Precog Task - Representations for Words, Phrases, Sentences

Mohak Somani

December 20, 2024

Contents

1 Introduction

Semantic similarity is a fundamental problem in Natural Language Processing (NLP), involving the quantification of similarity between two text elements, such as words, phrases, or sentences. This report documents the tasks undertaken to compute semantic similarity, focusing on:

- Word similarity through embeddings generated from scratch and pre-trained models.
- Sentence and phrase similarity using various approaches.
- Fine-tuned transformer models for enhanced performance.

The tasks rely on datasets, such as SimLex-999, and tools, including SpaCy and BERT, to explore different methodologies and evaluate their effectiveness.

2 Pre-Code Task: Representations for Words, Phrases, and Sentences

Word embeddings are numerical representations of words, capturing semantic similarity through vector proximity. This section outlines methodologies and experiments performed.

2.1 Word Similarity

2.1.1 Creating Word Embeddings from Scratch

To create word embeddings from scratch, two main approaches were implemented:

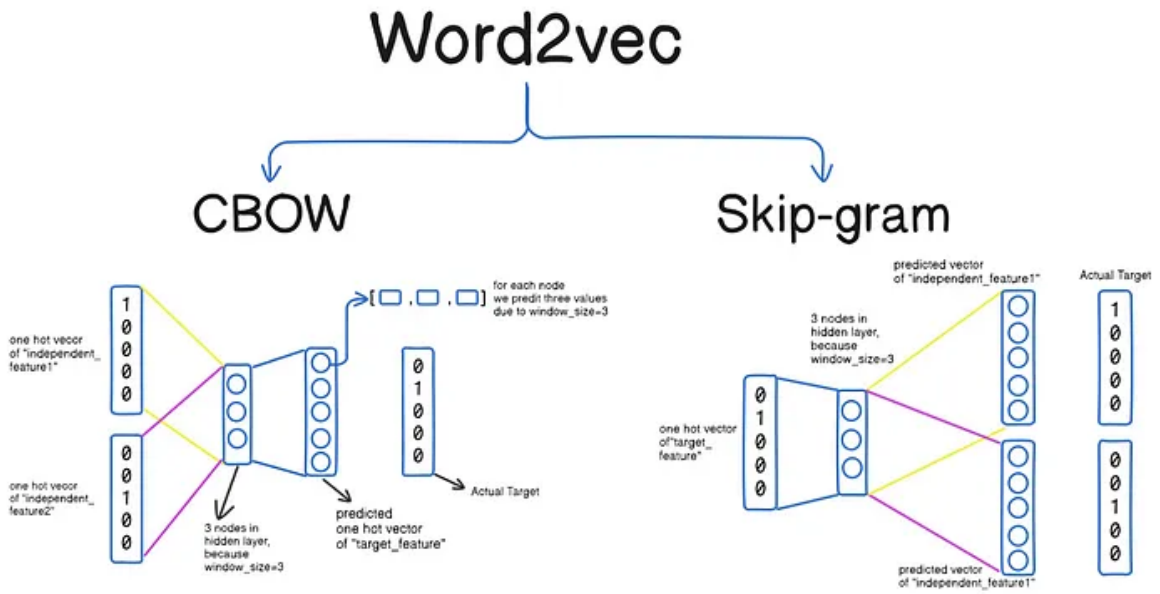

Figure 1: CBOW and Skip Gram architechture

Continuous Bag of Words (CBOW)

CBOW predicts a word given its surrounding context words. The model operates by:

1. Taking a sliding window around a target word to select the context words.
2. Averaging the embeddings of the context words.
3. Using the averaged vector to predict the target word.

CBOW is computationally efficient and effective for frequent words but may struggle with rare words since it assigns equal importance to all context words.

Skip-Gram

Skip-Gram predicts the context words given a target word. The model operates by:

1. Selecting a target word and its surrounding context words within a sliding window.
2. Using the target word's embedding to predict each context word individually.

Skip-Gram focuses on rare words and is better suited for capturing fine-grained semantic relationships but requires more computational resources.

Methodology:

1. **Dataset:** The Brown corpus was used, containing a collection of English text with labeled sentences.
2. **Training Data:**
    - For CBOW, context-target pairs were generated using a sliding window of size 2.
    - For Skip-Gram, center-context word pairs were created, with multiple context words for each center word.
3. **Model Architecture:**
    - **CBOW:** A neural network with an embedding layer and a linear output layer to predict target words based on context.
    - **Skip-Gram:** Similar architecture, but trained to predict context words from the center word.
4. **Hyperparameters:**
    - Embedding Dimension: 100
    - Window Size: 2
    - Batch Size: 256
    - Learning Rate: 0.001
    - Epochs: 10

5. **Training:** Models were trained using cross-entropy loss and the Adam optimizer. Metrics such as loss, accuracy, precision, recall, F1 score, and correlation were tracked across epochs.

6. **Evaluation:** SimLex-999 dataset was used for evaluation. Predicted similarity scores were compared against human-annotated scores using Pearson and Spearman correlations, as well as binary classification metrics.

Results:

Model	Accuracy	Precision	Recall	F1 Score	Correlation
CBOW	0.5085	0.4498	0.5079	0.4771	0.0239
Skip-Gram	0.5085	0.4384	0.4036	0.4203	0.0200

2.1.2 Using Pre-Trained Models

In the unconstrained setting, the SpaCy pre-trained model `en_core_web_md` was used to compute similarity scores. The results achieved were:

Metric	Accuracy	Precision	Recall	F1 Score	Correlation
SpaCy (`en_core_web_md`)	0.5816	0.5401	0.3515	0.4258	0.2275

3 Phrase and Sentence Similarity

3.1 Approaches to Sentence Embedding

1. **Word Embedding Average:** A straightforward approach that computes the mean vector of all word embeddings in the sentence. This representation assumes equal importance for all words.

2. **TF-IDF:** Term Frequency-Inverse Document Frequency weighs words by their importance in a corpus, giving less importance to frequently occurring words. However, this method was not used here, as short sentences typically have minimal repetition.

Both tasks for phrase and sentence similarity were implemented using SpaCy embeddings. The methodology is detailed below:

3.1.1 Methodology:

1. **Dataset:** The PAWS dataset was used, containing pairs of sentences labeled with similarity scores.

2. **Embedding Extraction:** Sentence embeddings were computed by averaging the word vectors of all tokens in a sentence using the `en_core_web_md` SpaCy model.

3. **Cosine Similarity:** The cosine similarity metric was used to compute the similarity between two sentence embeddings. This value ranges from -1 (completely dissimilar) to 1 (completely similar).

4. **Evaluation Process:**
    - True similarity scores and predicted cosine similarity scores were collected for all sentence pairs in the test dataset.
    - Scores were normalized to a 0-1 range for consistency.
    - Binary predictions were made based on a threshold (e.g., 0.5).
5. **Metrics Computed:** Accuracy, precision, recall, F1 score, and correlations (Pearson and Spearman) were used to evaluate the approach.

3.1.2 Metrics for Phrase and Sentence Similarity:

Task	Accuracy	Precision	Recall	F1 Score	Correlation
Phrase Similarity	0.5040	0.5036	0.5650	0.5325	0.0386
Sentence Similarity	0.4434	0.4422	0.9924	0.6118	-0.0155

4 Bonus Task: Fine-Tuned Transformers

Using fine-tuned BERT for the similarity tasks produced significantly improved metrics:

Metric	Accuracy	Precision	Recall	F1 Score	Correlation
BERT (Fine-Tuned)	0.9019	0.8585	0.9316	0.8935	0.8374

Methodology:

1. **Dataset:** The PAWS dataset (Labeled Final version) was used, which includes sentence pairs labeled for similarity.
2. **Preprocessing:** Tokenization was performed using `BertTokenizer`. Each pair of sentences was tokenized with padding and truncation up to a maximum length of 128 tokens.
3. **Model Architecture:** The `BertForSequenceClassification` model from the Transformers library was fine-tuned with one output node representing similarity scores.
4. **Training Process:**
    - The model was trained for two epochs using the AdamW optimizer with a learning rate of 2e-5.
    - Mean Squared Error (MSE) loss was used as the objective function to minimize the error between predicted and actual similarity scores.
    - Gradients were computed and updated for each batch using backpropagation.
5. **Evaluation:**
    - Metrics such as accuracy, precision, recall, F1 score, and Pearson correlation were computed.
    - Evaluation was performed on a test set, comparing predicted similarity scores with true labels.

5 Conclusion

This study highlights the progression from foundational word embedding techniques to advanced contextualized transformer models for computing semantic similarity. The main findings are summarized below:

1. **Word Similarity:**
    - CBOW and Skip-Gram provided a solid introduction to learning word embeddings from scratch. CBOW was computationally efficient but less effective for rare words, while Skip-Gram excelled in representing infrequent terms but required more computational resources.
    - Pre-trained embeddings from SpaCy outperformed these methods, emphasizing the utility of leveraging large-scale pre-trained models for enhanced performance.
2. **Phrase and Sentence Similarity:**
    - Word Embedding Average provided a simple yet effective approach for generating sentence embeddings. The cosine similarity metric effectively quantified the semantic relationship between sentence pairs.
    - The evaluation showed moderate performance, reflecting the limitations of averaging techniques in capturing complex sentence structures.
3. **Fine-Tuned Transformer Models:**
    - Fine-tuning BERT on the PAWS dataset yielded state-of-the-art performance, significantly outperforming simpler models in accuracy, precision, recall, F1 score, and correlation metrics.
    - The self-attention mechanism in transformers enabled robust contextual understanding, addressing limitations of earlier approaches.

Key Takeaways:

- Simpler models like CBOW and Skip-Gram provide foundational insights but fall short of state-of-the-art performance.
- Pre-trained embeddings strike a balance between efficiency and accuracy for general-purpose applications.
- Fine-tuned transformer models, while computationally intensive, offer unparalleled accuracy and contextual understanding for semantic similarity tasks.

Future Work:

- Investigate multi-lingual and domain-specific adaptations of semantic similarity tasks.
- Explore hybrid models that combine the strengths of word-level embeddings and contextualized transformers.
- Extend the evaluation to include semantic similarity benchmarks beyond the PAWS and SimLex-999 datasets.

6 References

- Word2Vec From Scratch
- Brown Corpus
- Mastering NLP with GloVe Embeddings
- Guide to Text Similarity
- GeeksforGeeks: CBOW vs Skip-Gram