Text Mining and Natural Language Processing Project

24/06/2024

[Matteo Caviglia, 513026]
[Sofia Gaia Formenti, 514495]

[Project: Paraphrase Recognition (PR)]

Introduction

The primary task of this project is to tackle the problem of paraphrase identification using the PAWS (Paraphrase Adversaries from Word Scrambling) dataset.
Paraphrase identification involves determining whether two sentences express the same meaning despite being phrased differently. This task is crucial in various NLP applications such as text summarization, information retrieval, and question answering systems.
We took as inspiration the Research paper "PAWS: Paraphrase Adversaries from Word Scrambling" by Yuan Zhang, Jason Baldridge and Luheng He and we attempted to replicate most of its results. Here we provide the link to the mentioned paper: PAWS.

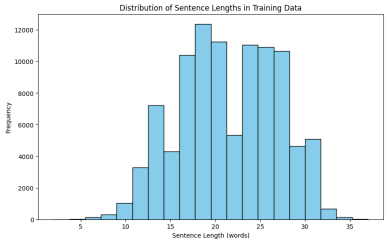
To address this problem, we explore multiple methodologies, including traditional and advanced machine learning models, to compare their effectiveness in accurately identifying paraphrases.

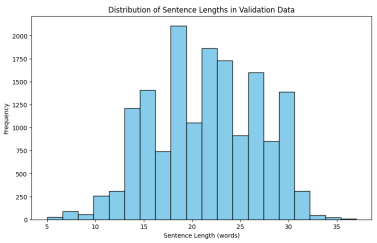
Data

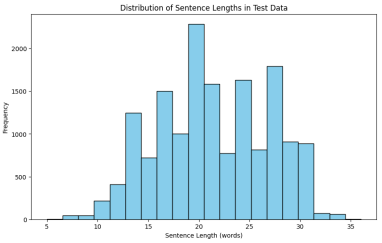
The dataset used in this project is the PAWS dataset, which is known for its challenging and adversarial sentence pairs. The PAWS dataset consists of sentence pairs where the words have been scrambled to create hard negatives, making the paraphrase identification task more complex and realistic.
The **training set** contains a substantial number of sentence pairs, which are used to train the models. This set includes both positive examples (true paraphrases) and negative examples (non-paraphrases). The **development set**, or validation set, is used to tune model parameters and select the best model based on performance. This set is used for some models, but not for others. It helps in preventing overfitting and ensures the model generalizes well to unseen data. The **testing set** is used to evaluate the final performance of the trained models. This set is kept separate from the training and development sets to provide an unbiased evaluation of the model's effectiveness.
Note: Sometimes we only take some fractions of the training data, in order to lower the waiting time.

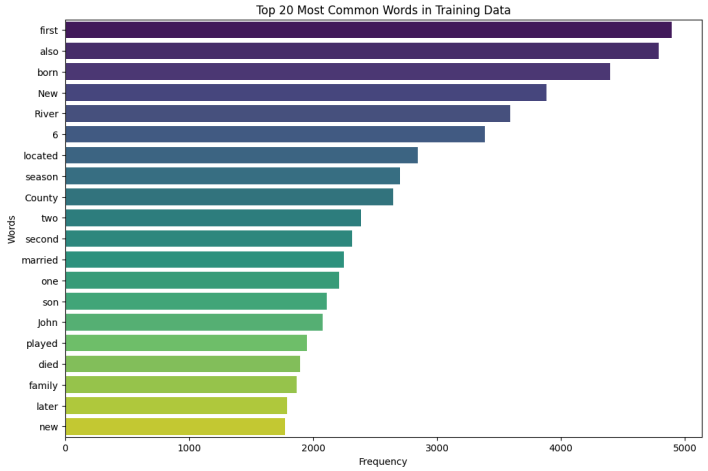
Dataset Statistics:
Number of Sentences:
Training set: 49,401 sentence pairs
Development set: 8,000 sentence pairs
Testing set: 8,000 sentence pairs

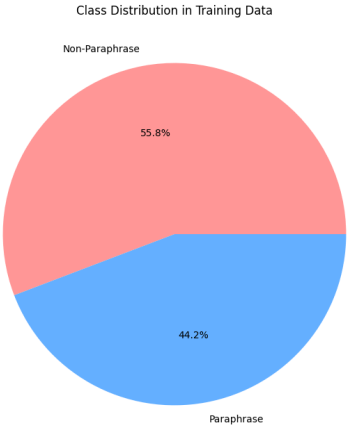
Average Sentence Length: Sentences in the dataset vary in length, typically ranging from 5 to 15 words.



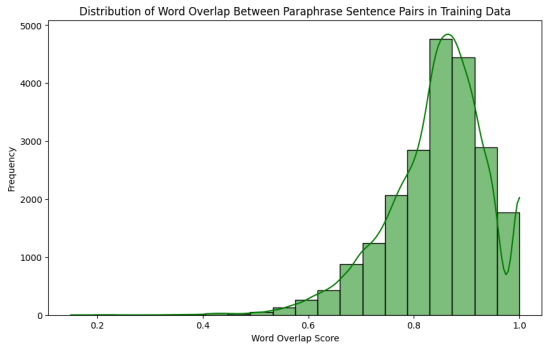


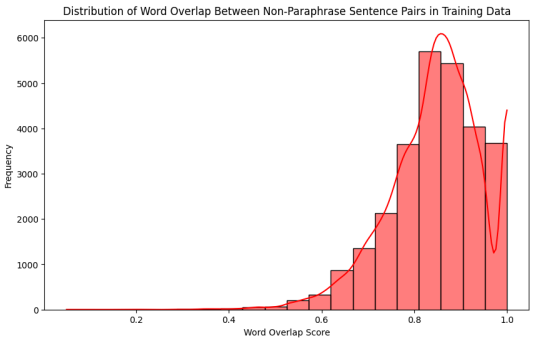




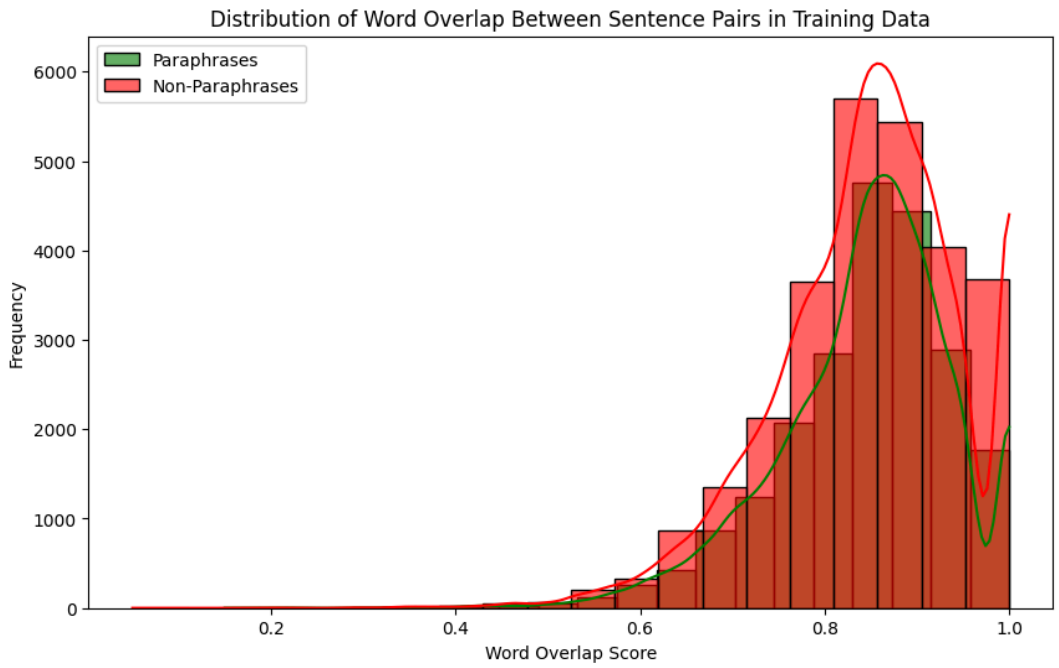


The following graphs underline the difficulty of this classification task, that is the fact that the two phrases to compare have high word overlap. As we can see in both cases (paraphrase and not paraphrase) the overlap is high, this is shown in the first graph where the phrases are paraphrases and in the second one where they are not.





And finally the following graph shows the comparison between these two situations.



The code with the above mentioned graphs can be consulted at the following link: code

Methodology

In this section, we provide a description of the models employed in our project.

Bag of Words (BoW) Model

The first model we implemented was the Bag of Words (BoW) model. After loading the dataset and converting it into Pandas DataFrames we combined all sentences to build a comprehensive vocabulary using the CountVectorizer function, which transformed the text into numerical vectors while removing stopwords. Each sentence pair was then represented by adding the BoW vectors.The data was then split into training and testing sets and a logistic regression classifier was trained. The model was then evaluated on the test set, achieving an accuracy score of ~0.523.
This method provided a straightforward baseline for our paraphrase identification task; its low score didn't discourage us, as it was expected from the beginning.

Embeddings Layer and Double Input model

The embeddings layer and double input model starts with the preprocessing and tokenization of the data (lowercasing, elimination of punctuation and of stopwords), the creation of a vocabulary list with the most frequent words and the unknown (<unk>) token. The sentence tokens are then mapped to their index in the vocabulary and padded.
The next step was to build a neural network model in Keras, which involved creating two input layers for the two sentences, converting word indices to dense vectors using embedding layers, which are then flattened into 1D vectors and concatenated into one merged vector. This merged vector was passed through a dense layer with 64 units and ReLU activation, followed by a final dense output layer with sigmoid activation for binary classification. The model was compiled with the Adam optimizer, binary cross-entropy loss, and accuracy as a metric and gave an accuracy of ~0.601.

RoBERTa with Siamese Network

The next model paired RoBERTa (a robustly optimized BERT approach) with a Siamese network architecture to measure sentence similarity. The main idea was to pass every sentence in a pair through RoBERTa to generate embeddings, compute the similarity between these embeddings with the Siamese network architecture and predict whether the sentences are paraphrases with a classifier with the similarity score. This design was inspired by a public colab project, here is the link: Colab Notebook

The first step was to prepare the data through a process that ensured the data was in the correct format for input into the machine learning model. This was done by setting a size for the training and validation batch and a number of training and validation batches, mapping the dataset to extract sentences and labels, batching the dataset, limiting the number of batches and prefetching data to optimize performance. Features and labels were then extracted from the prepared TensorFlow datasets for training and validation, converted into NumPy arrays, and reshaped.
The next step was to import the necessary tools for RoBERTa from presets, specifically the RobertaPreprocessor and RobertaBackbone (both set to "roberta_base_en"). A normal encoder model was defined by creating an input layer that accepted string data. The input was then processed by the RoBERTa preprocessor, followed by the RoBERTa backbone to obtain contextual embeddings. The embeddings were pooled using GlobalAveragePooling1D, and then normalized using UnitNormalization. A Keras model roberta_normal_encoder is created with the processed inputs and normalized embeddings as outputs.
A custom cosine similarity layer was created to compute the cosine similarity between two vectors; the RegressionSiamese model uses this custom layer along with a shared RoBERTa encoder to process the pairs of sentences by splitting the input into two sentences, encoding each sentence using the RoBERTa encoder, computing the cosine similarity between the embeddings, and outputting the similarity probabilities.
With this model we obtained an accuracy of ~0.708.

BERT Word Embeddings

The last model used was DeBERTa (Decoding-enhanced BERT with disentangled attention), a variant of BERT that includes enhancements for better performance on natural language understanding tasks. To use BERT we needed to conform the dataset to a format suitable for BERT-based models, which was done by concatenating the two sentences with special tokens [CLS] at the beginning and [SEP] between the two sentences and at the end. A DeBERTaV3 classifier for binary classification was then set up with the sparse categorical cross entropy loss function for handling raw logits and the Adam optimizer. Additionally, the model's backbone is set to be trainable, allowing for fine-tuning of the pre-trained DeBERTaV3 model on the specific dataset to improve task-specific performance.
This model gave an accuracy of ~0.933.

Enhanced Sequential Inference Model (ESIM) (not in Python Notebook)

After reading from the developers' report the good results that had been obtained from the ESIM model we decided to explore it.
The starting point was to load and preprocess the PAWS dataset using the HuggingFace datasets library, tokenizing the sentences with the BERT tokenizer, and ensuring uniform sequence lengths through padding. The ESIM model was then implemented, consisting of embedding, encoding with bidirectional LSTMs, local inference modeling via attention mechanisms, and inference composition.

Despite ESIM's robust architecture and its successful application in the original PAWS dataset paper, our implementation achieved a validation accuracy of only 0.5576, leading us to exclude it from the final project. Some possible reasons for the model's suboptimal performance could include insufficient hyperparameter tuning, potential overfitting or underfitting, and limitations in the dataset's ability to capture diverse paraphrase nuances. For the future, we could try to re-implement it using more extensive hyperparameter optimization and experimenting with more recent transformer-based architectures like BERT or RoBERTa.
At the following link there is the code for the application of ESIM model on PAWS: ESIM notebook

Results and Analysis

Models	Accuracy
Bag of Words (BoW)	0.523
Embeddings layer and double input	0.601
RoBERTa with Siamese Network	0.708
DeBERTaV3	0.933

Of the four models we implemented the best-performing one was by far DeBERTaV3. The low results obtained by the BoW implementation were expected, as it is very difficult for a simple bag of words model to understand the nuances of different semantics of phrases with high word overlap.
All the other models also performed as expected, with the DeBERTaV3 model, which proved to be the most suited among the implementations for this task.

Conclusion

This project aimed to address the challenge of paraphrase identification using the PAWS dataset, which is known for its complex sentence pairs. We explored a range of methodologies, from traditional to advanced machine learning models, to determine their effectiveness in identifying paraphrases.

1. Bag Of Words (BOW):
   As expected, this procedure did not provide great results, due to its inability to properly capture semantic relationships between words, especially because many words overlap in our dataset.
2. Embedding Layer and Model with double Input:
   This method wants to exploit the Deep Neural Architecture to continuously learn embeddings from our data. This approach provided a significant improvement in accuracy, even though it still had limitations in capturing complex paraphrases.
3. RoBERTa with a Siamese Network:
   The RoBERTa encoder combined with a Siamese network architecture had a better performance, even though it was expected to perform much better. This is probably due to the fact that the original model was designed to fit a different problem.
4. DeBERTa V3:
   The last model significantly outperformed the other models. Its enhanced attention mechanisms and the fine-tuning on the PAWS dataset allowed it to capture the details

necessary for performing our task properly.

5. ESIM:
   Even if the robust architecture would suggest some good results for this model, our practical implementation did not perform well, having a performance very similar to the BOW method.

The results we obtained want to demonstrate how advanced models based on Transformers perform much better than the rest. The sequential nature of text and the role of context played a crucial role in our task, and Transformers proved to understand these features much better.

In order to further elaborate this project one could perform some Hyperparameter tuning on our model to potentially improve performance, particularly for our models that performed poorly. Also, one could explore different and newer architectures to further improve the performance.

In conclusion, the project demonstrated the ability of different Machine Learning techniques to identify complex paraphrases on the challenging PAWS dataset. While more traditional models provided a useful baseline, the more advanced ones significantly outperformed them, showcasing the power of Transformer-based architectures.

AI Policies

We used the help of ChatGPT-4o for the following tasks:

- Model Architecture Development, e.g. ESIM:
  Query: “I am developing an NLP project on the PAWS dataset, I have tried using various models like BOW, ROBERTa in combination with the Siamese network and finally I would also like to implement an ESIM model. How would you suggest implementing it?”
- Errors and unexpected behaviors solving:
  Query: “[[CODE]] How to fix this?”
- Consistent structure of the code:
  Query: “[[CODE]] Make this code look better but don’t change how it works”
- Graphic representation of the dataset:
  Query: “suggest to me some graphics to describe paws dataset”
- Outline of the Project Report:
  Query: “What can I talk about in a project report that has the following structure: Introduction, Data, Methodology, Results, Conclusion. The project is about Natural Language Processing and the associated dataset is PAWS”
- Some other prompts of minor relevance.