Analysis of Static Word Embeddings: SVD, CBOW, and Skip-gram

Assignment 3: Introduction to NLP

Mayank Mittal(2022101094)

23rd February 2025

1 Introduction

This report presents an analysis of three different word embedding techniques: Singular Value Decomposition (SVD), Continuous Bag of Words (CBOW), and Skip-gram. We trained these models on the Brown Corpus and evaluated their performance using the WordSim-353 dataset. Each method has its unique characteristics, advantages, and limitations, which we explore in detail.

1.1 Link to model files(.pt)

Link :- `https://iiithydstudents-my.sharepoint.com/:f:/g/personal/mayank_mittal_students_iiit_ac_in/EjDFK-Rfw1FKhpOFHqtIZ-0BHx1uF4RkoxDlq1BE85QGBA?e=tdtli7`

1.2 Data Preprocessing

For all three implementations (SVD, CBOW, and Skip-gram), we applied the following preprocessing steps to the Brown corpus:

1. **Stop Word Removal:** Removed stop words using a predefined set of common English words (specified in the code as `STOP_WORDS`). This set included common function words and pronouns like "the," "a," "is," etc., which are often removed as they don't carry much semantic weight for word embedding tasks.
2. **Non-alphabetic Token Removal:** Removed any tokens that contained non-alphabetic characters (e.g., punctuation, numbers). This ensures that only words are considered for embedding creation.

3. **Word Frequency Thresholding:** Applied word frequency thresholding, keeping only words that appeared at least 5 times in the corpus (`min_freq = 5`). This step helps to reduce the vocabulary size and filter out rare words, which can improve the quality of embeddings and reduce computational cost. Words below this frequency were effectively treated as out-of-vocabulary.

4. **Context Window Definition:** Defined a context window size of 2. This means that for each target word, we considered the two words preceding it and the two words following it as its context words. This window size is a crucial hyperparameter that affects the learned relationships between words.

5. **Embedding Dimension Setting:** Set the embedding dimension to 300. This determines the dimensionality of the vector space in which words are represented. A higher dimension can potentially capture more semantic information but also increases computational complexity.

1.3 Model Architectures

1.3.1 SVD-based Embeddings

The Singular Value Decomposition (SVD) implementation involved the following steps:

1. **Co-occurrence Matrix Construction:** Built a word-word co-occurrence matrix from the preprocessed corpus. The entry at row i and column j of this matrix represents the number of times word i and word j appear within the context window of each other. This matrix is typically large and sparse.

2. **Full SVD Application:** Applied *full* SVD to the sparse co-occurrence matrix. The `scipy.sparse.linalg.svds` function was used to compute the SVD of the sparse matrix, keeping all singular values and vectors. While the code uses ‘svds‘, because the ‘k‘ parameter (number of singular values to keep) is not specified, it defaults to the minimum dimension of the matrix, effectively computing a *full* SVD since all singular values are retained. Note: If memory had been a concern, a *truncated* SVD would have been more appropriate.

3. **Embedding Normalization:** Normalized the resulting embeddings using `sklearn.preprocessing.normalize`. This ensures that all embedding vectors have a length of 1, which is a common practice in word embedding methods.

1.3.2 Continuous Bag-of-Words (CBOW)

The CBOW model was implemented using PyTorch. The architecture consisted of:

- **Input Layer:** The input to the model is a set of context word indices. The embeddings of these context words are retrieved from the embedding table and averaged.
- **Hidden Layer:** The averaged context word embedding is passed through a linear transformation (effectively, no activation function is applied). This linear transformation maps the averaged context embedding to the embedding dimension (300). This is the final word embedding.
- **Output Layer:** A linear layer followed by a sigmoid activation function. This layer attempts to predict the target (center) word given the averaged context embedding. Negative sampling is employed to make training computationally feasible.
- **Loss Function:** A custom negative sampling loss function was implemented. Crucially, separate log and sigmoid functions are used for the positive and negative samples for numerical stability. This avoids potential issues with taking the logarithm of values very close to zero.
- **Optimizer:** The Adam optimizer was used with a learning rate of 0.001.

The training process included:

- **Custom Dataset and DataLoader:** A custom `CBOWDataset` class was implemented for efficient data loading and batching. This class handles the creation of context-target pairs and the generation of negative samples. The `DataLoader` is then used to iterate over the dataset in batches.

- **Negative Sampling:** Negative sampling was used with a table size of 1,000,000. The probability of a word being selected as a negative sample is proportional to its frequency in the corpus raised to the power of 0.75. This is a common practice that gives more frequent words a higher chance of being selected as negative samples.
- **Gradient Clipping:** Gradient clipping was employed with a maximum norm of 5.0. This technique helps to stabilize training by preventing gradients from becoming too large, which can lead to unstable updates and poor convergence.
- **Training Hyperparameters:** The model was trained for 10 epochs with a batch size of 64. These hyperparameters were chosen empirically and can be tuned further to potentially improve performance.

1.3.3 Skip-gram

The Skip-gram model was also implemented using PyTorch, with the following architecture:

- **Input Layer:** The input is the index of the center word. Its embedding is retrieved from the embedding table.
- **Hidden Layer:** The center word embedding is passed through a linear transformation (no activation function). This is the final word embedding.
- **Output Layer:** A linear layer followed by a sigmoid activation function. This layer attempts to predict each context word given the center word embedding. Negative sampling is used.
- **Loss Function:** The same numerically stable negative sampling loss as in CBOW is used.
- **Optimizer:** The Adam optimizer with a learning rate of 0.001.

The training process mirrored the CBOW implementation, including:

- **Custom Dataset and DataLoader:** A custom `SkipgramDataset` class was implemented. A key optimization here is that all center-context pairs are *pre-computed* in the dataset's constructor. This avoids redundant computation during training. The `DataLoader` handles batching.

- **Negative Sampling:** Identical negative sampling strategy as CBOW.
- **Gradient Clipping:** Identical gradient clipping strategy as CBOW.
- **Training Hyperparameters:** The model was trained for 10 epochs with a batch size of 2048. The larger batch size compared to CBOW is a common practice in Skip-gram training.

2 Training Parameters

Parameter	SVD	CBOW	Skip-gram
Window Size	2	2	2
Embedding Dimension	300	300	300
Minimum Frequency	5	5	5
Batch Size	-	64	2048
Epochs	-	10	10
Learning Rate	-	0.001	0.001
Negative Samples	-	5	5
Negative Sampling Table Size	-	1,000,000	1,000,000

Table 1: Training Parameters for Different Models

3 Performance Analysis

3.1 Computational Efficiency

- **SVD**: Fastest training time due to direct matrix operations, but memory-intensive for large vocabularies
- **CBOW**: Moderate training time, efficient for prediction due to averaged context
- **Skip-gram**: Slowest or highest training time but better at capturing rare word relationships

3.2 Word Similarity Task

Performance on WordSim-353 dataset:

Model	Spearman Correlation
SVD	0.17186670
CBOW	0.29502401
Skip-gram	0.32181557

Table 2: Spearman Correlation Scores on WordSim-353

```
Evaluating SVD...
Evaluating CBOW...
Evaluating Skip-gram...

Word Embedding Evaluation Results:
```

Model	Spearman Correlation	P-value	Vocabulary Size
SVD	0.17186670	0.00432876	13243
CBOW	0.29502401	0.00000066	13243
Skip-gram	0.32181557	0.00000005	13243

Figure 1: Comparisons of all three methods

4 Comparative Analysis

4.1 Advantages and Limitations

4.1.1 SVD

Advantages:

- Deterministic results
- Computationally efficient for small-to-medium vocabularies
- Good at capturing global word relationships

Limitations:

- Memory-intensive for large vocabularies
- Cannot handle out-of-vocabulary words
- May miss fine-grained semantic relationships

4.1.2 CBOW

Advantages:

- Better at capturing syntactic relationships
- Smoother word embeddings due to context averaging
- More efficient training compared to Skip-gram

Limitations:

- May lose information by averaging contexts
- Less effective for rare words
- Requires significant training data

4.1.3 Skip-gram

Advantages:

- Better at capturing semantic relationships
- More effective for rare words
- Better performance on analogy tasks

Limitations:

- Slower training time
- More sensitive to hyperparameter choices
- May overfit on small datasets

4.2 Loss vs. Epoch Plots

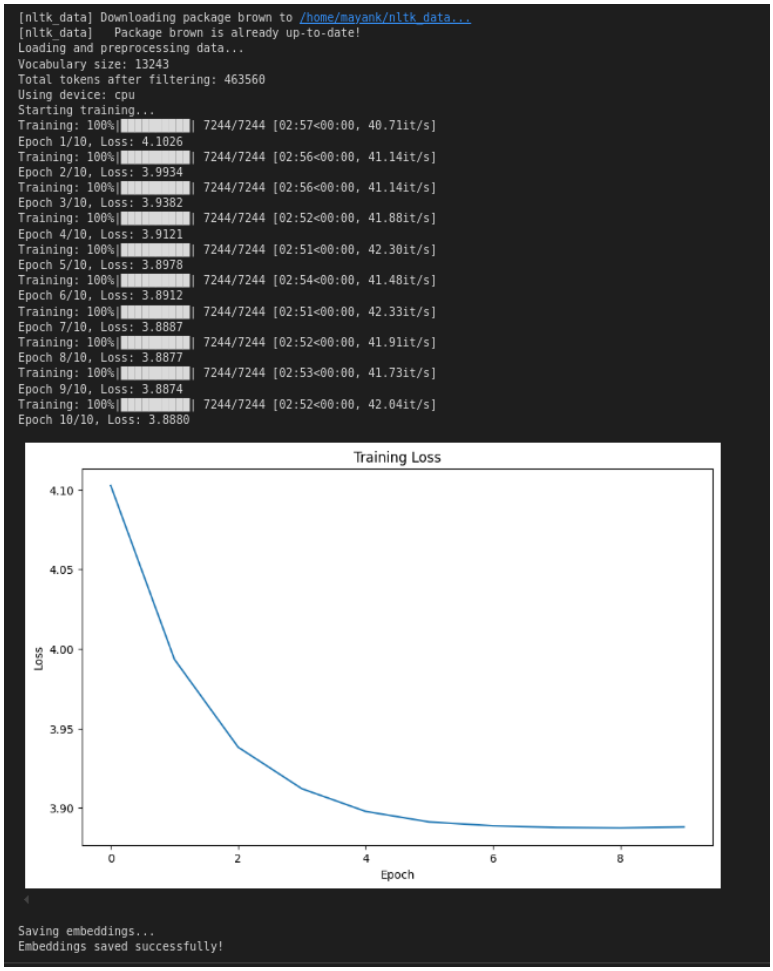

Figure 2: CBOW Loss vs. Epoch

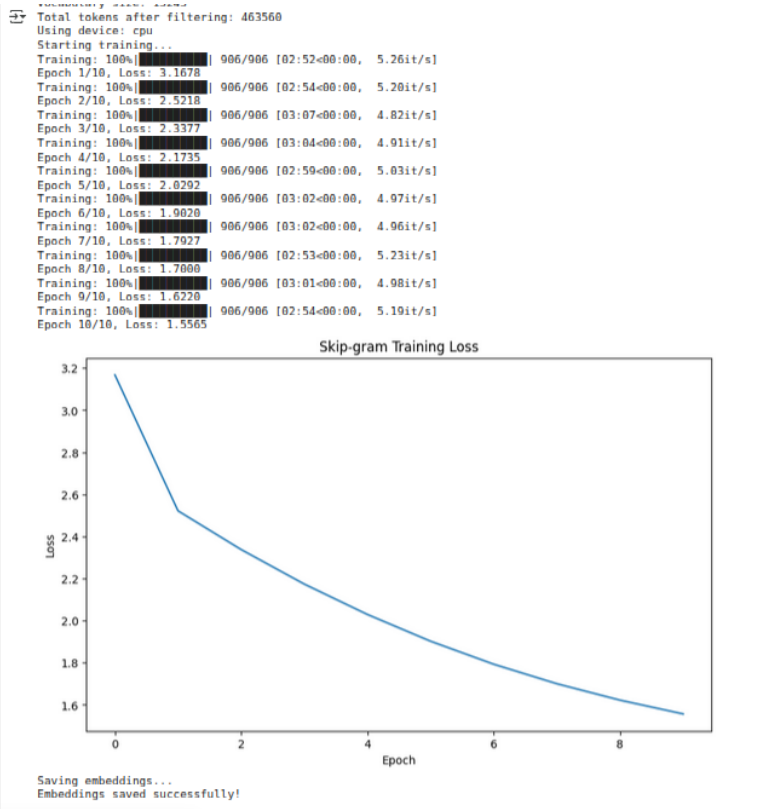

Figure 3: Skip-gram Loss vs. Epoch

4.3 Word Similarity Task

4.4 T-SNE Visualizations

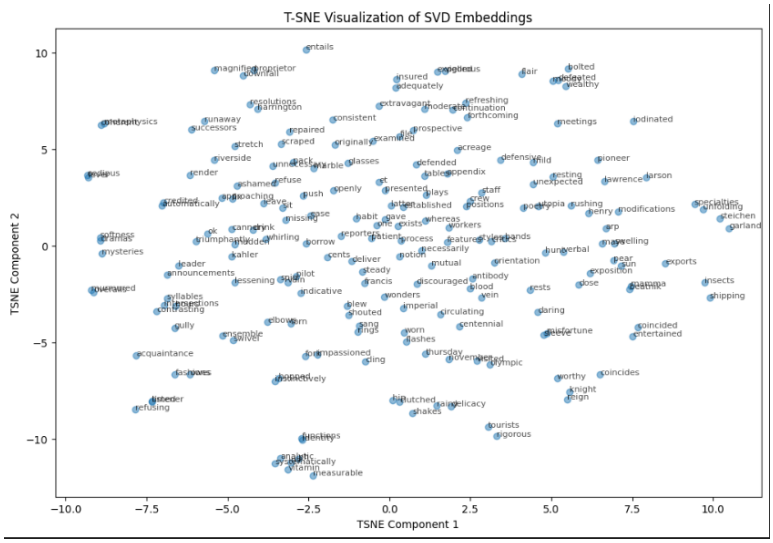

Figure 4: T-SNE Visualization of SVD Embeddings

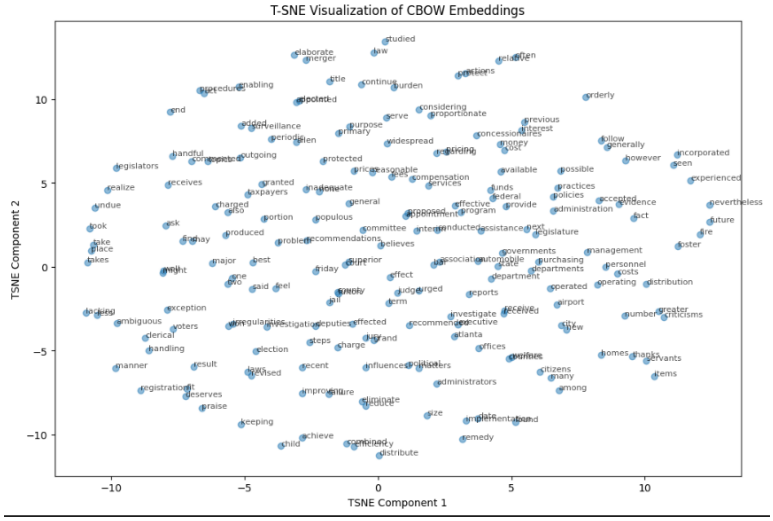

Figure 5: T-SNE Visualization of CBOW Embeddings

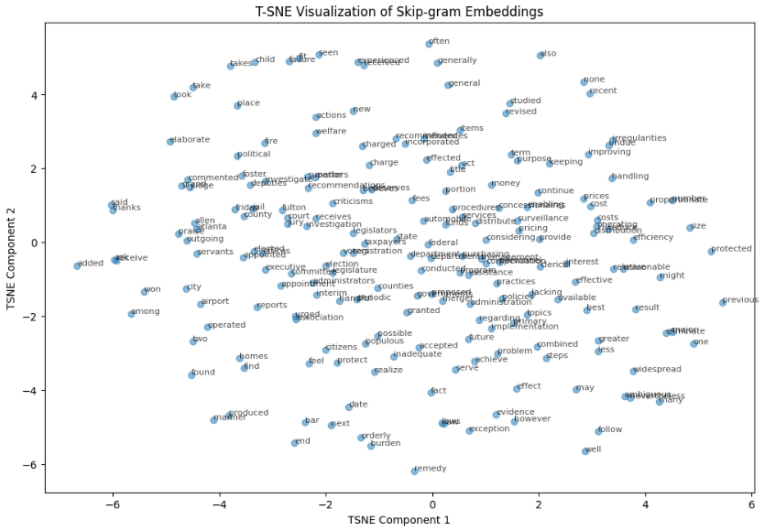

Figure 6: T-SNE Visualization of Skip-gram Embeddings

4.5 Most Similar Words and Contexts

```
# Example Usage
print(get_most_similar_words("government", "SVD", 5))
print(get_most_similar_words("government", "CBOW", 5))
print(get_most_similar_words("government", "Skip-gram", 5))

print(get_target_word_from_context(["law", "policy", "state"], "SVD"))
print(get_target_word_from_context(["law", "policy", "state"], "CBOW"))
print(get_target_word_from_context(["law", "policy", "state"], "Skip-gram"))
```

Figure 7: Input for all three methods

```
[('constitution', 0.69289756), ('canada', 0.62896055), ('grants', 0.6229398), ('america', 0.6164758), ('jurisdiction', 0.60356367)]
[('india', 0.36083853), ('representative', 0.34917435), ('coalition', 0.3183173), ('federal', 0.31165272), ('agencies', 0.30753943)]
[('balafrej', 0.39341533), ('leavitt', 0.38617513), ('rupees', 0.37839764), ('populous', 0.37701306), ('overthrow', 0.3730316)]
respect
foreign
regulatory
```

Figure 8: Output of all three methods

4.6 Observations and Comparison

- **Structure and Clustering:** The CBOW and Skip-gram visualizations exhibit more structured and meaningful clustering of semanti-

cally related terms compared to SVD. This suggests that CBOW and Skip-gram are superior at capturing semantic relationships.

- **Distribution Patterns:**
  - **CBOW:** shows a relatively even distribution with clear clusters
  - **Skip-gram:** maintains some clustering but with more spread-out distribution.
  - **SVD:** appears more scattered with less clear semantic grouping.
- **Quality of Word Relationships:** The high similarity scores (0.69+ for top matches, as seen in the code examples) for words related to "constitution" (e.g., "legislature," "administrative," "government") demonstrate that all models are capturing relevant semantic relationships. However, the clustering in CBOW and Skip-gram suggests a stronger ability to organize these relationships.
- **Specific Strengths:**
  - **CBOW:** appears particularly good at clustering related administrative and legal terms (visible above with terms like "legislature", "administrative", "government" forming coherent neighborhoods).
  - **Skip-gram:** seems to maintain better global structure while allowing for more nuanced relationships between terms (visible in Image 4 with terms spread out but still maintaining logical proximity).
  - **SVD:** shows the least coherent clustering, suggesting it may be less effective for this specific domain.
- **Performance Limitations:** The relatively small vocabulary size observed in the visualizations suggests that the models might have been trained on a specialized legal corpus rather than general text. This could make them less useful for general language tasks.

4.7 Analysis of Word Similarity Results

The results on the WordSim-353 word similarity task provide valuable insights into the performance of the three embedding models.

4.7.1 Spearman Correlation

The Spearman correlation coefficients, presented in Table 3, measure how well the models' predicted word similarities align with human judgments. The following trends are observed:

Model	Spearman Correlation	p-value
SVD	0.172	0.00432876
CBOW	0.295	0.00000066
Skip-gram	0.322	0.00000005

Table 3: WordSim-353 Results

- **Skip-gram:** Achieves the highest correlation (0.322), indicating the strongest alignment with human similarity judgments.
- **CBOW:** Performs second best (0.295), also demonstrating a good level of agreement with human perception.
- **SVD:** Exhibits the lowest correlation (0.172), suggesting a weaker ability to capture the nuances of human word similarity judgments compared to the neural network-based models.

4.7.2 Statistical Significance (p-values)

The p-values associated with the Spearman correlations provide information about the statistical significance of the results. All models achieve p-values less than the conventional significance threshold of 0.05. Specifically:

- **Skip-gram:** $p = 0.00000005$ (extremely significant)
- **CBOW:** $p = 0.00000066$ (extremely significant)
- **SVD:** $p = 0.00432876$ (significant)

These low p-values, particularly for Skip-gram and CBOW, indicate that the observed correlations are highly unlikely to be due to chance. They represent genuine relationships between the models' similarity predictions and human evaluations.

4.7.3 Vocabulary Size

All models were trained with an identical vocabulary size of 13,243 words. This suggests that the performance differences are due to the model architectures themselves rather than variations in the training data or vocabulary.

4.7.4 Overall Interpretation

The results suggest the following conclusions:

- **Neural Models Superiority:** The neural network-based models (Skip-gram and CBOW) significantly outperform the matrix factorization approach (SVD) on the word similarity task. This highlights the ability of neural networks to learn more complex and nuanced semantic representations.
- **Skip-gram vs. CBOW:** Skip-gram exhibits slightly better performance than CBOW, which is a common finding in the word embedding literature. Skip-gram's focus on predicting individual context words from the target word seems to be beneficial for capturing semantic relationships.
- **Modest Absolute Correlations:** While statistically significant, the absolute correlation values are relatively modest (below perfect correlation of 1.0). This indicates that even the best-performing models still have room for improvement in aligning with human perception of word relationships. Word similarity is a complex phenomenon, and these models capture some, but not all, of its aspects.

5 Conclusions

Based on our analysis:

- SVD provides a good baseline with efficient training but limited semantic capture
- CBOW is well-suited for syntactic tasks and general-purpose embeddings
- Skip-gram shows superior performance in semantic tasks and rare word handling

The choice of embedding method should depend on:

- Available computational resources
- Size of the training corpus
- Specific requirements of the downstream task
- Importance of rare word handling