COMP9318 18s1 Project

Final Project Report

Group: BayesBlessUs
Author: Tianpeng Chen z5176343
Fengting Yang z5089358 (Group Manager)

Introduction

This report briefly illustrates the efforts we made to accomplish this project. Our task is to fool a target classifier by building an algorithm to modify tokens in the given test data.

The target classifier is a binary classifier belonging to the SVM family, which divides data into two categories, **class-0** and **class-1**. Given part of its training data, a sample of 540 paragraphs, 180 for **class-1** and 360 for **class-0**, our program will use these data to train an SVC model and modify data according to the result.

Our SVC will help us to decide the tokens to delete from or add into the test data. That is, if the SVC predicts a symbol belongs to the positive class, this symbol will make the sample more likely to be classified to **class-1** by the target classifier, which suggests us to delete this symbol from this sample. By contrast, it suggests us to add some symbols into the sample which are classified to the negative class.

Vectorisation

Before training our model, we use a Tfidf vectoriser from module *sklearn.feature_extraction.text.TfidfVectorizer* to vectorise our training data (the sample of 540 paragraphs), which afterwards gives us a matrix of TF-IDF features for model training.

Model Selection and Training

To fool a classifier, we need some reference to modify the test data. A linear SVM is a good choice. After training, this model will identify a hyperplane which divides the sample dataset into two classes (positive and negative). It also gives us an array of weights assigned to all features, which helps us to decide what tokens to modify in the test data.

Determine Optimal Parameters

Since we are using a Linear SVM model, the only parameter having a significant effect on our model is 'C'. We used a grid search module from *sklearn.grid_search.GridSearchCV* to determine the optimal value of parameter 'C', and hardcoded this parameter with the optimal in our final *submission.py* program to reduce the running time.

Token Modification Strategy

Fed with the sample dataset, our linear SVM will give us a weight array indicating weights for all tokens in the sample dataset. After several attempts, we finally decide to modify the test data by:

1. For each sample, calculate the weight for every token from weight array. If it doesn't appear in the sample dataset, the weight will be assigned 0.
2. Get all tokens with positive weights (which means their occurrence is more likely to let this sample classify as **class-1**). Then delete them from the sample by descending weight until 20 tokens have deleted.
3. If there are insufficient tokens to be deleted, we now consider adding some tokens in the sample. From the weight array, get some tokens which:
    a. Have the highest negative weights (which means their occurrence is more likely to let the sample classify as **class-0**).
    b. Don't appear in the sample.

In conclusion, for each paragraph in test data, we delete tokens which are more related to **class-1** as much as possible (up to 20). For the rest, we add tokens which didn't appear in the sample and are more related to **class-0**.

Algorithm Illustration

1. Vectorise the sample data (class-0.txt and class-1.txt combined)
2. Train a linear SVM model with the vector, get a weight array.
3. From the weight array, get a list of candidates by sorting their weights ascending. If we are unable to delete 20 tokens in a sample, we will add tokens to the sample from these candidates.
4. Now modify the data following the algorithm above.