Developing an Ensemble Classifier to Generalize Sentiment Analysis Across Different Domains

Zane Denmon[1], Bin Dong[2], and Dillon Schetley[3]

[1]zdd3@pitt.edu

[2]bid7@pitt.edu

[3]dis43@pitt.edu

[1,2,3]University of Pittsburgh

April 27, 2019

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[1], IMDB movie reviews [5], Amazon product reviews [11], and tweets collected from Twitter [3]. 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.

Related Work

Other papers have explored ensemble learning methods for sentiment analysis and other problems. In Verma & Mehta 2017 [9], 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.

da Silva, Hruschka, & Hruschka 2014 [8] 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, Xia, Zong, & Li 2011 [10] explored 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

[1] The Yelp dataset we used can be found at https://www.yelp.com/dataset/challenge

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.

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, different compared to 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 [6], 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).

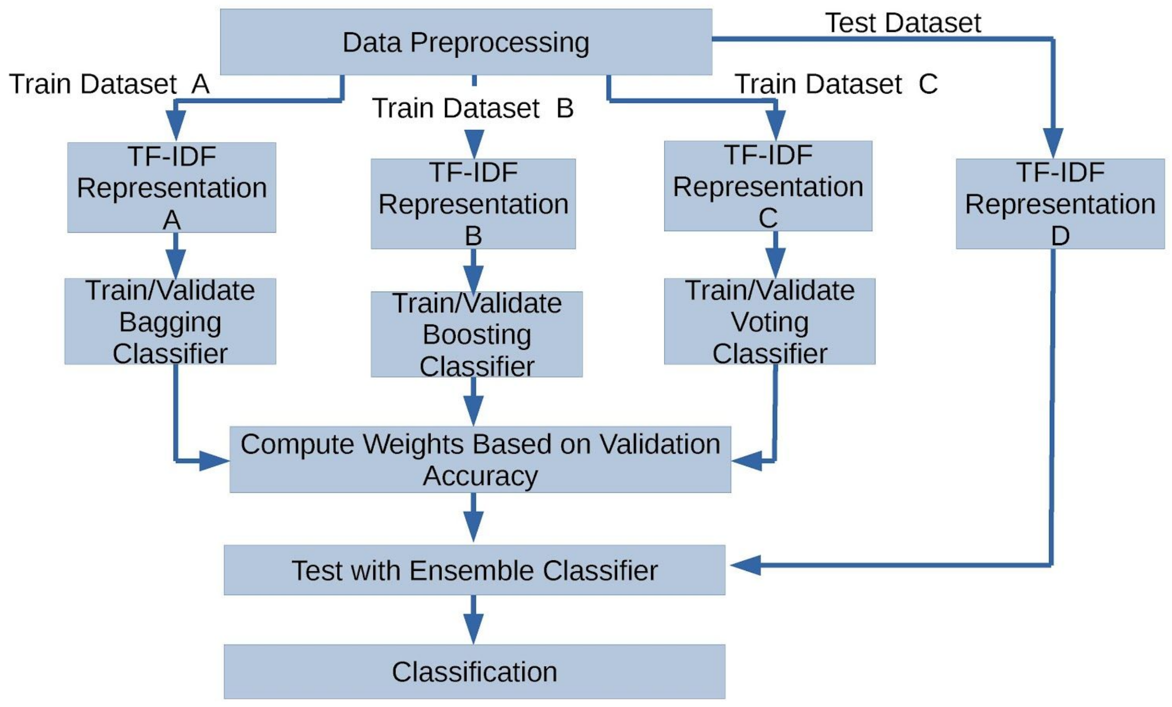

Figure 1: Ensemble Pipeline

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 [6]. 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.

The first piece of our ensemble classifier utilized the Bootstrap Aggregating (commonly known as Bagging) meta-algorithm first discussed in Breiman 1996 [1]. The idea of bagging is to mitigate a models overfitting, and this is achieved by sampling the training data with replacement for each of some number M estimators. In our model, each of these is a decision tree estimator which is then trained on the training data subset. After all the decision trees are trained, the classifiers are combined using voting to give an overall prediction.

The second classifier in our ensemble utilized Adaptive Boosting (referred to as AdaBoost), a variant of Boosting introduced by Freund & Schapire 1997 [2]. Normal boosting is a meta-algorithm for taking the output many weak learners (in our case, decision trees) and combining the outputs in a weighted sum based off of how effective a particular classifier was performing. AdaBoost adaptivity comes from, through multiple iterations of training, weighting misclassified samples more heavily such that future iterations have a better chance of accurately classifying them.

The final part of our ensemble classifier utilized voting to combine the outputs of three other classification models. More specifically, we utilized “soft voting,” wherein the overall output is a set of probabilities for each class being predicted. The models we selected were logistic regression, random forest, and naive Bayes. Each of these models have been used individually in sentiment analysis tasks in the past, and in particular naive Bayes has performed well in ensemble sentiment classification models in other works such as Xia, Zong, & Li 2011 [10].

Due to hardware limitations (see **Future Work**), for each adaboost, bagging, and voting classifier, we sample 3,000 data points without replacement from their assigned corpus of 50,000 data. The classifiers are then trained on 2,500 samples from the 3,000. After training, we test the classifiers on a validation set of the remaining 500 and record the accuracy. Each accuracy z_k is then used in a softmax equation to produce a weight for each classifier 1 through K. The resulting weight for each classifier $\sigma(z)_i$ is used in the main overarching ensemble for prediction.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{K}^{j=1} e^{z_j}} \mathit{for}\, i = 1, \ldots, K \text{ and } z = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

Equation 1: Softmax Equation

After obtaining the weights, we wrap all three classifiers into one classifier, which is our ensemble. Each classifier would then predict on the test data. For each test data, the three classifiers would return the predicted probability of the label being either positive or negative. 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

We have conducted many experiments with many different combinations of dataset. As explained previously, each model gets a total of 3,000 data randomly sampled without replacement from their corresponding corpus. Out of the 3,000 data, 2,500 data are used for training and 500 are used for validation. Each model will train on their validation set to obtain a weighted percentage used for the overall weighting scheme. After all the weights are obtained, the three models will be combined into an ensemble and be tested on a held-out test dataset of size 1000.

Note that all experimental runs are in sets of three and averaged together at the end.

Run	Boosting Acc	Bagging Acc	Voting Acc	Weighted Acc	Weighted F1	Unweighted Acc	Unweighted F1
1	0.87	0.858	0.84	0.591	0.59082	0.578	0.49666
2	0.902	0.856	0.848	0.604	0.59074	0.59	0.57719
3	0.84	0.84	0.83	0.587	0.55631	0.604	0.60387
Average	0.87067	0.85133	0.83933	0.594	0.57928	0.59067	0.55924

Figure 1: Baseline using Yelp Reviews on all three models

Run	Boosting Acc	Bagging Acc	Voting Acc	Weighted Acc	Weighted F1	Unweighted Acc	Unweighted F1
1	0.478	0.746	0.716	0.534	0.42949	0.591	0.58093
2	0.536	0.728	0.696	0.52	0.40606	0.53	0.51278
3	0.766	0.736	0.71	0.57	0.53831	0.55	0.52840
Averag	0.59333	0.73667	0.70733	0.54133	0.45795	0.557	0.54070

e							

Figure 2: Baseline using IMDB Reviews on all three models

Run	Boosting Acc	Bagging Acc	Voting Acc	Weighted Acc	Weighted F1	Unweighted Acc	Unweighted F1
1	0.748	0.706	0.694	0.571	0.53699	0.597	0.59434
2	0.758	0.77	0.69	0.611	0.61014	0.603	0.55214
3	0.702	0.706	0.67	0.573	0.54229	0.579	0.57842
Average	0.736	0.72733	0.68467	0.585	0.56314	0.593	0.57497

Figure 3: Baseline using Amazon Reviews on all three models

Run	Boosting Acc	Bagging Acc	Voting Acc	Weighted Acc	Weighted F1	Unweighted Acc	Unweighted F1
1	0.872	0.726	0.692	0.599	0.57841	0.558	0.52212
2	0.886	0.75	0.718	0.596	0.5611	0.525	0.497739
3	0.882	0.734	0.696	0.612	0.611381	0.574	0.51607
Average	0.88	0.73676	0.702	0.60233	0.58363	0.55233	0.51197

Figure 4: Best combination of dataset with Yelp being assigned to Boosting, Amazon being assigned to Bagging, and IMDB being assigned to Voting (YAI).

	Boosting Acc	Bagging Acc	Voting Acc	Weighted Acc	Weighted F1	Unweighted Acc	Unweighted F1
Best Run	0.886	0.75	0.718	0.596	0.5611	0.525	0.497739
Average	0.88	0.73676	0.702	0.60233	0.58363	0.55233	0.51197

Figure 5: YAI Best Run vs YAI Average

	Boosting Acc	Bagging Acc	Voting Acc	Weighted Acc	Weighted F1	Unweighted Acc	Unweighted F1
Yelp Baseline	0.87067	0.85133	0.83933	0.594	0.57928	0.59067	0.55924
IMDB Baseline	0.59333	0.73667	0.70733	0.54133	0.45795	0.557	0.54070
Amazon Baseline	0.736	0.72733	0.68467	0.585	0.56314	0.593	0.57497
YAI	0.88	0.73676	0.702	0.60233	0.58363	0.55233	0.51197

Figure 6: Average Baselines vs Average YAI

Each of the individual accuracies (Boosting Accuracy, Bagging Accuracy, and Voting Accuracy) represents the accuracy of the individual model on the validation dataset. The Weighted Accuracy and F1 Score is the value for the ensemble (after combining all three models) on Twitter dataset using the weighted scheme we proposed. The Unweighted Accuracy and F1 score is the value for the ensemble (after combining all three models) on Twitter dataset using the majority vote scheme that is commonly used.

In figure 4, we were able to see roughly on average a 5% increase in accuracy and roughly an average of 7% increase in F1 score using the weighted scheme proposed. In figure 5 where we only look at the best run directly, we were able to see a 7% increase in accuracy and a 7% increase in F1 score.

In figure 6, we can see that assigning a unique dataset to each of the ensemble methods ultimately leads to a better average score. With any classifiers, there will be some degree of learning errors. These learning errors are typically due to noise, bias, and variance. With that in mind, assigning the right dataset to the right model is crucial. If this was done improperly, a drop in accuracy would be notable. Such a case can be seen in Figure 6. Boosting is an ensemble method that improves variance but does not solve the overfitting issue. When we assigned IMDB to the Boosting classifier, the average accuracy tanked to roughly 59%. This is due to IMDB reviews being significantly longer than Amazon and Yelp leading to an increase in feature size.

An increase in feature size results in overfitting due to the learning model being unable to figure out which weights are more important than the others.

Conclusion

When we compare YAI to our baseline models, we do see slight improvements in both weighted accuracy and weighted F1 scores. We predict that with our current best configuration and more training data (greater than 3,000 samples), we would be able to see an more obvious improvement in using an ensemble classifier trained on different dataset than having an ensemble classifier trained on the same dataset for all its ensemble methods (such as bagging, boosting, and voting) such as Verma & Mehta 2017 [9] did.

Future Work

Due to hardware and time constraints, our TF-IDF embedding limited the amount of training data we could use to represent each of the three training datasets. This is due to the size of each of the TF-IDF embeddings being of size [number of samples x vocabulary]. Using more than 3,000 elements from each data set would cause us to run out of memory or result in unmanageable times for training. Future work would aim to drastically reduce the vocabulary size, or find a more efficient word embedding altogether to improve runtime performance.

We arbitrarily chose accuracy as a means of quantifying the weights assigned to each of the three classifiers' predictions during testing. It is unknown at this time how other scoring metrics such as precision and recall would change the performance of our ensemble classifier. Future iterations of our research project, however, will include a way of learning the weights assigned to each classifiers' prediction during testing by using methods such as veto voting as discussed in Shahzad & Lavesson 2017 [7] or no-regret learning as discussed in Haghtalab, Noothigattu & Procaccia 2017 [4].

References

[1] Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), 123-140. doi:10.1007/BF00058655

[2] Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119-139. doi:10.1006/jcss.1997.1504

[3] Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, *1*(12), 2009.

[4] Haghtalab, N., Noothigattu, R., & Procaccia, A. D. (2017). Weighted voting via no-regret learning.

[5] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011, June). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the*

association for computational linguistics: Human language technologies-volume 1 (pp. 142-150). Association for Computational Linguistics.

[6] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, *12*(Oct), 2825-2830.

[7] Shahzad, R. K., & Lavesson, N. (2013). Comparative analysis of voting schemes for ensemble-based malware detection. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 4(1), 98.

[8] da Silva, N. F. F., Hruschka, E. R., & Hruschka, E. R. (2014). Tweet sentiment analysis with classifier ensembles. *Decision Support Systems, 66*, 170-179. doi:10.1016/j.dss.2014.07.003

[9] Verma, A., & Mehta, S. (2017). A comparative study of ensemble learning methods for classification in bioinformatics. Paper presented at the 155-158. doi:10.1109/CONFLUENCE.2017.7943141

[10] Xia, R., Zong, C., & Li, S. (2011). Ensemble of feature sets and classification algorithms for sentiment classification. Information Sciences, 181(6), 1138-1152. doi:10.1016/j.ins.2010.11.023

[11] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems* (pp. 649-657).