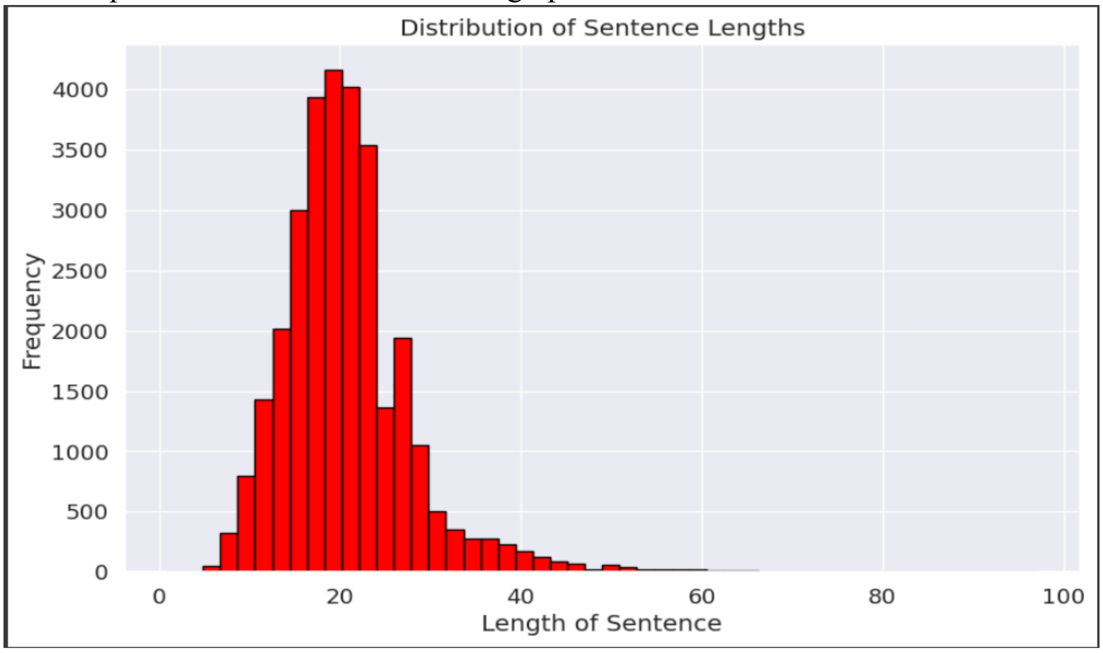
Assignment 3: Word Vectorization

Name : Udrasht Pal
Roll No: 2022201020

For this Assignment I have taken 30000 sentence of length more then equal 5 word and less then equal to 35 word as show below graph most of the sentence lie between 5-35.



Why Choose LSTM for Downstream Tasks?

Long-Term Dependencies: LSTMs are great at understanding long sequences of data, something simple RNNs struggle with because they forget important information over time.

Complex Memory Cell Structure: LSTMs have a fancy memory structure that lets them remember or forget things as needed, making them more adaptable to different types of data.

Gating Mechanisms: LSTMs use these cool gates to control the flow of information, helping them focus on what's important and ignore what's not.

Gradient Stability: LSTMs are better at dealing with problems like information disappearing or exploding during training, so they can learn more reliably from data.

Performance on Complex Tasks: When it comes to tough jobs like understanding language or recognizing speech, LSTMs usually do a better job because they're better at picking up on patterns and understanding context.

Hyperparameter tuning

In this task I use context window size:1,3,5 because I try

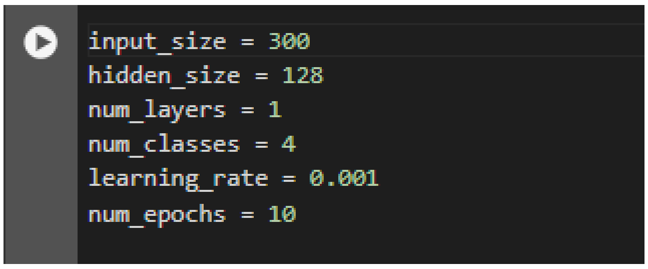
Using SVD word embeddings
Context window size=3
Embeddings Size=300

Test accuracy:83.19%

Train Accuracy: 85.90%

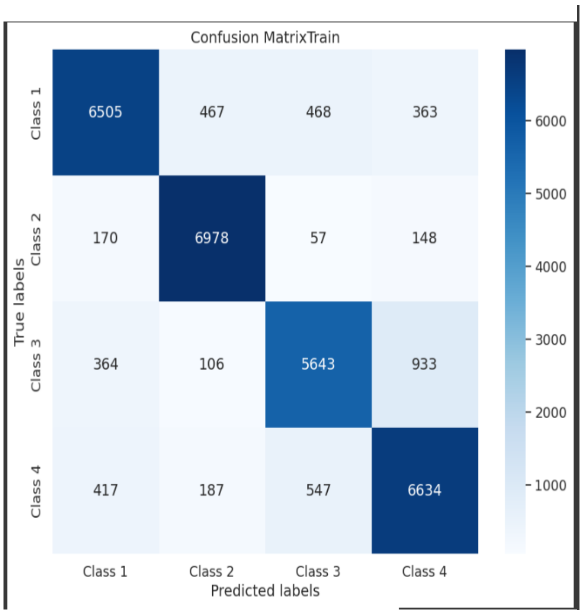
In Downstream I use RNN (LSTM)

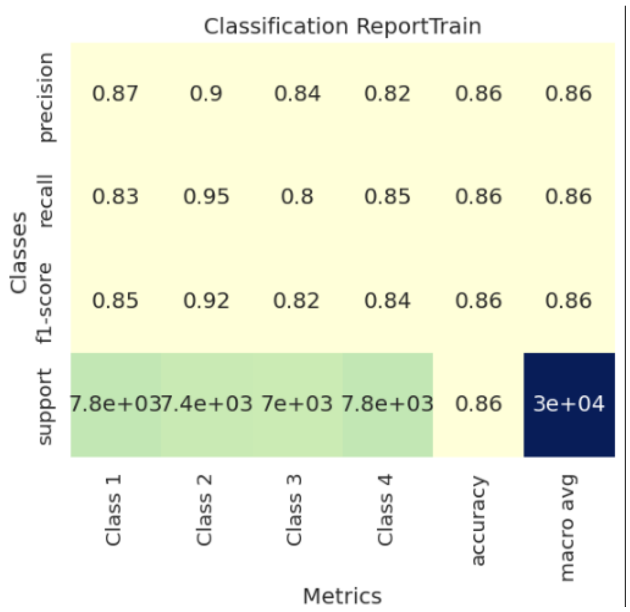
Downstream Parameters



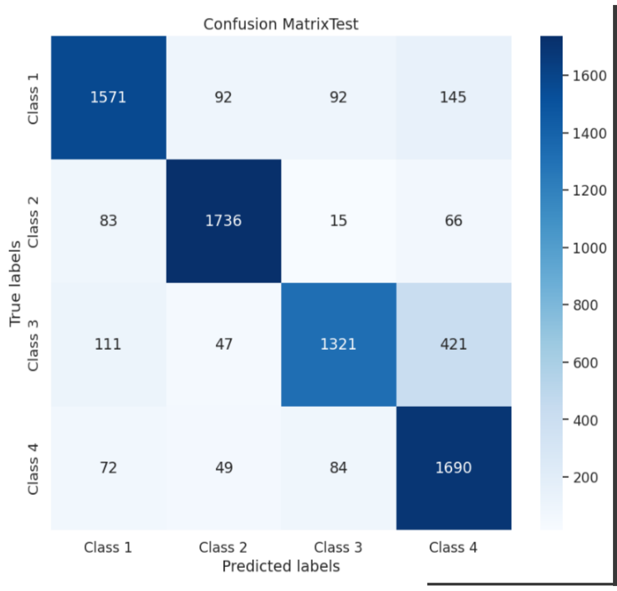
```
input_size = 300
hidden_size = 128
num_layers = 1
num_classes = 4
learning_rate = 0.001
num_epochs = 10
```

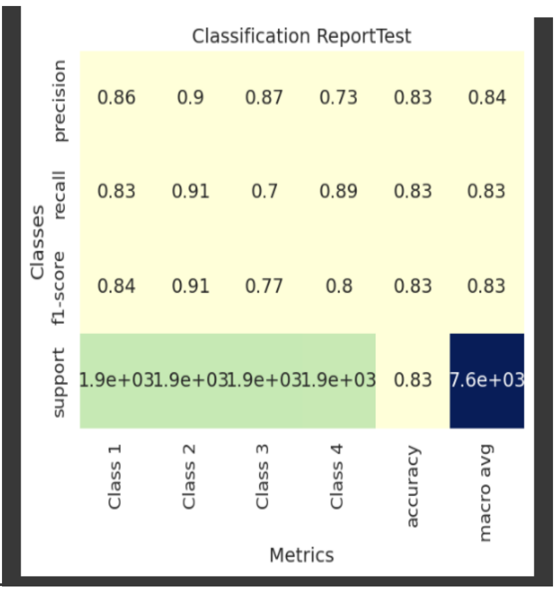
confusion matrix and Classification Report of Train data:





confusion matrix and Classification Report of Test data:





Using Skip-Gram word embeddings

Context window size=3

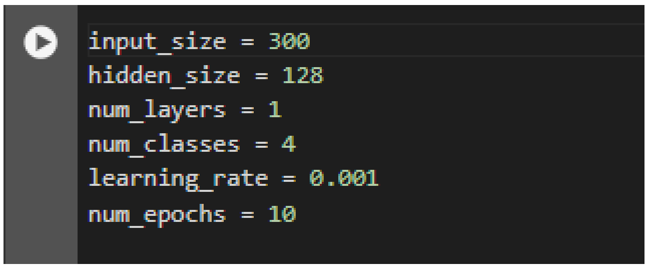
Embeddings Size=300

Test accuracy: 84.5%

Train Accuracy: 90.1%

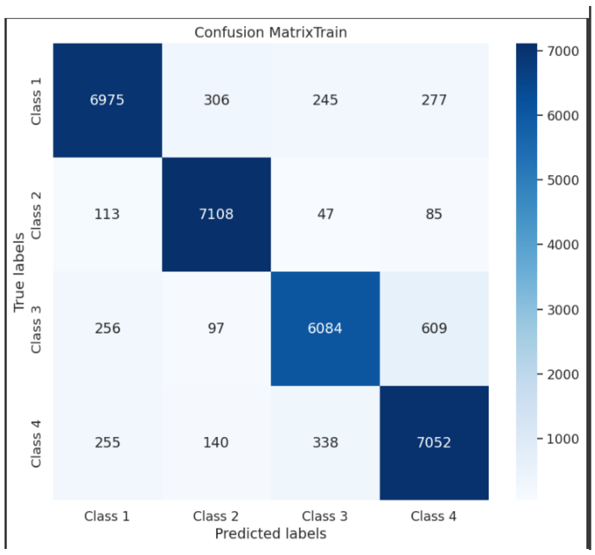
In Downstream I use RNN (LSTM)

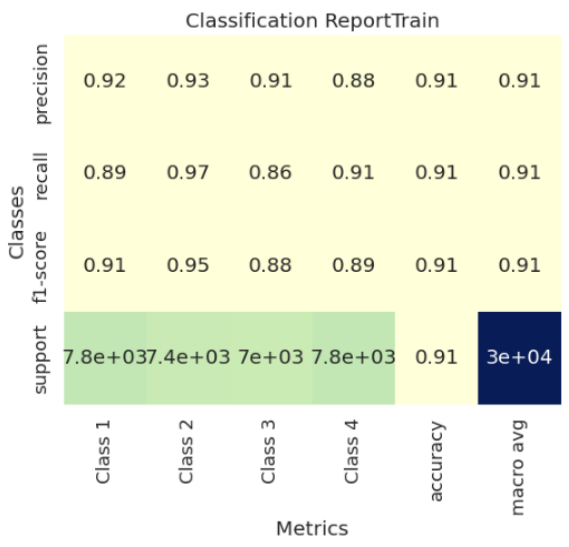
Downstream Parameters



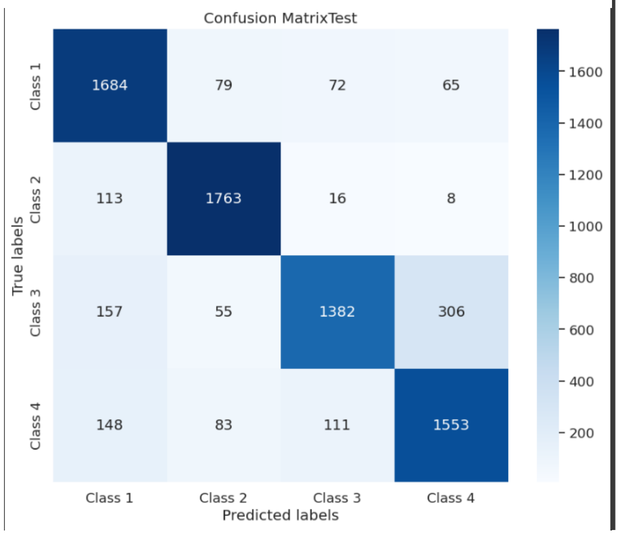
```
input_size = 300
hidden_size = 128
num_layers = 1
num_classes = 4
learning_rate = 0.001
num_epochs = 10
```

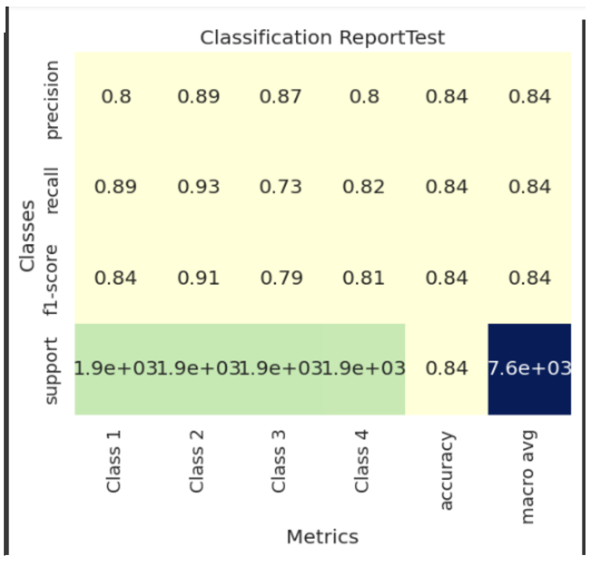
confusion matrix and Classification Report of Train data:





confusion matrix and Classification Report of Test data:





Using SVD word embeddings

Context window size=5

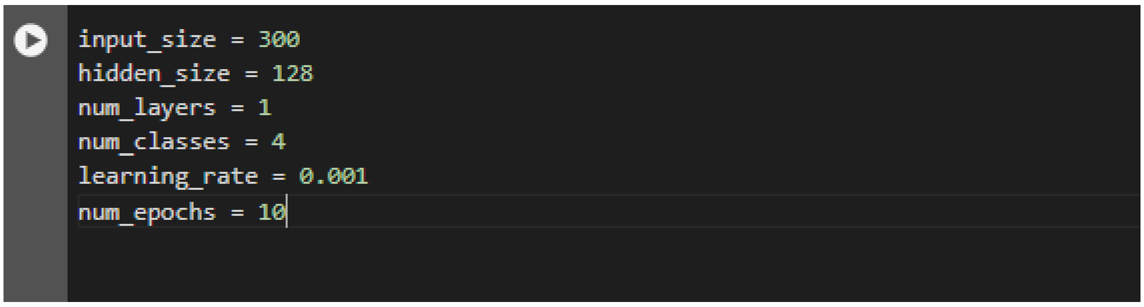
Embeddings Size=300

Test accuracy:82%

Train Accuracy:86.2%

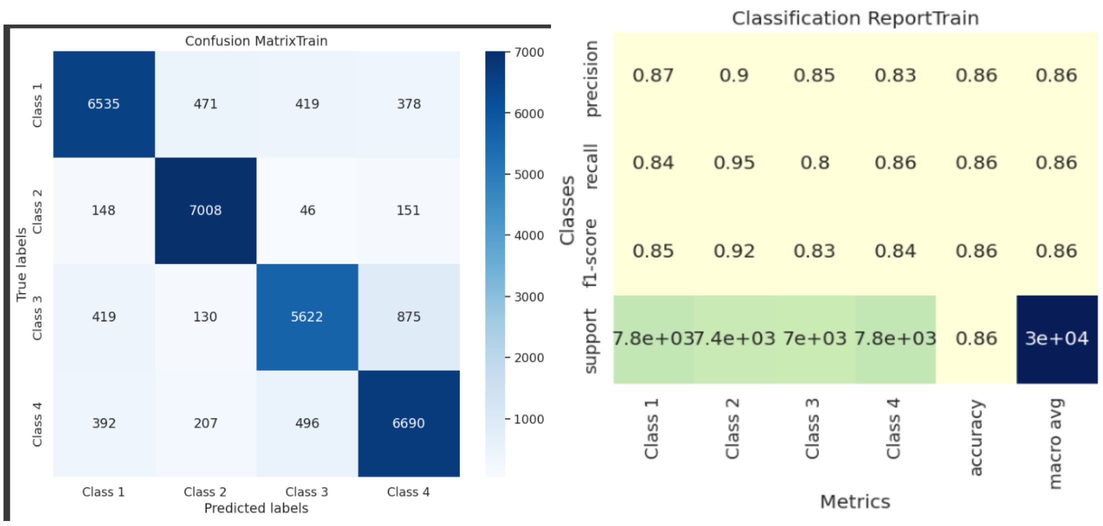
In Downstream I use RNN (LSTM)

Downstream Parameters

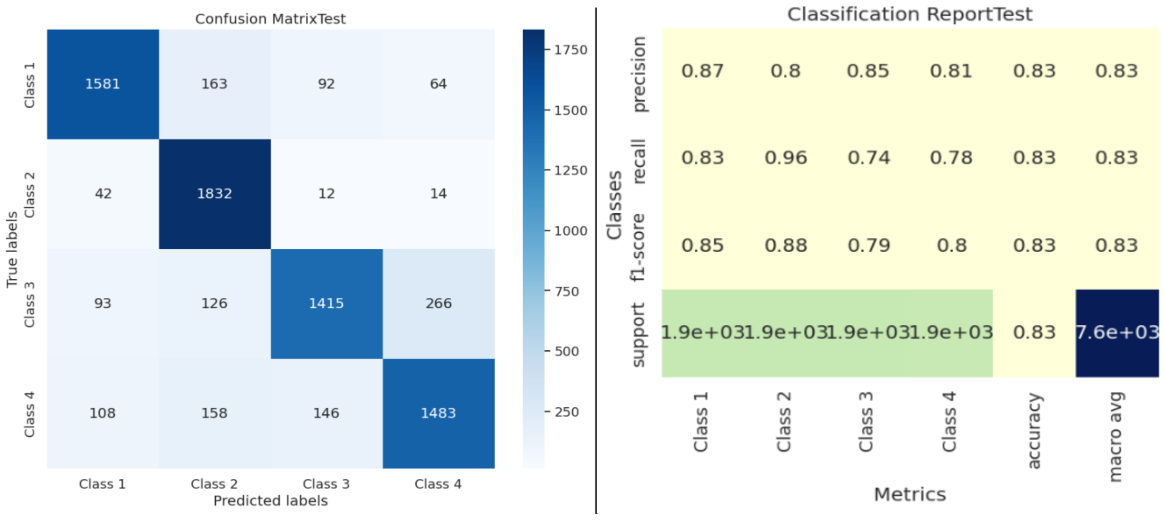


```
input_size = 300
hidden_size = 128
num_layers = 1
num_classes = 4
learning_rate = 0.001
num_epochs = 10
```

confusion matrix and Classification Report of Train data:



confusion matrix and Classification Report of Test data:



Using Skip-Gram word embeddings

Context window size=5

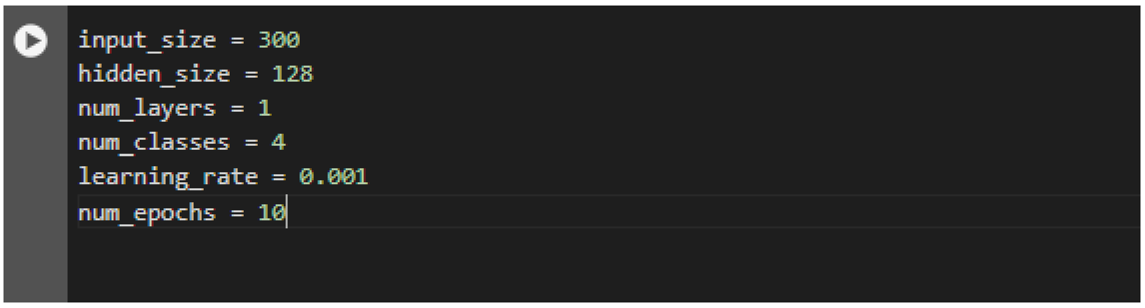
Embeddings Size=300

Test accuracy: 85%

Train Accuracy: 91.25%

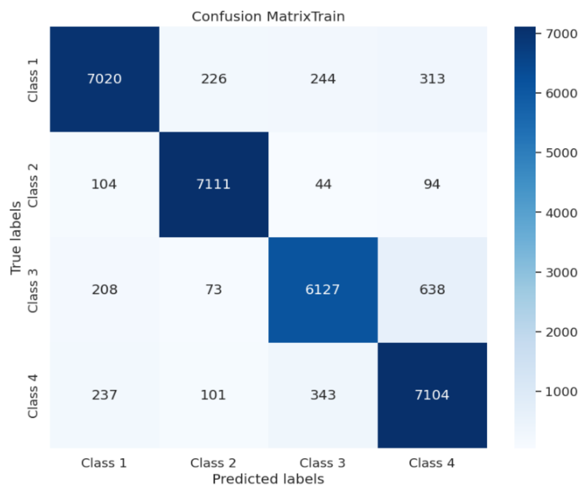
In Downstream I use RNN (LSTM)

Downstream Parameters



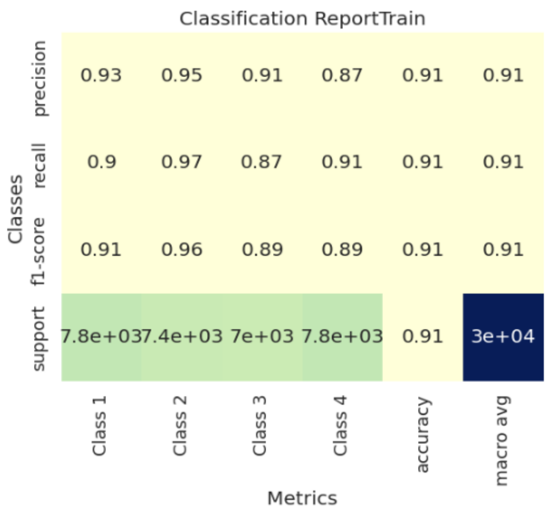
```
input_size = 300
hidden_size = 128
num_layers = 1
num_classes = 4
learning_rate = 0.001
num_epochs = 10
```

confusion matrix and Classification Report of Train data:



Confusion MatrixTrain

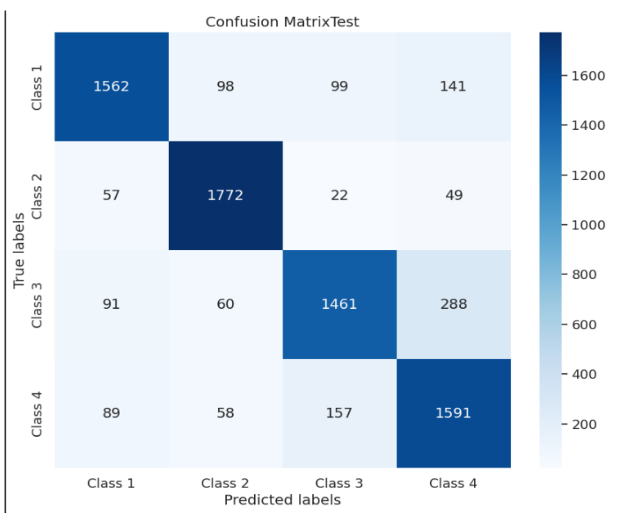
True labels \ Predicted labels	Class 1	Class 2	Class 3	Class 4
Class 1	7020	226	244	313
Class 2	104	7111	44	94
Class 3	208	73	6127	638
Class 4	237	101	343	7104

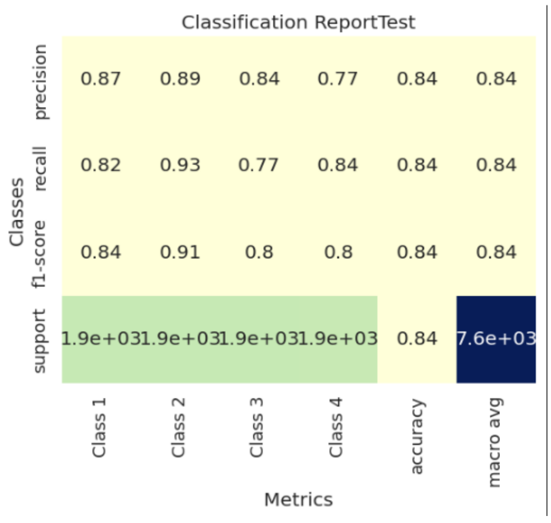


Classification ReportTrain

Classes \ Metrics	Class 1	Class 2	Class 3	Class 4	accuracy	macro avg
precision	0.93	0.95	0.91	0.87	0.91	0.91
recall	0.9	0.97	0.87	0.91	0.91	0.91
f1-score	0.91	0.96	0.89	0.89	0.91	0.91
support	7.8e+03	7.4e+03	7e+03	7.8e+03	0.91	3e+04

confusion matrix and Classification Report of Test data:





Using SVD word embeddings

Context window size=1

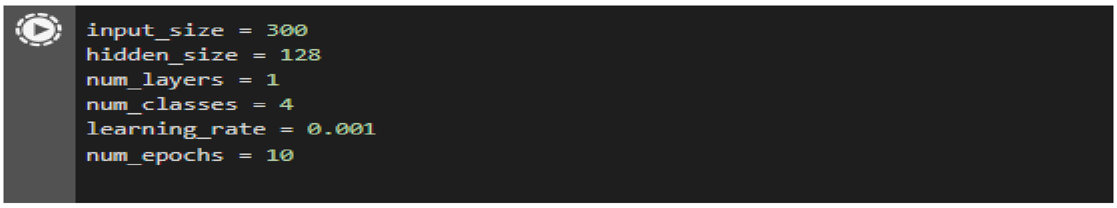
Embeddings Size=300

Test accuracy:81.29%

Train Accuracy:84.6%

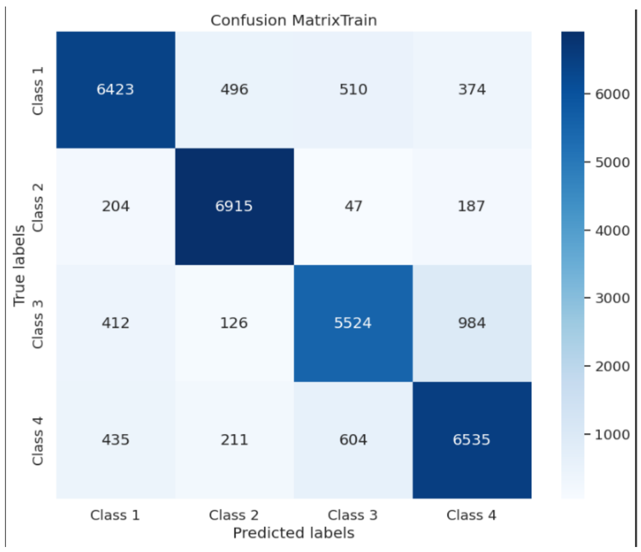
In Downstream I use RNN (LSTM)

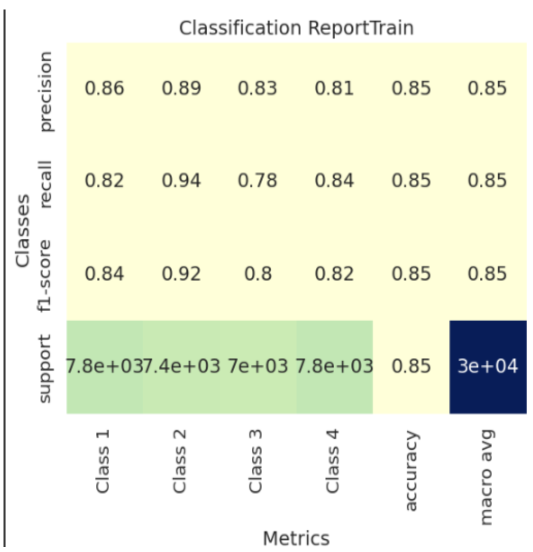
Downstream Parameters



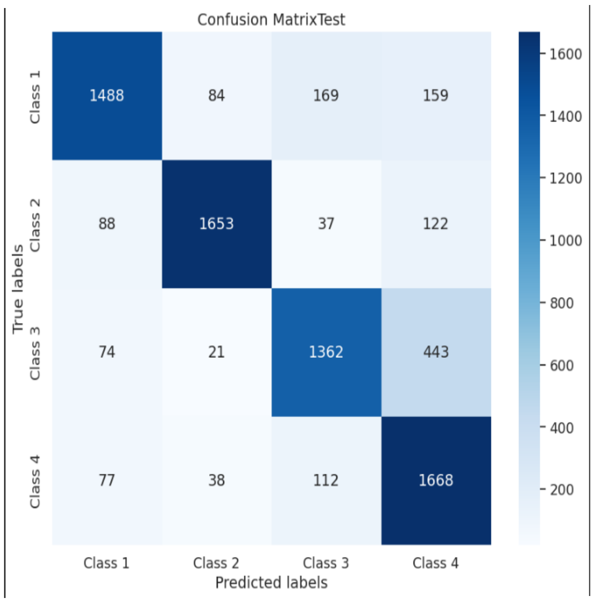
```
input_size = 300
hidden_size = 128
num_layers = 1
num_classes = 4
learning_rate = 0.001
num_epochs = 10
```

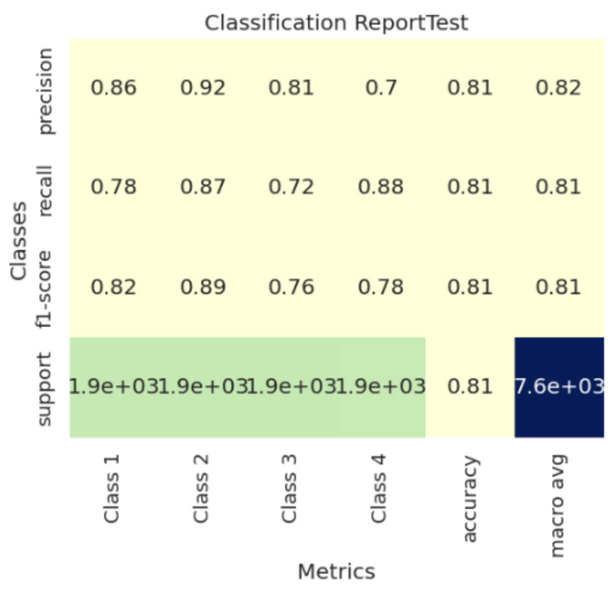
confusion matrix and Classification Report of Train data:





confusion matrix and Classification Report of Test data:





Using Skip-Gram word embeddings

Context window size=1

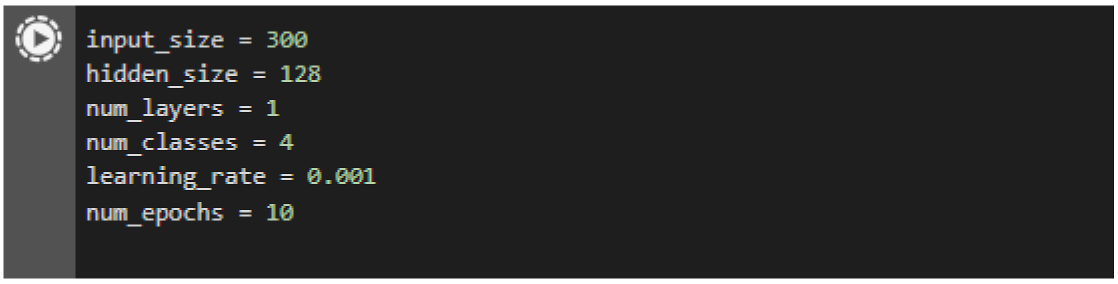
Embeddings Size=300

Test accuracy: 83%

Train Accuracy: 89.7%

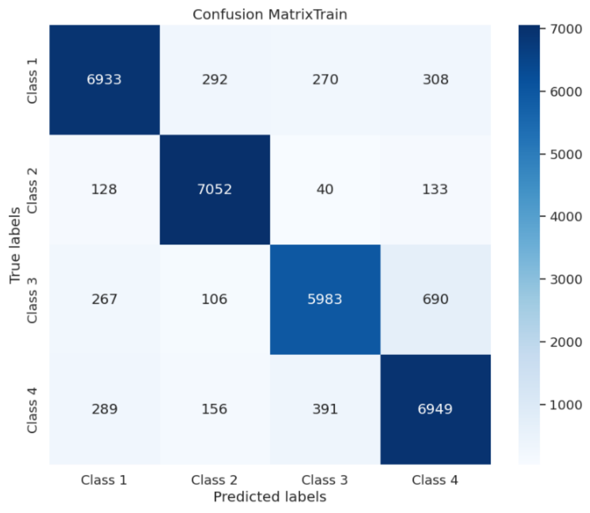
In Downstream I use RNN (LSTM)

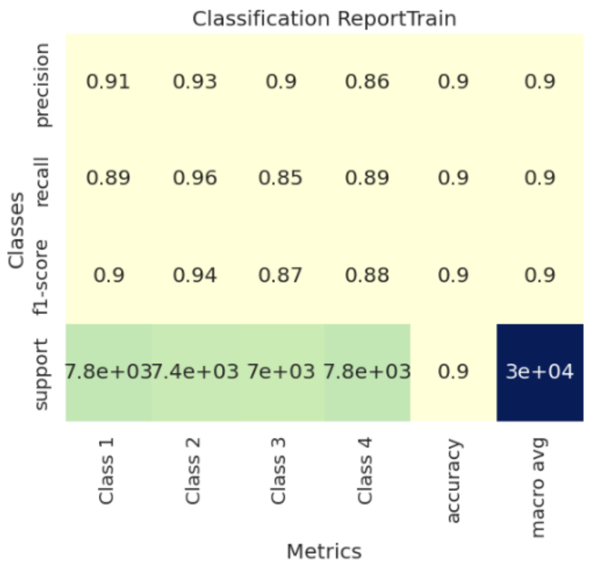
Downstream Parameters



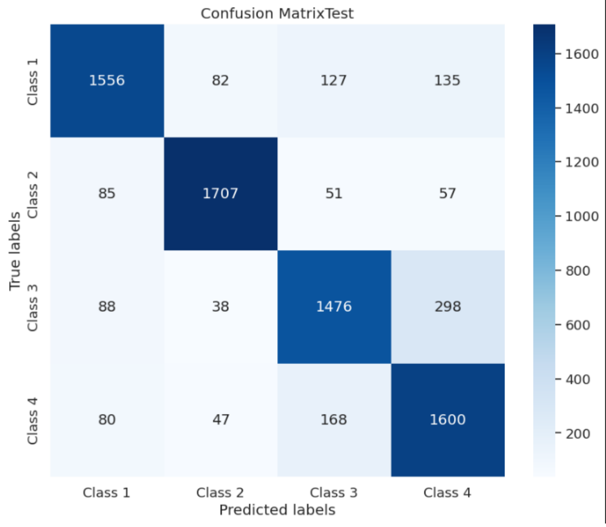
```
input_size = 300
hidden_size = 128
num_layers = 1
num_classes = 4
learning_rate = 0.001
num_epochs = 10
```

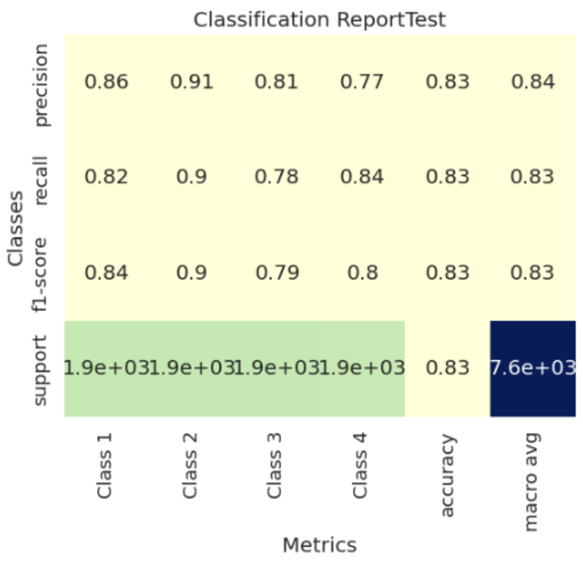
confusion matrix and Classification Report of Train data:





confusion matrix and Classification Report of Test data:

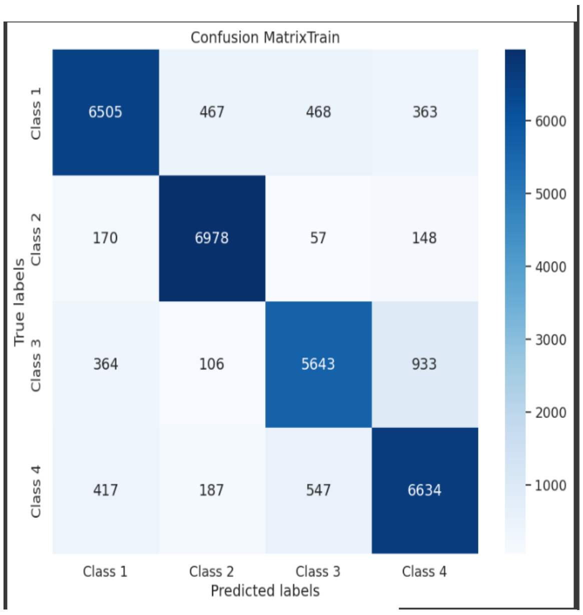


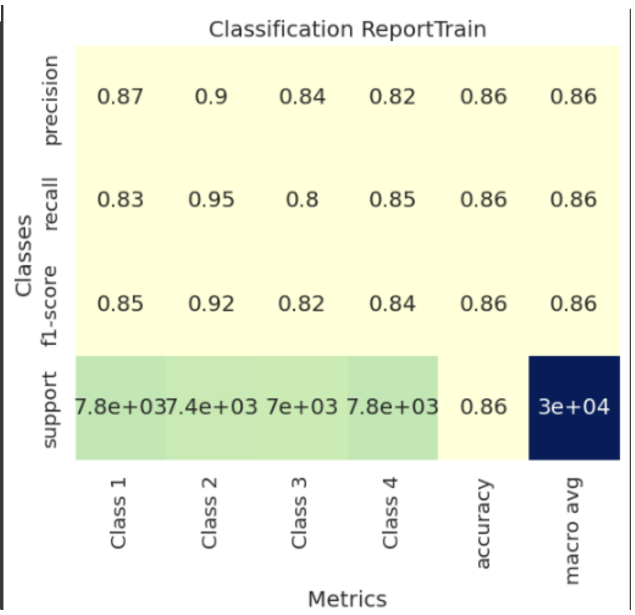


Analysis

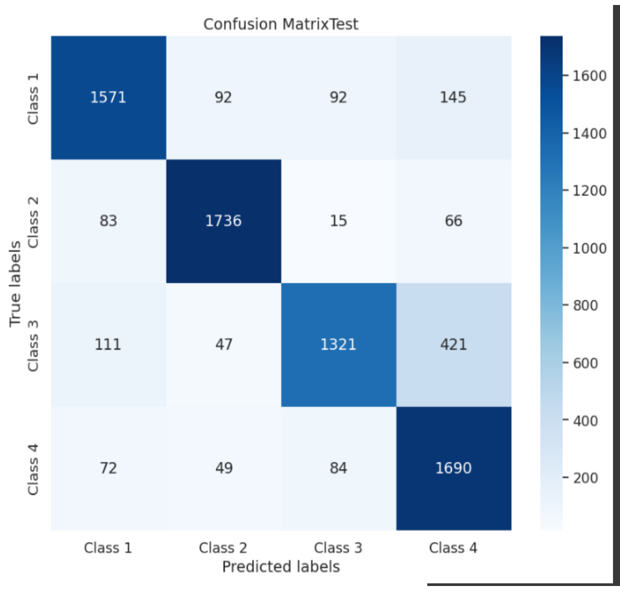
• When we used SVD embedding with a window size of 3 for the classification task, it gave us 83% accuracy on new data and 85% on the training data.

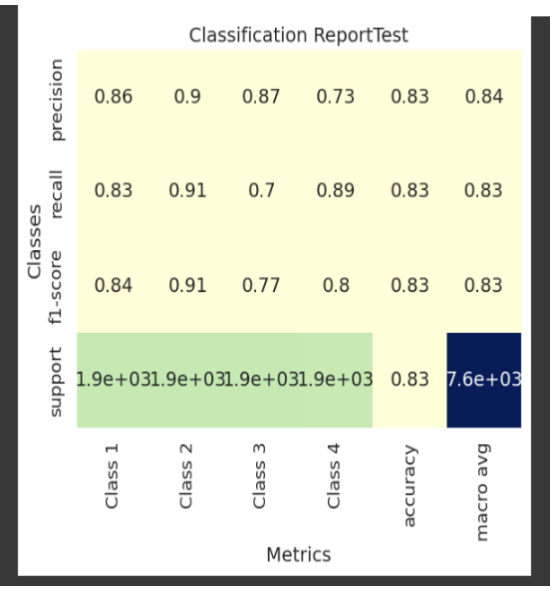
confusion matrix and Classification Report of Train data:





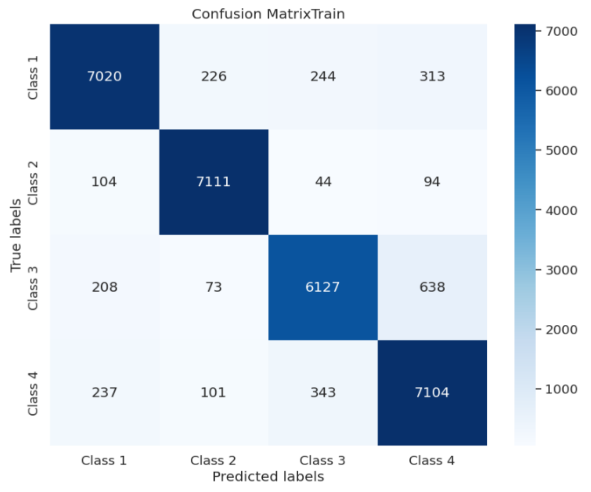
confusion matrix and Classification Report of Test data:

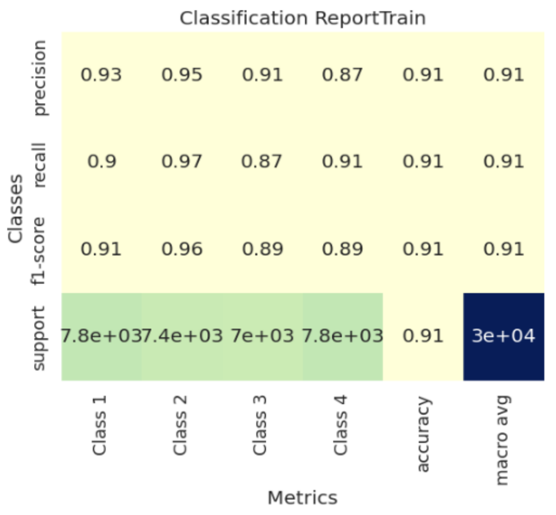




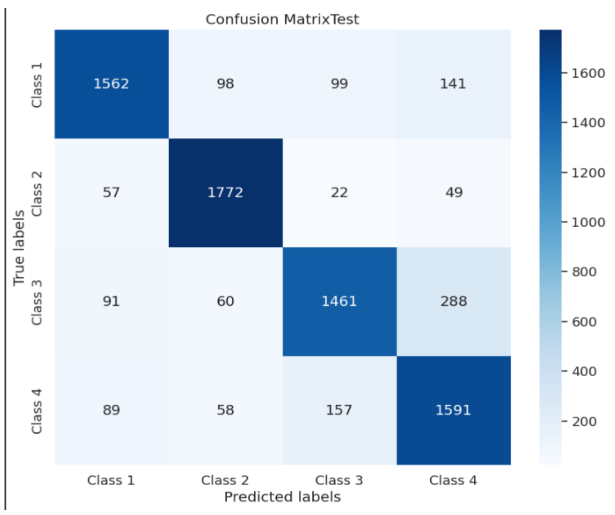
• On the other hand, when we used skip-gram embedding for the same task, but with a window size of 5, it performed even better. It got 85% accuracy on new data and an impressive 91% on the training data.

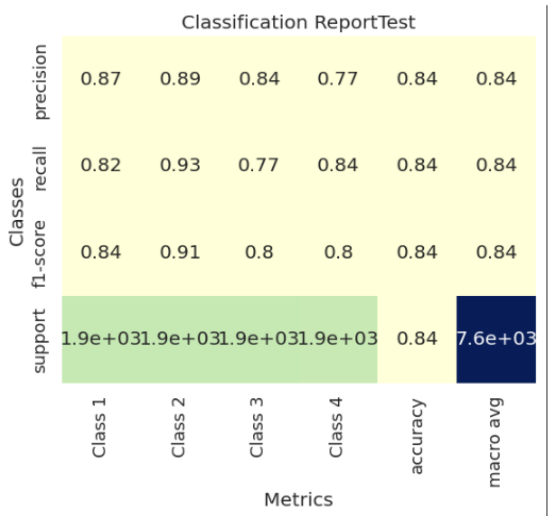
confusion matrix and Classification Report of Train data:





confusion matrix and Classification Report of Test data:





• Skip-gram consistently showed slightly better performance across all window sizes (1, 3, 5), though the difference in accuracy wasn't huge.

• What's interesting is that skip-gram stayed better even when we changed the context window sizes, showing it's good at handling different situations.

• Overall, both methods did well, but skip-gram seemed to do slightly better. One possible reason could be that in this task, I only used 30,000 for creating the embedding.

• Skip-gram might be better than SVD because it captures more detailed contextual information. This could help it make more accurate predictions.

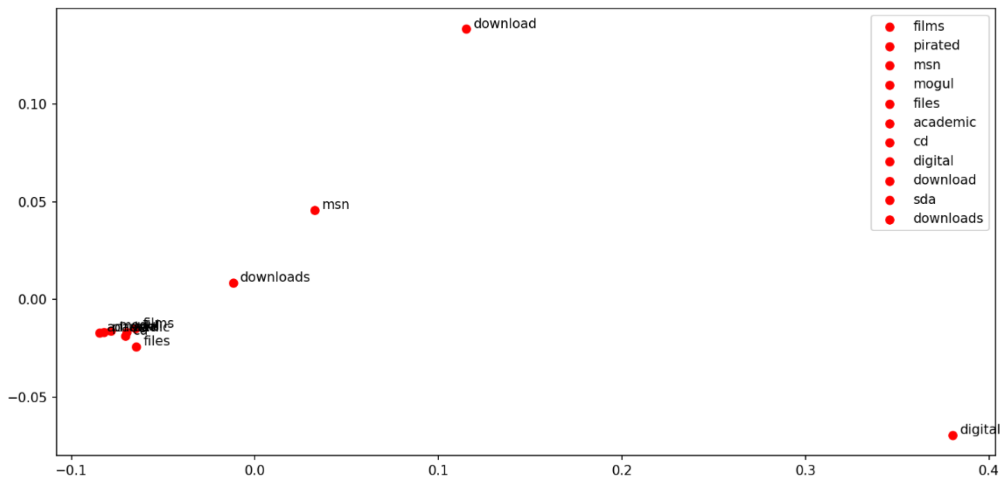
• Both models might perform similarly because they're both good at capturing relationships between words in the text. Even though they use different techniques, they still achieve comparable results in the end.

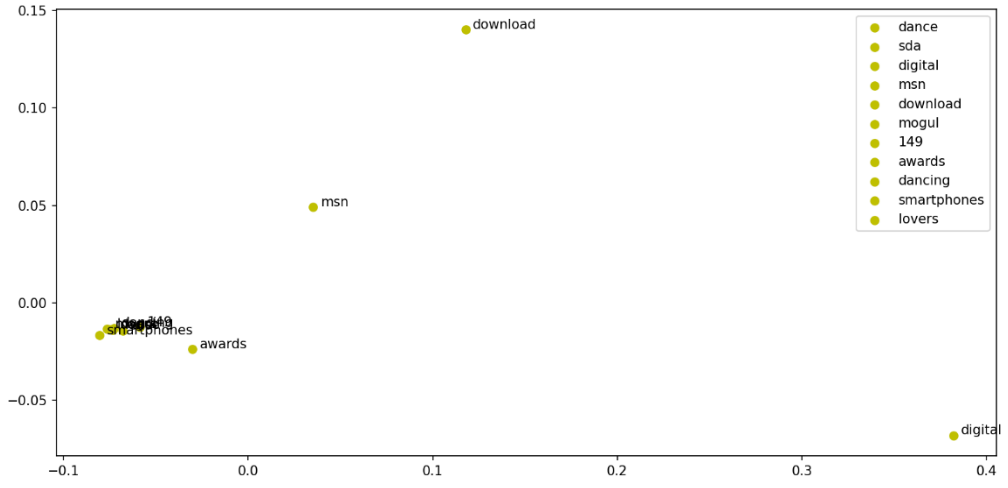
When skip-gram is better than SVD:

Capturing Contextual Similarities: Skip-gram tends to perform better in tasks where capturing contextual similarities between words is crucial. Its ability to predict the surrounding words given a target word allows it to capture more nuanced relationships within the text.

Handling Large Vocabularies: Skip-gram typically handles large vocabularies more efficiently than SVD. Its hierarchical softmax and negative sampling techniques enable faster training, making it more suitable for large-scale datasets. But in our case the vocabularies is not too large so that our model perform similar.

The graph below shows the 10 similar words for the given word "films" and “dance” using the SVD model.


download
msn
downloads
films
files
digital
films
pirated
msn
mogul
files
academic
cd
digital
download
sda
downloads


download
msn
smartphones
awards
digital
dance
sda
digital
msn
download
mogul
awards
dancing
smartphones
lovers