Foundations of Machine Learning Final Report: Paraphrase Identification

Christine Cho

October 6th, 2022

1 Introduction

The goal of this project was to train a multi-layer perceptron model for the task of classifying whether or not two sentences were paraphrases of each other. I worked on processing the data given, feature extraction, finding which model was best for accuracy, and tuning the model to improve performance. All of this was done in Google's Colab environment.

2 Features

The features and methods of feature extraction were the same from the midterm project. They were taken from different papers and GitHub repositories regarding paraphrase identification. These features include:

Total number of words in a sentence

The sentence length

The absolute value of the difference between sentence lengths

The longest common sub-sequence

The number of unique words shared by the two sentences

The total number of words from both sentences combined

The ratio of unique words to the total number of words

The minimum edit distance

The euclidean distance

Cosine similarity

Containment values using different n-grams.

3 Data Preprocessing

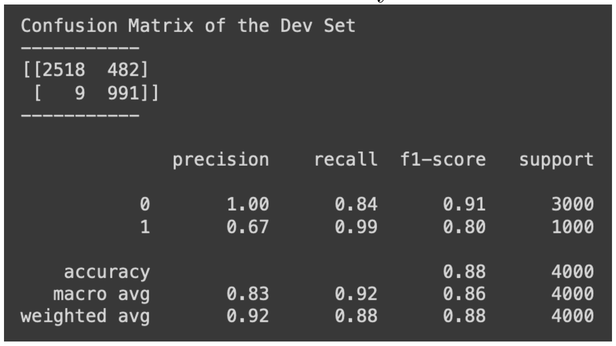
The test, train, and dev text files that were provided were read in and converted into DataFrames using the pandas library. The sentences in each row were stripped of punctuation and white-space at the front and end. Functions were created to extract features and store them in their own DataFrames. Most of the features extracted were used except for some containment values of different n-grams. The features were also standardized for scaling. The code used to process the data was the same from the midterm project. Since the feature extraction takes a long time with bigger datasets, it was easier to create the features once, save them in csv files, and load them when they were needed.

4 Algorithms and Libraries

Algorithms to calculate the features like cosine similarity, euclidean distance, etc, were provided by libraries like sklearn and scipy. Other algorithms like finding the containment value were implemented by hand and found from other papers and Github repositories. I used the PyTorch library to create different neural network models, DataLoader objects and Tensors. Since the data was imbalanced (More negative instances), I used SMOTE (Synthetic Minority Oversampling TEchnique) to over-sample and create new samples for the positive class.

5 Results

I ended up with a neural network made of ReLu, Sigmoid, Dropout, and BatchNorm layers, trained with a Binary Cross Entropy loss function, Stochastic Gradient Descent optimizer, 20 epochs, and a learning rate of 0.01. The F1-scores for the model on my dev set were:



```
Confusion Matrix of the Dev Set
-----------
[[2518  482]
 [   9  991]]
-----------

              precision    recall  f1-score   support

           0       1.00      0.84      0.91      3000
           1       0.67      0.99      0.80      1000

    accuracy                           0.88      4000
   macro avg       0.83      0.92      0.86      4000
weighted avg       0.92      0.88      0.88      4000
```

With this project I gained more experience in using the PyTorch and deep learning models. With the previous project, the focus was on the feature extraction, but this time, I was more focused on tuning the model with adjustments to the learning rate, layers, number of epochs, etc.