Ben Scuron

Professor Wenpeng Yin

Foundations of Machine Learning

27 October 2022

Paraphrase Identification

Table of Contents

1. Project Description

The goal of this project was to create a model that can predict whether two sentences are paraphrases. Paraphrases are sentences or phrases that convey the same meaning using different wording. An example of paraphrases would be: "The weather is bad" and "The weather is not good". Both sentences have a negative connotation associated with the weather. The model created in this project will be trained on labeled data, evaluated on labeled data, and make predictions on unlabeled data. The model should have accuracy better than 50%, given that there is a 1/2 chance of predicting the right label if predictions are made at random.

2. Model

The support vector machine (SVM) model was used to make predictions. I specifically used the support vector classifier (SVC) from sklearn. The classifier separates data points using a hyperplane with the largest amount of margin. The hyperplane that is created based off the training data is then used to make new predictions on unlabeled data.

3. Data

There are three different sets of data in this project: 'train_with_label.txt, 'dev_with_label.txt' and 'test_without_label.txt'. The training set has 4077 samples. Each sample consists of a unique identifier, the first sentence, the second sentence and whether the sentences are paraphrases (gold label/ground truth). The development (dev) set contains 724 samples. The test set contains 1000 samples, but does not include the gold label, since the test set is what we will make predictions on. It is important to note that the data in both the dev set and test set are balanced, meaning there is an equal number of positive and negative gold labels.

4. Data Preprocessing

The first step in the development process to create this model was to read the data into a data structure. I used the pandas library's data frames to store data from the training, dev and test sets. After reading in the data, I had to preprocess it so that the model can find the optimal hyperplane. Preprocessing the data consisted of lemmatizing and removing punctuation. Lemmatization is the process of converting every word to its base form of all its inflectional forms. I also tested removing stop words, but found that it produced poor results, so I decided to leave that step out of the final version.

5. Features

I used 7 features to feed into the SVM model. The features I used were: Levenshtein distance, cosine similarity, length difference between sentences, the number of shared words between sentences, the number of shared parts of speech between sentences, the nist score, and bleu score.

Levenshtein distance is a metric to determine the difference between sentences. Levenshtein distance calculates the number of edits a sentence would need to make to be equal to the other sentence. For this reason, Levenshtein distance is also known as edit distance.

Cosine similarity is another metric that is used to compare sentence similarity. Cosine similarity measures the similarity between two vectors of an inner product space. Since cosine similarity operates on vectors, each sentence must be converted to a vector. To complete this task, I converted each sentence to its TF-IDF vector representation. The corpus used was each sentence in the training data set.

The number of shared words was calculated by keeping a count of how many times each word appears in each sentence. There were two dictionaries to keep track of the count. The first dictionary would store the counts of words from sentence 1, and the second dictionary would store the counts of words from sentence 2. After the words were counted, the common words from each sentence were collected into a set, and the minimum count for each word in the dictionary was added to a list. In this way, a sentence: "I love data science" and "I hate data science" would have a shared words count of three words: "I", "love", "data science".

The next feature that was created was the number of shared parts of speech between sentences. This feature was implemented using the part of speech tagger from the nltk library. A similar technique to the number of shared words was used to keep track of the same number of parts of speech.

The last two features implemented were the nist score and bleu score. Both scores use n-gram co-occurrence statistics to evaluate the quality of sentence translation. While not traditionally used for this task, I found that using these scores improved my dev accuracy substantially.

There are other features that I implemented that I decided to not include in the final model. The reason why I did not include all the implemented features is because they did not have a good effect on dev accuracy. Examples of extra features were: 'ratio', 'partial ratio', 'token sort ratio', 'Ratcliff Obershelp score', and 'Jaccard similarity'.

6. Algorithms and Libraries

I used a variety of libraries throughout the implementation of this SVM model. The libraries I used are pandas, numpy, ssl, nltk, string, Levenshtein, sklearn, and difflib. The algorithms that were used in this project were mostly abstracted away in the form of imported library functions. The SVM model was used to make predictions using the features discussed in the last section.

7. Experience

Overall, I learned a lot throughout the course of this project. I learned how to: read in data, clean/preprocess data, extract useful features, fit data into a model, make predictions using a trained model, and evaluate performance. Another lesson that I learned is that a lot of data science is trial and error. For instance, I implemented many more features than I used in the final model. This is because only some features have the desired effect of improving accuracy. This was found through trial and error, by running the model many times with different features. I also obtained a deeper understanding of how to implement a solution to a real-word problem. I think it is impressive that I was able to achieve ~73% accuracy on the dev set, which is much higher than I thought I would be able to do. A lesson that I learned that is not specifically technology focus would be time management. Since I had a month to complete this project, it would have been easy to just wait until the last minute. However, I decided to work on it incrementally over time and I think that plays a huge role in my overall accuracy.