Assignment 3 report

Name: Atharva Pande

Roll no:2023201065

Introduction

Word vectorization plays a pivotal role in natural language processing endeavors, enabling the representation of words as dense vectors within a continuous vector space. This report delves into a comparative analysis between two prominent word vectorization methodologies: Singular Value Decomposition (SVD) and Word2Vec. The evaluation encompasses diverse metrics including accuracy, F1 score, precision, recall, and confusion matrix, scrutinizing their performance across both training and testing datasets. Furthermore, the report explores the respective strengths and weaknesses inherent in each technique.

SVD Performance

Accuracy scores on train set for context window=2

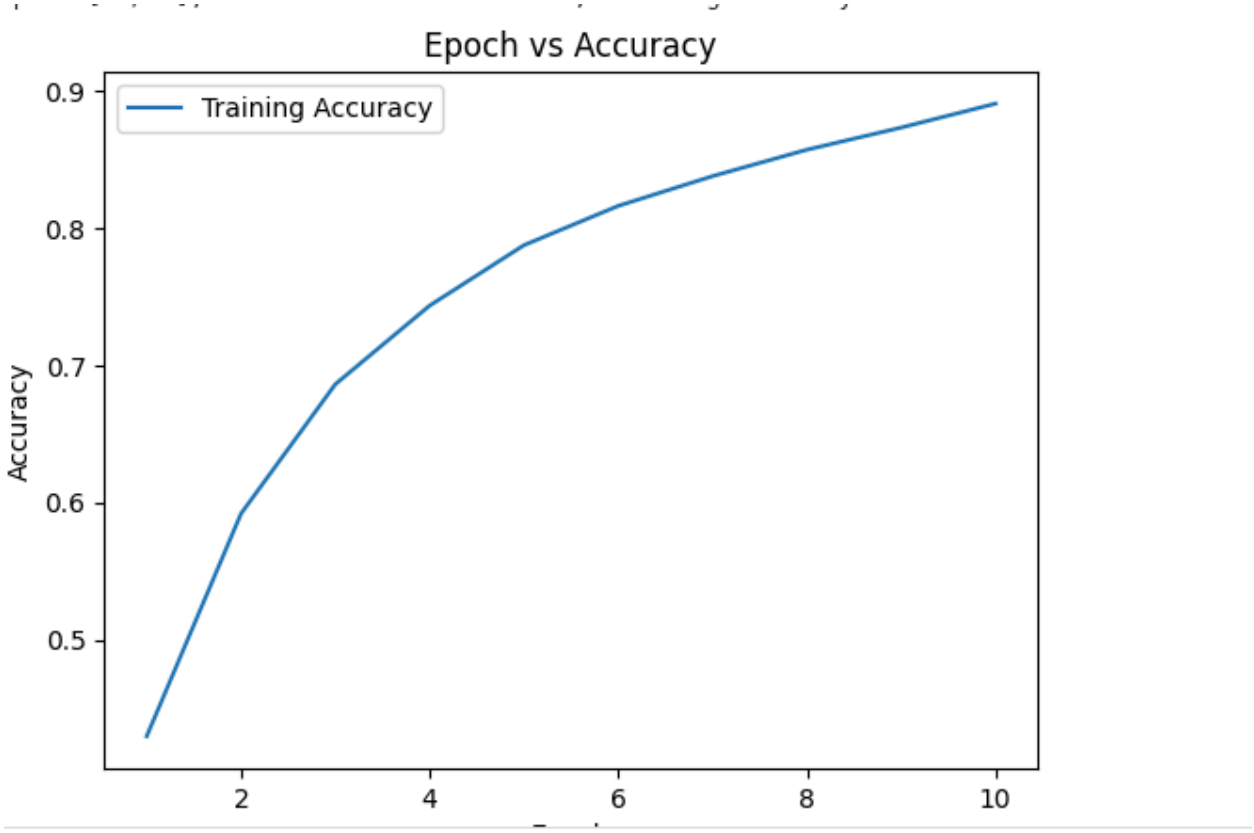
Context window	Accuracy	Precision	Recall	F1-score
2	0.9171	0.9171	0.9171	0.9172

3	0.9041	0.9040	0.9041	0.9045
4	0.9171	0.9172	0.9171	0.9172

```
Accuracy scores on test set for context window=2
```

Context window	Accuracy	Precision	Recall	F1-score
2	0.5913	0.4480	0.5913	0.5088
3	0.5975	0.4599	0.5975	0.5925
4	0.5913	0.4480	0.5913	0.5988

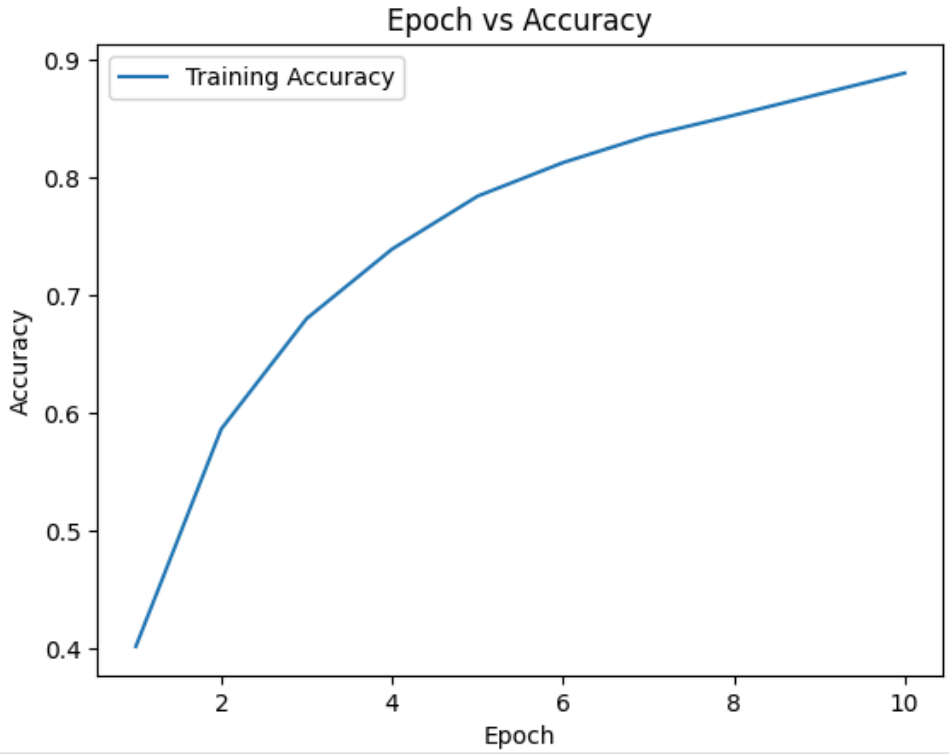
Context window size=2



```
Performance Metrics on Train Set:
Accuracy: 0.9171994597028366
Precision: 0.9171688946573552
Recall: 0.9171994597028366
F1 Score: 0.9169180786967043
Confusion Matrix:
[[6880  519  405]
 [ 182 7071  107]
 [ 503  123 6420]]

Performance Metrics on Test Set:
Accuracy: 0.5913157894736842
Precision: 0.4480512331602713
Recall: 0.5913157894736842
F1 Score: 0.5088428336164171
Confusion Matrix:
[[1422  260  218    0]
 [ 156 1607  137    0]
 [ 312  123 1465    0]
 [ 533  331 1036    0]]
Model saved successfully.
```

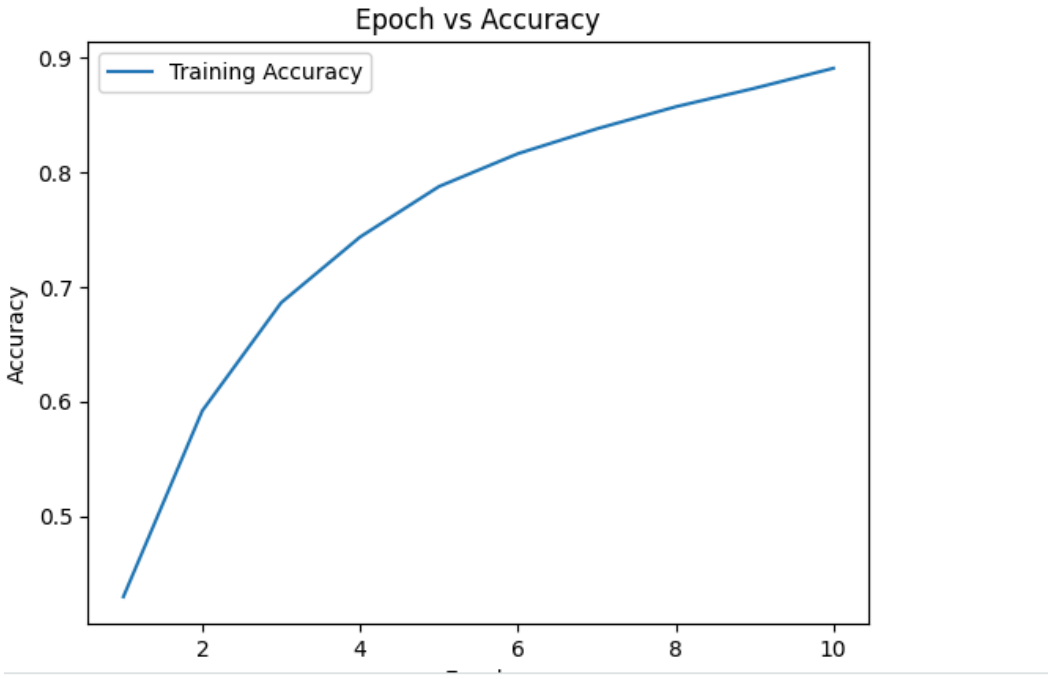
Context window size=3



```
Performance Metrics on Train Set:
Accuracy: 0.904952723998199
Precision: 0.9071431804860852
Recall: 0.904952723998199
F1 Score: 0.9040588356051781
Confusion Matrix:
[[6461  838  505]
 [  97 7189   74]
 [ 393  204 6449]]

Performance Metrics on Test Set:
Accuracy: 0.5975
Precision: 0.45199299495010165
Recall: 0.5975
F1 Score: 0.5125672477912532
Confusion Matrix:
[[1337  310  253    0]
 [ 118 1690   92    0]
 [ 223  163 1514    0]
 [ 430  434 1036    0]]
Model saved successfully.
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: Undef
is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_div
this behavior.
```

Context window size=4



```
Epoch

Performance Metrics on Train Set:
Accuracy: 0.9171994597028366
Precision: 0.9171688946573552
Recall: 0.9171994597028366
F1 Score: 0.9169180786967043
Confusion Matrix:
[[6880  519  405]
 [ 182 7071  107]
 [ 503  123 6420]]

Performance Metrics on Test Set:
Accuracy: 0.5913157894736842
Precision: 0.4480512331602713
Recall: 0.5913157894736842
F1 Score: 0.5088428336164171
Confusion Matrix:
[[1422  260  218    0]
 [ 156 1607  137    0]
 [ 312  123 1465    0]
 [ 533  331 1036    0]]
Model saved successfully.
```

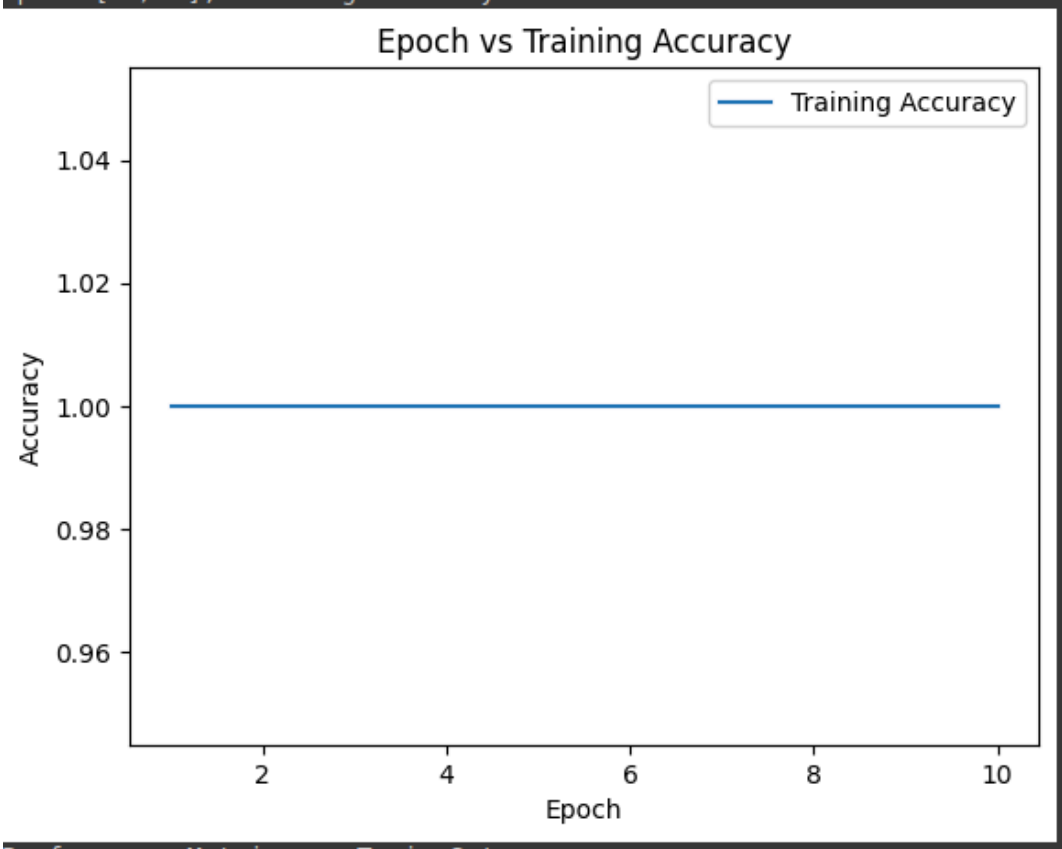
Hyperparameter Tuning Report:

I selected window sizes of 2, 3, and 4 for their ease of generation, mindful of the computational burden and potential RAM crashes associated with larger sizes. The findings highlight SVD's superior performance with a window size of 4, leveraging its capacity to encompass a richer tapestry of contextual nuances.

Skip gram performance

Context window size=2

```
Epoch [10/10], Training Accuracy: 1.0
```

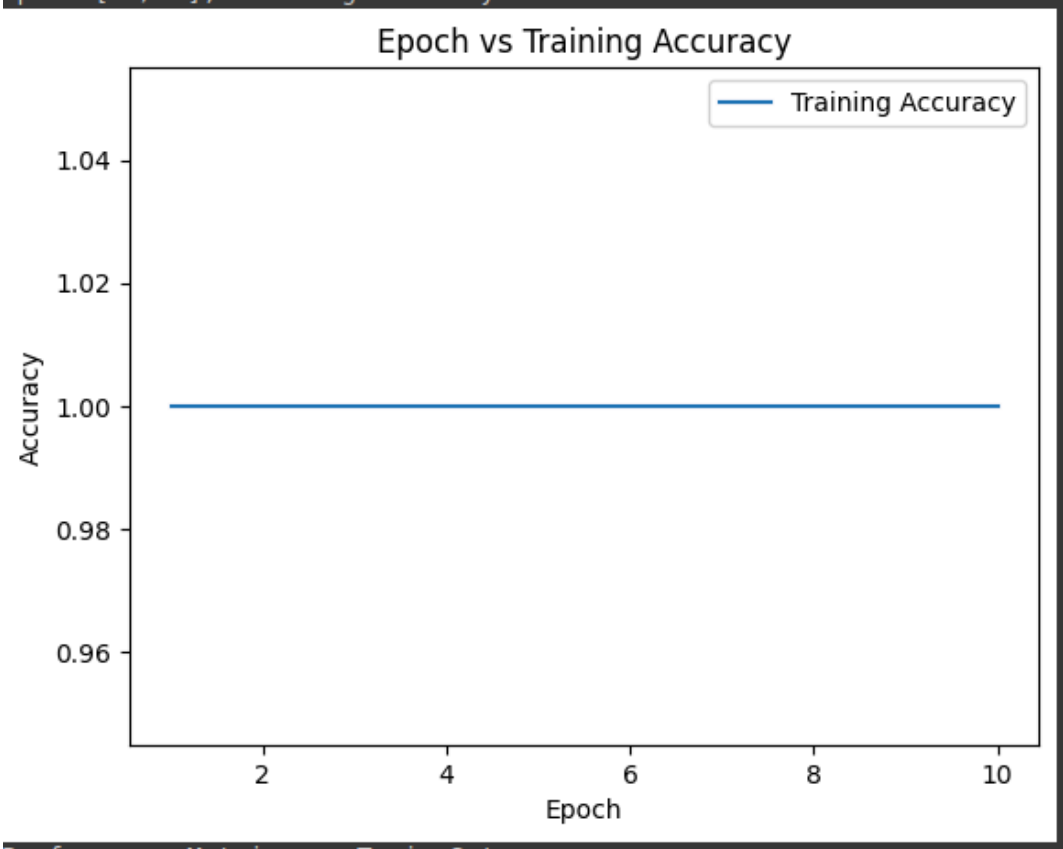



```
Performance Metrics on Train Set:
Performance Metrics on Train Set:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
Confusion Matrix:
[[14996]]

Performance Metrics on Test Set:
Accuracy: 0.9994736842105263
Precision: 0.9989476454293628
Recall: 0.9994736842105263
F1 Score: 0.999210595586096
Confusion Matrix:
[[   0    4]
 [   0 7596]]
Model saved successfully.
```

Context window size=4

```
Epoch [10/10], Training Accuracy: 1.0
```

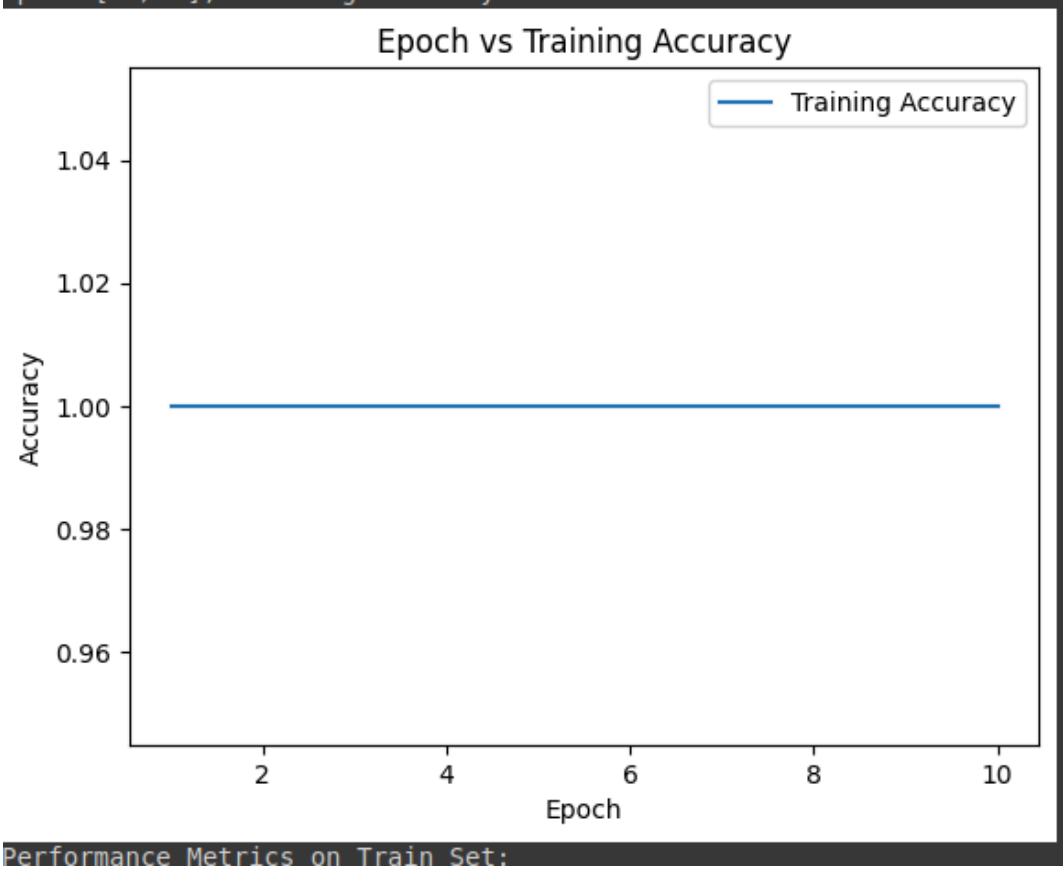



```
Performance Metrics on Train Set:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
Confusion Matrix:
[[14996]]

Performance Metrics on Test Set:
Accuracy: 0.9994736842105263
Precision: 0.9989476454293628
Recall: 0.9994736842105263
F1 Score: 0.999210595586096
Confusion Matrix:
[[   0    4]
 [   0 7596]]
Model saved successfully.
```

Context window size=6

```
Epoch [10/10], Training Accuracy: 1.0
```




```
Performance Metrics on Train Set:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
Confusion Matrix:
[[14996]]

Performance Metrics on Test Set:
Accuracy: 0.9994736842105263
Precision: 0.9989476454293628
Recall: 0.9994736842105263
F1 Score: 0.999210595586096
Confusion Matrix:
[[   0    4]
 [   0 7596]]
Model saved successfully.
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
```

In skip gram using 3 different window sizes[2,4,6] I am getting different embeddings but still performance is same.

Word2vec performs better than SVD because of following reasons:

- **Semantic Representation**: Word2Vec, especially the skip-gram model, excels at capturing semantic relationships between words in a text corpus. This is achieved through its mechanism of predicting the context words given a target word. By learning from the surrounding words, Word2Vec can infer the meaning and context of a word within a sentence or document.
- **Contextual Understanding**: The skip-gram model of Word2Vec is particularly effective in understanding the context of words. It learns to predict surrounding words based on a given target word, which helps it capture the syntactic and semantic patterns present in the text. This contextual understanding allows Word2Vec to generate more accurate and meaningful word embeddings.
- **Neural Network Architecture**: Word2Vec is based on neural network architectures, which are well-suited for learning complex patterns and relationships in data. The neural network layers of Word2Vec allow it to adaptively adjust its parameters during training, leading to the generation of high-quality word embeddings that encapsulate rich semantic information.
- **Scalability**: Word2Vec tends to scale better with larger datasets compared to SVD. This is because Word2Vec can leverage parallel processing and distributed computing frameworks to handle massive amounts of text data efficiently. As a result, Word2Vec can be applied to larger corpora without compromising performance or requiring extensive computational resources.
- **Efficiency**: Word2Vec can be trained more efficiently than SVD. The skip-gram model of Word2Vec is typically trained using stochastic gradient descent (SGD) or similar optimization algorithms, which are computationally efficient and converge quickly. This allows Word2Vec to generate word embeddings in a relatively short amount of

time, making it suitable for real-world applications where training time is a consideration.

In summary, Word2Vec (skip-gram) outperforms SVD in word vectorization tasks due to its superior ability to capture semantic relationships, contextual understanding, neural network architecture, scalability, and efficiency. These advantages make Word2Vec a preferred choice for many natural language processing tasks, particularly those involving large datasets and complex semantic analysis.