.8944, Recall: 0.8900
Val Accuracy: 0.8377, F1: 0.8379, Precision: 0.8494, Recall: 0.8377
Test Accuracy: 0.8595, F1: 0.8594, Precision: 0.8647, Recall: 0.8595
Train Confusion Matrix:
[[20807   783   706  1878]
 [  302 22895   106   401]
 [  677   410 19404  3269]
 [  580   538   908 22336]]
Val Confusion Matrix:
[[4758  242  212  614]
 [ 117 5901   72  206]
 [ 268  234 4437 1301]
 [ 146  204  278 5010]]
Test Confusion Matrix:
[[1603   71   64  162]
 [  48 1799   15   38]
 [  71   52 1448  329]
 [  45   61  112 1682]]
```

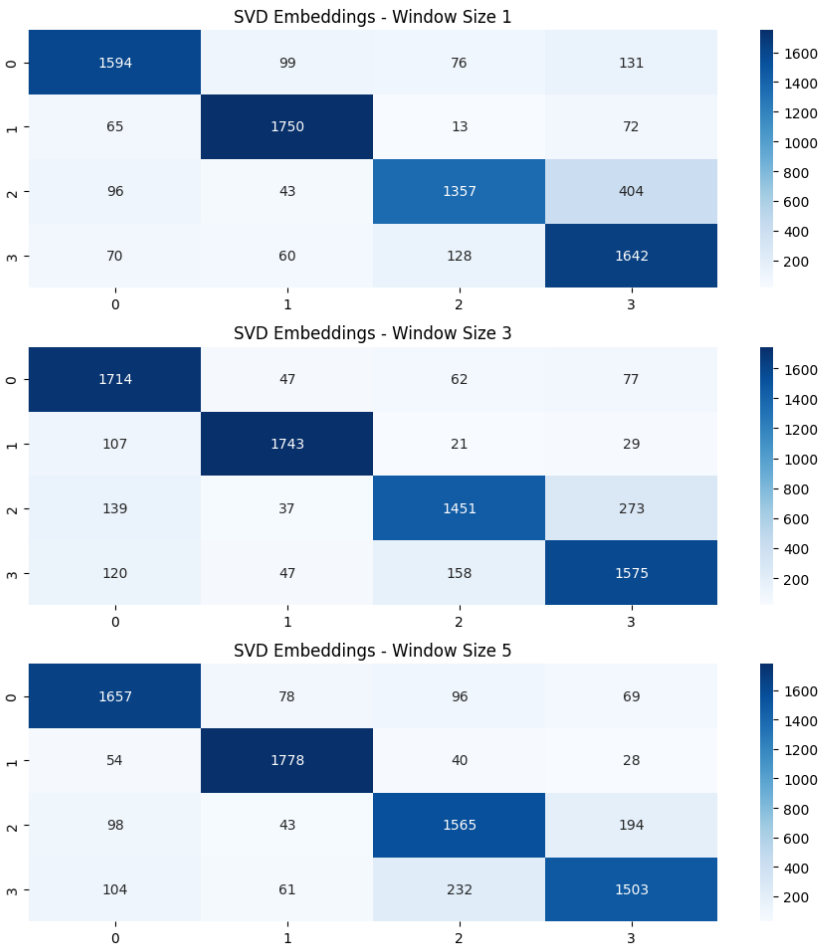
5. window_size = 5

```
Epoch 1/10, Train Loss: 1.2490709602137406, Val Loss: 1.300754476706187
Epoch 2/10, Train Loss: 1.3556280295054117, Val Loss: 1.365399753411611
Epoch 3/10, Train Loss: 1.3769077394803364, Val Loss: 1.3847289667129516
Epoch 4/10, Train Loss: 1.3836114383538565, Val Loss: 1.3791054145495096
Epoch 5/10, Train Loss: 1.028744673460722, Val Loss: 0.628220422744751
Epoch 6/10, Train Loss: 0.5047161530156931, Val Loss: 0.4800482624967893
Epoch 7/10, Train Loss: 0.4354155140593648, Val Loss: 0.47585027970870336
Epoch 8/10, Train Loss: 0.4021103948329886, Val Loss: 0.4399717993537585
Epoch 9/10, Train Loss: 0.37773877624422314, Val Loss: 0.42604168633619943
Epoch 10/10, Train Loss: 0.3574291691556573, Val Loss: 0.41052693366010984
```

Metrics on train,val and test data:

```
Train Accuracy: 0.8830, F1: 0.8827, Precision: 0.8829, Recall: 0.8830
Val Accuracy: 0.8480, F1: 0.8478, Precision: 0.8479, Recall: 0.8480
Test Accuracy: 0.8557, F1: 0.8552, Precision: 0.8552, Recall: 0.8557
Train Confusion Matrix:
[[21366   833  1089   886]
 [  432 22700   301   271]
 [  918   239 20636  1967]
 [ 1214   508  2574 20066]]
Val Confusion Matrix:
[[5000  215  308  303]
 [ 167 5832  164  133]
 [ 376  157 4946  761]
 [ 292  159  613 4574]]
Test Confusion Matrix:
[[1657   78   96   69]
 [  54 1778   40   28]
 [  98   43 1565  194]
 [ 104   61  232 1503]]
```

Confusion Matrix for SVD:



Skip-gram with negative sampling:

1. window_size = 1

```
Epoch 1/10, Train Loss: 0.7397350309863686, Val Loss: 0.4103415648837884
Epoch 2/10, Train Loss: 0.3022068803831935, Val Loss: 0.33677122673889004
Epoch 3/10, Train Loss: 0.20985974161388973, Val Loss: 0.368316494256258
Epoch 4/10, Train Loss: 0.14792790957229832, Val Loss: 0.3776210038661957
Epoch 5/10, Train Loss: 0.10680290064231182, Val Loss: 0.4284280130888025
Epoch 6/10, Train Loss: 0.08410987315008728, Val Loss: 0.4500953996976217
Epoch 7/10, Train Loss: 0.06738389002147596, Val Loss: 0.47779922564079363
Epoch 8/10, Train Loss: 0.05901920721672165, Val Loss: 0.49290918962657454
Epoch 9/10, Train Loss: 0.049305214996158614, Val Loss: 0.5421957424680391
Epoch 10/10, Train Loss: 0.044558977355191015, Val Loss: 0.557301673065871
```

Metrics on train,val and test data:

```
Train Accuracy: 0.9912, F1: 0.9912, Precision: 0.9913, Recall: 0.9912
Val Accuracy: 0.8666, F1: 0.8673, Precision: 0.8702, Recall: 0.8666
Test Accuracy: 0.8824, F1: 0.8827, Precision: 0.8838, Recall: 0.8824
Train Confusion Matrix:
[[23848    26   156   144]
 [   41 23605    23    35]
 [   29     7 23539   185]
 [   27     3   167 24165]]
Val Confusion Matrix:
[[4876  140  369  441]
 [ 108 5918  114  156]
 [ 239   77 5079  845]
 [ 185   61  466 4926]]
Test Confusion Matrix:
[[1656   46   98  100]
 [  38 1796   27   39]
 [  69   14 1590  227]
 [  54   25  157 1664]]
```

2. window_size = 2

```
Epoch 1/10, Train Loss: 0.6505382782742382, Val Loss: 0.4165544869800409
Epoch 2/10, Train Loss: 0.3010971236849825, Val Loss: 0.35678951513767243
Epoch 3/10, Train Loss: 0.209426248965164, Val Loss: 0.3452133706212044
Epoch 4/10, Train Loss: 0.14643550544977188, Val Loss: 0.36716558135549227
Epoch 5/10, Train Loss: 0.10482399648784971, Val Loss: 0.3949331919848919
Epoch 6/10, Train Loss: 0.08137315982099001, Val Loss: 0.4309458295603593
Epoch 7/10, Train Loss: 0.06397419914416969, Val Loss: 0.5110007557285329
Epoch 8/10, Train Loss: 0.05411557350002113, Val Loss: 0.5254167571937044
Epoch 9/10, Train Loss: 0.047344972590430794, Val Loss: 0.5418796133361756
Epoch 10/10, Train Loss: 0.04237167195799217, Val Loss: 0.5636056760844464
```

Metrics on train,val and test data:

```
Train Accuracy: 0.9915, F1: 0.9915, Precision: 0.9915, Recall: 0.9915
Val Accuracy: 0.8695, F1: 0.8692, Precision: 0.8699, Recall: 0.8695
Test Accuracy: 0.8892, F1: 0.8890, Precision: 0.8890, Recall: 0.8892
Train Confusion Matrix:
[[23900    84   100    90]
 [   17 23677     8     2]
 [   50    26 23527   157]
 [   57    41   186 24078]]
Val Confusion Matrix:
[[4980  220  308  318]
 [  95 6030   77   94]
 [ 305  148 5035  752]
 [ 224  139  451 4824]]
Test Confusion Matrix:
[[1688   58   77   77]
 [  25 1833   16   26]
 [  72   31 1599  198]
 [  68   41  153 1638]]
```

3. window_size = 3

```
Epoch 1/10, Train Loss: 0.708465016067028, Val Loss: 0.44559928023815154
Epoch 2/10, Train Loss: 0.33011516322692236, Val Loss: 0.34748461078604065
Epoch 3/10, Train Loss: 0.23108584584109484, Val Loss: 0.3743716403692961
Epoch 4/10, Train Loss: 0.16067933715259036, Val Loss: 0.37689049521585305
Epoch 5/10, Train Loss: 0.11424359449030211, Val Loss: 0.4155543219447136
Epoch 6/10, Train Loss: 0.08416584554407745, Val Loss: 0.4639181452294191
Epoch 7/10, Train Loss: 0.06713667537795845, Val Loss: 0.5101136630711456
Epoch 8/10, Train Loss: 0.05610245911239569, Val Loss: 0.49229884425426523
Epoch 9/10, Train Loss: 0.049695050051852985, Val Loss: 0.5224997405894101
Epoch 10/10, Train Loss: 0.04392968681532269, Val Loss: 0.546423682782799
```

Metrics on train,val and test data:

```
Train Accuracy: 0.9911, F1: 0.9911, Precision: 0.9911, Recall: 0.9911
Val Accuracy: 0.8641, F1: 0.8643, Precision: 0.8668, Recall: 0.8641
Test Accuracy: 0.8880, F1: 0.8879, Precision: 0.8889, Recall: 0.8880
Train Confusion Matrix:
[[24012    29    47    86]
 [  121 23550    21    12]
 [   94    16 23362   288]
 [   41    11    92 24218]]
Val Confusion Matrix:
[[5056  148  248  374]
 [ 206 5879   83  128]
 [ 376   97 4898  869]
 [ 252   98  382 4906]]
Test Confusion Matrix:
[[1716   41   61   82]
 [  51 1806   15   28]
 [  84   26 1557  233]
 [  82   31  117 1670]]
```

4. window_size = 4

```
Epoch 1/10, Train Loss: 0.7589104712357123, Val Loss: 0.5089001317620278
Epoch 2/10, Train Loss: 0.37502379511048395, Val Loss: 0.38068418723344805
Epoch 3/10, Train Loss: 0.2718186924209197, Val Loss: 0.35859786182641984
Epoch 4/10, Train Loss: 0.1966735929880912, Val Loss: 0.3773839933772882
Epoch 5/10, Train Loss: 0.14108167534849295, Val Loss: 0.4075242143149177
Epoch 6/10, Train Loss: 0.10489747565604436, Val Loss: 0.4454908497557044
Epoch 7/10, Train Loss: 0.08148934991867282, Val Loss: 0.4726647880052527
Epoch 8/10, Train Loss: 0.06605881202521656, Val Loss: 0.5232370049692691
Epoch 9/10, Train Loss: 0.056760409677110144, Val Loss: 0.5741058316907535
Epoch 10/10, Train Loss: 0.05155688165824783, Val Loss: 0.5750190240535885
```

Metrics on train,val and test data:

```
Train Accuracy: 0.9893, F1: 0.9893, Precision: 0.9894, Recall: 0.9893
Val Accuracy: 0.8622, F1: 0.8623, Precision: 0.8657, Recall: 0.8622
Test Accuracy: 0.8774, F1: 0.8773, Precision: 0.8786, Recall: 0.8774
Train Confusion Matrix:
[[23991    45    57    81]
 [   69 23605    12    18]
 [  152     9 23287   312]
 [  151    13   104 24094]]
Val Confusion Matrix:
[[5146  121  234  325]
 [ 174 5896   71  155]
 [ 439   82 4781  938]
 [ 319   78  371 4870]]
Test Confusion Matrix:
[[1717   41   67   75]
 [  57 1785   18   40]
 [ 104   18 1529  249]
 [ 106   26  131 1637]]
```

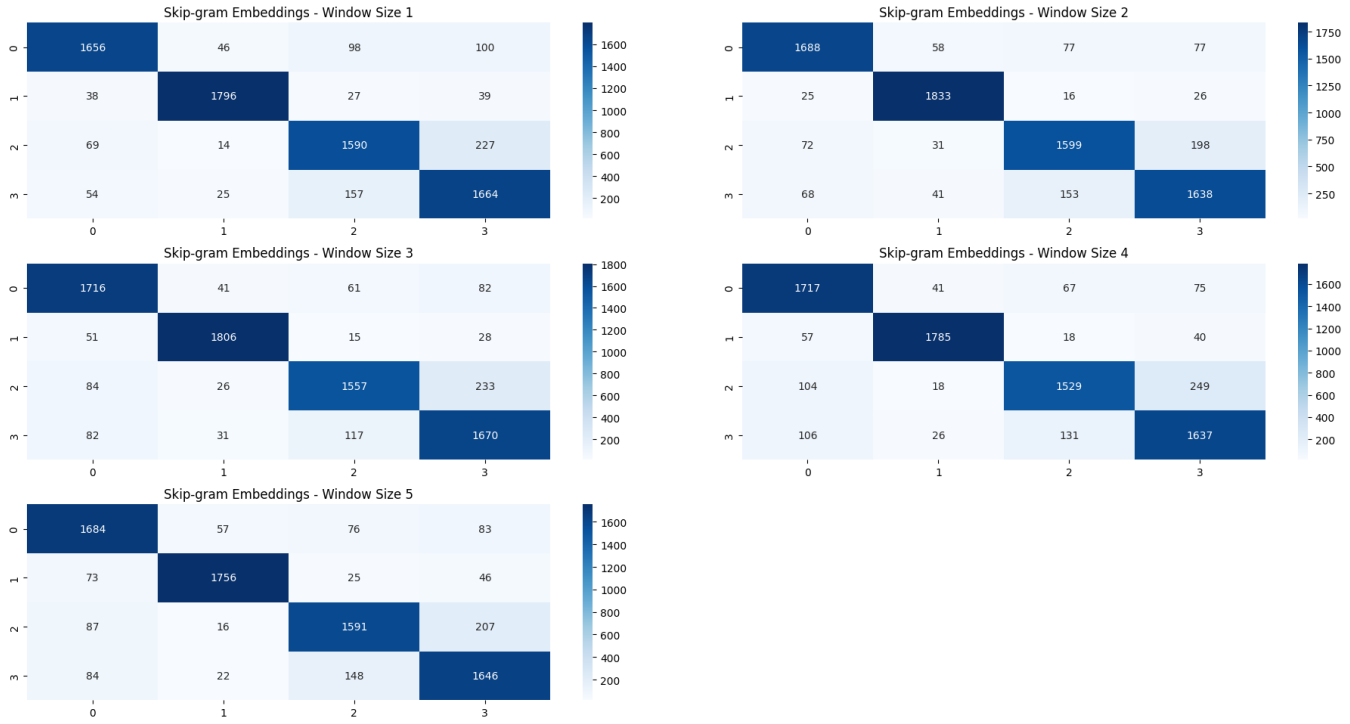
5. window_size = 5

```
Epoch 1/10, Train Loss: 0.8628407831539711, Val Loss: 0.5256161418159803
Epoch 2/10, Train Loss: 0.4035567385231455, Val Loss: 0.4065963141620159
Epoch 3/10, Train Loss: 0.28792482609425984, Val Loss: 0.3792282610088587
Epoch 4/10, Train Loss: 0.20839875359646975, Val Loss: 0.3894744526197513
Epoch 5/10, Train Loss: 0.1465983435229088, Val Loss: 0.43021492673953377
Epoch 6/10, Train Loss: 0.10925466901514058, Val Loss: 0.4771103011717399
Epoch 7/10, Train Loss: 0.08356482195629117, Val Loss: 0.4720266615773241
Epoch 8/10, Train Loss: 0.06775031995129151, Val Loss: 0.5486564802825451
Epoch 9/10, Train Loss: 0.05891942900636544, Val Loss: 0.5381950335701307
Epoch 10/10, Train Loss: 0.050471640179496416, Val Loss: 0.5941881098896264
```

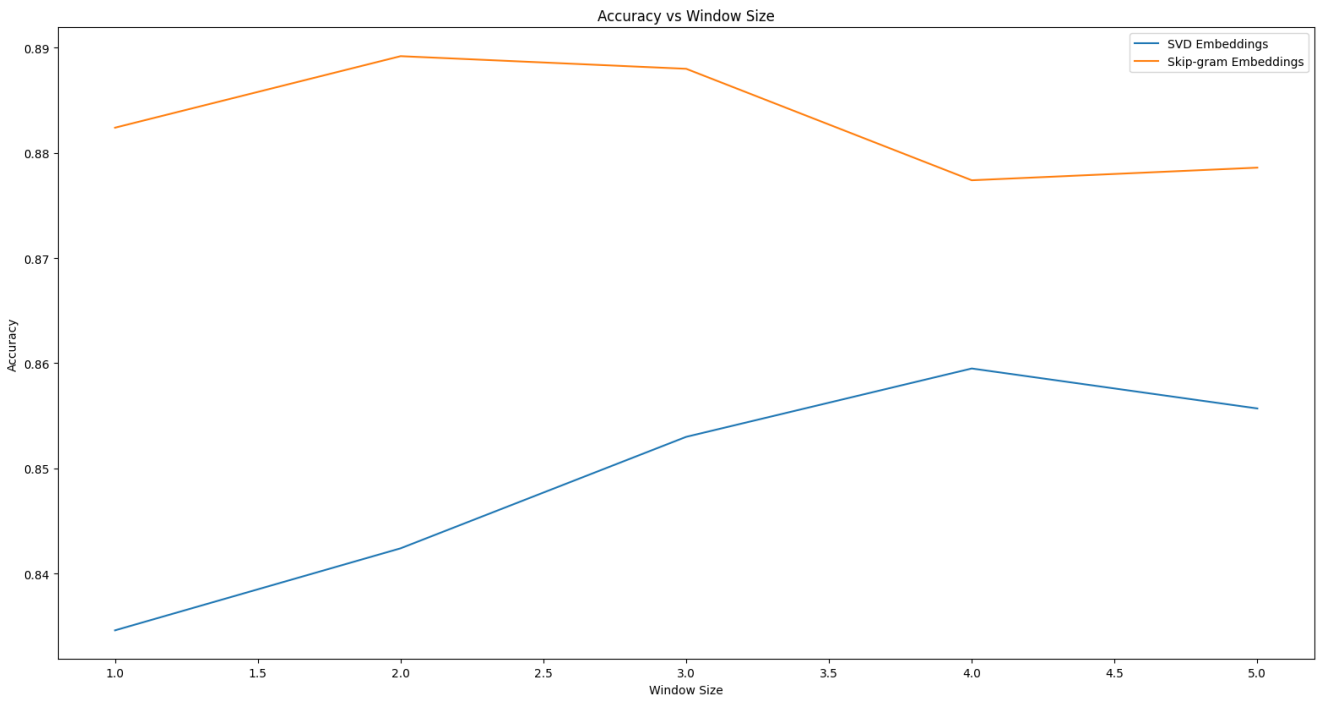
Metrics on train,val and test data:

```
Train Accuracy: 0.9900, F1: 0.9900, Precision: 0.9900, Recall: 0.9900
Val Accuracy: 0.8571, F1: 0.8576, Precision: 0.8598, Recall: 0.8571
Test Accuracy: 0.8786, F1: 0.8788, Precision: 0.8794, Recall: 0.8786
Train Confusion Matrix:
[[23928    36    73   137]
 [  128 23488    43    45]
 [   72     7 23448   233]
 [   58    10   118 24176]]
Val Confusion Matrix:
[[4968  184  285  389]
 [ 267 5734  137  158]
 [ 322   94 4975  849]
 [ 214   90  441 4893]]
Test Confusion Matrix:
[[1684   57   76   83]
 [  73 1756   25   46]
 [  87   16 1591  206]
 [  84   22  148 1646]]
```

Confusion Matrix for SGNS:



ANALYSIS



The above graph is plotted between test accuracy vs window size for SVD and SGNS.

- In case of SVD, accuracy increases with window size till 4 and then decreases for 5.
- In case of SGNS, highest accuracy is observed with context window 2. (2>3>1>5>4)

Reasons:

- Initially, as the window size increases, the accuracy of SVD also increases. This suggests that a larger context window allows SVD to capture more semantic information from the text, leading to better performance in certain cases.
- However, after reaching a window size of 4, the accuracy starts to decrease. This could indicate that beyond a certain window size, the model starts to incorporate too much irrelevant context, leading to noise and decreased performance.
- Unlike SVD, the accuracy of SGNS peaks at a smaller window size of 2. This indicates that SGNS benefits more from a narrower context window.

Observations:

Skip gram with negative sampling performs better than SVD.

Reasons why SGNS outperforms SVD:

1. **Efficiency**: SGNS is generally more computationally efficient compared to SVD, especially when dealing with large datasets. This is because SGNS uses stochastic gradient descent (SGD) to update word vectors incrementally, making it more scalable to massive datasets. (This is the reason why only 20k sentences are used for training SVD whereas entire corpora is used for SGNS)

2. **Contextual Adaptability**: SGNS directly learns word embeddings by predicting context words given a target word. This allows it to capture finer-grained semantic and syntactic relationships between words based on their contextual usage. SVD, on the other hand, creates embeddings by performing a matrix factorization on a global word co-occurrence matrix, which may not capture as nuanced contextual information.
3. **Handling Large Vocabularies**: SGNS can handle large vocabularies more effectively than SVD. This is because SGNS uses negative sampling, where only a small subset of "negative" samples (i.e., randomly chosen words that are not contextually related) are considered during training, making it more practical for large vocabulary sizes.
4. **Sparse Data**: SGNS tends to perform better when dealing with sparse data, such as in the case of rare words or in languages with limited textual resources. This is because SGNS relies on local context windows, allowing it to capture meaningful embeddings even for less frequent words.
5. **Robustness to Noise**: SGNS is generally more robust to noisy data compared to SVD. Since SGNS learns word embeddings based on local context windows, it can effectively filter out noise in the training data and focus on learning meaningful word representations.

Shortcomings of SVD:

1. **Computational Complexity**: SVD can be computationally expensive, especially for large matrices. The time and memory requirements increase with the size of the dataset, making it less practical for very large corpora.
2. **Global Context Dependency**: SVD relies on a global word co-occurrence matrix, which means it considers all words in the corpus simultaneously. This may not capture fine-grained local context information, leading to less effective embeddings for capturing semantic nuances.
3. **Memory Intensive**: Storing the entire co-occurrence matrix and performing matrix factorization can be memory intensive, especially for large vocabularies. This can limit scalability and make it challenging to apply SVD to very large datasets.
4. **Difficulty Handling Sparse Data**: SVD may struggle with sparse data, particularly when dealing with rare words or languages with limited textual resources. Sparse matrices can lead to less accurate embeddings and require additional preprocessing or smoothing techniques.

Shortcomings of SGNS:

1. **Hyperparameter Sensitivity**: SGNS often requires careful tuning of hyperparameters such as the dimensionality of word vectors, the size of the context window, and the number of negative samples. Poor choices of hyperparameters can lead to suboptimal performance.
2. **Limited Semantic Interpretability**: While SGNS embeddings often perform well on downstream NLP tasks, the resulting vectors may not always align intuitively with human semantic understanding. Some dimensions in the embedding space may not correspond directly to interpretable linguistic features.
3. **Training Instability**: SGNS training can sometimes be unstable, especially when dealing with very large datasets or noisy training examples. Convergence may be slower, and the model might get stuck in suboptimal solutions or suffer from overfitting.
4. **Quality of Negative Samples**: The effectiveness of SGNS heavily depends on the quality of negative samples chosen during training. If the negative samples are not representative of words that do not occur together with the target word, it can lead to biased embeddings.