Developing an Ensemble Classifier to Generalize Sentiment Analysis Across Different Domains

Zane Denmon, Bin Dong, and Dillon Schetley

Introduction

Within the field of Natural Language Processing, sentimental analysis is a large problem with a wide range of specific cases and applications. Sentimental analysis is the process of identifying, analyzing, and classifying a piece of text to extract the author's overall attitude, usually on a particular topic. Some particularly apparent and useful domains of sentiment analysis are reviews, articles, and tweets. Being able to quickly and accurately extract sentiment from a review would enable automatic, large-scale recognition and organization of these reviews, making these often massive datasets more useful and easier to understand. Articles could be examined for skew or emotional influence. Tweets could be grouped and accessed by the user's emotional response to a particular event.

In prior work in the field, most of the sentimental analysis tools are implemented to focus on a particular domain. The model is trained on a subset of the data collected from a particular group of texts from one or a group of similar sources. This can result in models which perform very well on their specific dataset, but these models will not always achieve comparable performance when tested on texts that do not follow the same format or cover the same subject matter as the original. In order to solve this problem of model flexibility, we created a model which takes multiple distinct datasets and trains on them all with the goal of improving performance when tested on data of a format or subject matter not seen in the training data. To

achieve this, we created an ensemble classifier in which each corpus is fed to a different model for training. Using different types of models for each dataset allows for increased flexibility in terms of which corpus is given to which model. Those models are then combined using the ensemble technique voting.

The collections of data we used for our experiments are four sets of reviews: Yelp restaurant reviews, IMDB movie reviews, Amazon product reviews, and tweets collected from Twitter. The tweets are used as the test set while the other three are used for training because the tweets differ most from the other datasets. The three models we use are logistic regression, random forest, and naive Bayes. Using this train/test split and these models, we run experiments seeing if using our ensemble classifier on the three datasets results in better performance on the test set of new data.

Background (Optional)

Approach (Methods)

Before we begin to create an ensemble classifier, we must consider our possible datasets. We came to the conclusion of using Yelp, IMDB, Amazon, and Twitter Reviews strictly because of the areas that they covered and the availability of the annotated data. Out of the four reviews, three are selected to be the training set with the last one being our testing set. We have selected Yelp, IMDB, and Amazon reviews to be our training set due to the fact that the data that the dataset provided was more rich (longer sentences). Yelp reviews focuses more on the restaurant side of services, including food, customer services, and user experiences. IMDB focuses on the entertainment domain, specifically movies. Amazon focuses more on products, instead of

services or entertainment. The combination of these three will allow us to have a large coverage area, spanning from services to entertainments and products. Twitter is chosen to be our training set mainly because it is a social-media platform and the user reviews can be on anything.

Once we have hand-selected and divided our training and testing corpus, we begin our preprocessing phase. We decided that at least in terms of preprocessing, we would preprocess all four datasets the same exact way. What this means is that if we preprocessed a certain string from Yelp dataset, we would also preprocess that same string from IMDB, Amazon, and Twitter dataset. This allows us to have a more uniformed dataset between all four corpus as well as a more uniformed feature extraction.

Between all four datasets, we have preprocessed out emojis, URL links, @ tags, and raw html code. After preprocessing, we have to pick which classifiers we would want to include in our ensemble. Most related works ensemble uses three of the four award winning models, BBSV (Bagging, Boosting, Stacking, and Voting). The python packages we use, Scikit-learn, only has support for Bagging, Boosting, and Voting. As a result, the three models that we implemented are Bagging, Boosting, and Voting. Each model will only be trained on one unique corpus (either Amazon, Yelp, or IMDB corpus).

To implement our ensemble classifier, we chose to utilize the BaggingClassifier for bagging, the AdaBoostClassifier for boosting, and VotingClassifier for voting from the sklearn.ensemble class. The input for these models requires an array-like sparse matrix. As such, we decided to utilize a TF-IDF representation over One-Hot encoding, such that our encoding would be more context-rich and keep the general sequence of the input sentences.

For each classifier, there are 2,000 data randomly sampled without replacement to form a smaller dataset from their assigned corpus of 50,000 data. Each dataset trained on those 2,000. For each model after training, we would test them on their trained data to produce their percentage of accuracy (how well they are trained). Those accuracies would be used in a softmax equation to produce a weight associated with the classifiers to be used in the overall ensemble. For example, say classifier A, B, and C produces accuracy of 95%, 96%, and 91% respectively, then classifier A would be assigned a weight of .0.3366, classifier B would be assigned a weight of 0.3399 and classifier C would be assigned a weight of 0.3234.

After obtaining the weights, we wrap all three classifiers into one, which is our ensemble. Each classifier would predict on the test data. For each test data, the three classifiers would return the predicted probability of positive label and negative label. We take each positive label percentage from the three classifiers and we multiply it by the classifier's weight. After multiplying, we sum the results and if the sum is greater than 0.5, then the ensemble classifier would classify it as positive. If it is less than 0.5, the ensemble classifier would classify it as negative.

To achieve the best training accuracy for the bagging, boosting, and voting classifiers of our ensemble, we first tuned the hyperparameters using an adaption of exhaustive grid search, which trained and tested the model based on each possible value the applicable hyperparameters could be, and reported back the best arguments to instantiate the classifiers with. Further optimization for our models involved deciding which of the three classifiers performed the best with each of the three training datasets. This was achieved by training and testing our models on each different configuration of the three training datasets.

After we have our classifiers fine tuned and combined into one ensemble as described above, we test the modifier on the training data from twitter and the results are shown in experiment results.

Experimental Setup

Our experiment is to determine whether or not an ensemble classification system can generalize to a new domain by training each of the individual classifiers on three different sources of the same domain. Since our system relies on three datasets of similar domain and one dataset of a different domain, we accumulated four different datasets total. For the datasets of similar domain, we chose to collect data from internet reviews; specifically, from IMDB movie reviews, Amazon product reviews, and Yelp restaurant reviews. For the single dataset of different domain, we chose to collect tweets from Twitter. Once the data was collected, we preprocessed each of the four datasets similarly, such that we removed HTML tags, emojis, URLs, and @tags. To determine the weight each classifier of the ensemble contributes to the prediction during testing, we utilize the training accuracy score of bagging, boosting, and voting. During testing itself, we utilize both accuracy as well as f1_score with macro averaging to evaluate the efficiency of our ensemble classifier.

Experimental Results and Discussions

TBD, we have not finished our experiment yet.

Related Work

In one study, the researchers use bagging, boosting, and stacking to create a model ensemble for classification. During evaluation, accuracy is calculated based on the number of correct percentage of classifiers and root mean squared error. Like our work, this study utilizes

multiple datasets for training/testing, however, our project differs in that we interleave the data from different datasets and train each model on a different dataset. Then, during evaluation, we weigh each model's output during testing based on the softmax of their accuracy achieved during training. We believe that our methodology will lead to a multi-model system that can more easily generalize to a new, unseen datasource. What's interesting is that they note that one model performs better than the other two models, however, they do not account for this in their final prediction during testing as we do. They concluded that their multi-model system worked much more efficiently than their baseline one-model system.

Another study utilizes both a bag-of-words and feature hashing representation to create a model ensemble for classifying sentiment analysis using tweets from Twitter. The research links to another research study that lists three reasons for using an ensemble based system: statistical (different classifiers combined for a final output), computational (relieves the issue of a global optima that may be an issue for a single model), and representational (certain tasks may be difficult for only a single model to perform). This study, too, combines the final output using a summation of the polarity of the class and class probabilities to make a final prediction. Our approach differs slightly, since we're training and testing our models using different corpora at training time and combining the results in a novel way as well. The study proved conclusive that a multi-model system performed better than a standard single model system, however, we believe since we're testing our model on a different, unseen datasource, it will perform better than their ensemble model.

Finally, another study explores using an ensemble of Part of Speech (PoS) tagging and Word Relation (WR) based feature sets being trained on a Naive Bayes classifier, a maximum entropy classifier, and support vector machines (SVMs). These feature sets and classifiers are

then combined using fixed combination, weighted combination, and meta-classifier combination. The paper then compares the performance of various combinations of these elements on five sentiment analysis data sets. Our goal is slightly different, as we aim to evaluate the flexibility of a certain model when applied to different genres of data, but the study is similar to ours in that it initially combines various models to achieve superior performance when compared to one individual model. This study also concluded, like the one above, that ensemble models generally perform than their single model counterparts, but it did not cover what we want to explore, which is how ensemble models trained on certain data generalize to other genres of data.

Conclusion and Future Work

TBD, we have not finished our experiments to draw conclusions