Semantic Paraphrase Identification using a Siamese LSTM Network

Development of a deep learning model for Paraphrase Identification, a core task in Natural Language Processing (NLP). The objective was to build a system capable of determining if two sentences have the same semantic meaning. A Siamese Long Short-Term Memory (LSTM) network, enhanced with pre-trained GloVe word embeddings, was implemented. The model was trained and evaluated on the Microsoft Research Paraphrase Corpus (MRPC). The final architecture successfully learns to map sentences into a meaningful vector space where semantic similarity can be measured, achieving a robust performance on the unseen test set and demonstrating the effectiveness of deep learning for understanding nuanced semantic relationships in text.

Github Link:
https://github.com/Rupesh4604/Paraphrase_Identification_on_MRPC_Dataset

Introduction

Paraphrase identification is the task of recognizing whether two sentences, though structured differently, convey the same meaning. This capability is crucial for a wide range of NLP applications, including plagiarism detection, search engine optimization, text summarization, and building more intelligent conversational agents.

Traditional methods often rely on lexical overlap or handcrafted features (e.g., TF-IDF), which can fail to capture the deeper semantic context of a sentence. For instance, "The company laid off 500 workers" and "The firm let go of 500 employees" are paraphrases but share few identical words.

This project moves beyond these limitations by implementing a Siamese LSTM network. This deep learning architecture is specifically designed to learn a "similarity function" between two inputs, making it an ideal choice for this task. The goal is to create a model that understands the meaning of sentences, not just the words they contain.

Dataset

The project utilizes the **Microsoft Research Paraphrase Corpus (MRPC)**, a standard benchmark dataset for this task. It consists of sentence pairs collected from online news sources, each labeled by human annotators as either being a paraphrase (semantically equivalent) or not.

- **Training Set:** Contains 4,076 sentence pairs.
- **Test Set:** Contains 1,725 sentence pairs.

Data Preprocessing and Preparation

1. **Tokenization:** Sentences were converted into sequences of individual words (tokens).

2. **Integer Encoding:** A Tokenizer was fit exclusively on the training data to build a vocabulary, mapping each unique word to a unique integer. This prevents data leakage from the test set.
3. **Padding:** All integer sequences were padded to a uniform length (MAX_SEQUENCE_LENGTH = 35) to enable batch processing by the neural network.

Word Embeddings: GloVe

To inject semantic knowledge into the model from the start, pre-trained **GloVe (Global Vectors for Word Representation)** embeddings were used.

- The glove.6B.100d.txt file, containing 100-dimensional vectors for 400,000 words, was used.
- An embedding matrix was created to map the integers from our vocabulary to their corresponding 100-dimensional GloVe vectors. This matrix was used to initialize the weights of the model's Embedding layer.
- The Embedding layer was set to be non-trainable to preserve the rich semantic information learned during GloVe's pre-training.

Model Architecture: Siamese LSTM Network

A Siamese network architecture was chosen for its effectiveness in learning similarity.

- **Shared Layers:** The core of the network consists of an Embedding layer and an LSTM layer that are **shared** by both input towers. This weight sharing forces the network to learn a single, universal function for processing sentences.
- **Input Towers:** The model takes two inputs, sentence1 and sentence2. Each sentence is passed through the shared Embedding and LSTM layers in parallel. The LSTM layer, with 64 units, processes the sequence of word vectors and outputs a single 64-dimensional vector that represents the contextual meaning of the entire sentence.
- **Distance Metric:** The two output vectors from the LSTM towers are then compared using the **Manhattan distance**. This metric calculates the similarity between the sentence vectors. The distance is transformed with an exponential function to produce a final similarity score between 0 and 1.

Training Process

- **Loss Function:** binary_crossentropy was used, as this is a binary classification problem (paraphrase or not).

- **Optimizer:** The Adam optimizer was chosen for its efficiency and effectiveness.
- **Early Stopping:** To prevent overfitting and find the optimal number of training epochs, an EarlyStopping callback was implemented. It monitored the validation accuracy and stopped the training process if no improvement was seen for 5 consecutive epochs, restoring the model weights from the best-performing epoch.
- **Epochs:** The model was set to train for a maximum of 250 epochs, though early stopping typically concluded the training much sooner.

Results and Evaluation

The model was trained on 90% of the MRPC training set, with 10% used for validation. The final, definitive performance was measured on the unseen MRPC test set.

The key performance metrics on the test set were:

- **Accuracy:** 91.42
- Classification Report:

```
        precision   recall  f1-score   support


       0     0.92     0.82     0.86      578

       1     0.91     0.96     0.94     1146

    accuracy                  0.91     1724
```

- macro avg 0.92 0.89 0.90 1724
- weighted avg 0.91 0.91 0.91 1724
- **Confusion Matrix:**

```
[[472  106]

 [  42 1104]]
```

These results demonstrate the model's ability to generalize its learned understanding of semantic similarity to new, previously unseen data.

Conclusion and Future Work

This project successfully implemented a Siamese LSTM network for paraphrase identification. By leveraging pre-trained GloVe embeddings and a deep learning architecture designed for comparison, the model effectively captures the semantic meaning of sentences, outperforming simpler, feature-based methods. The use of Early Stopping ensured an efficient training process that mitigated overfitting.

While the model performs well, there are several avenues for future improvement:

1. **Hyperparameter Tuning:** Systematically tuning parameters like LSTM units, batch size, and optimizer learning rate could yield further performance gains.
2. **Bidirectional LSTMs:** Replacing the LSTM layer with a Bidirectional LSTM (BiLSTM) would allow the model to process sentences in both forward and backward directions, potentially capturing context more effectively.
3. **Attention Mechanisms:** Incorporating an attention mechanism could help the model focus on the most relevant words in each sentence when determining similarity.
4. **Transformer Models:** For state-of-the-art performance, the next step would be to fine-tune a pre-trained Transformer model like **BERT** or **RoBERTa**, which have shown superior performance on a wide range of NLP tasks, including paraphrase identification.