**Sentiment Analysis of “RateMyProfessor” Reviews to Determine Trends in Student Evaluations**

By Natasha LaChac

CSC-480, American University

December 11, 2022

**Abstract**

Sentiment analysis has been useful to perform text classification tasks. It has been used to analyze product reviews, movie reviews, tweets, and other texts. In this paper, I aimed to build a sentiment analysis model that will look at student reviews of teaching and categorize their assessment of the professor as good, average, or poor. This study was performed using data from RateMyProfessors.

**Introduction**

Universities typically release student feedback forms at the end of the semester. These forms allow the university to determine the performance of their professors, as well as the professors to determine which methods worked best for their classes.

American University releases Student Evaluation of Teaching (SET) surveys at the end of the semester. The surveys are confidential. SET surveys come with pre-determined open-ended and closed-ended questions. Professors can customize their surveys to ask class-specific questions. Results for closed-ended questions are made available to the professor as well as university administrators. Results for open-ended questions and professor customized questions are only visible to the professor (American University, 2022).

Professors can have multiple classes per semester. Lecture halls can have hundreds of enrolled students. It can be difficult for a professor to parse through each individual response and get a sense of how the students ranked their performance. To aid end of the semester performance evaluations, I created and tested three models designed to classify student reviews of the professor into three categories: good, average, and poor. These algorithms are meant to expedite the review process.

Natural language processing has been used previously to determine review sentiment. One limitation of sentiment analysis is that there is not always a ground truth (Fang & Zhan, 2015); however, my data did have a ground truth of good, average, or poor as determined by star ratings. Thus, my project did not encounter that common hurdle. One research paper analyzed over 5.1 million Amazon reviews and found that both an SVM and Naive Bayesian model outperformed a random forest model on vectorized data (Fang & Zhan, 2015). The study also found that the researchers found it difficult to classify neutral reviews on vectorized data, and both best performing models still struggled with this class (Fang & Zhan, 2015). Another study examined movie reviews using a Naive Bayesian model and a random forest classifier based on 1,000 positive and 1,000 negative reviews from 2000. It found that the Naive Bayesian model and the random forest classifier both performed well, but the Naive Bayes model performed the best and achieved 81.45% accuracy (Baid et al., 2017)

**Methods and Materials**

This study used RateMyProfessor data collected by Dr. Jibo He. The dataset was most recently updated on March 4, 2020. This means that the data is all pre-COVID-19 and virtual learning. This is significant because teaching styles adjusted to teach online. However, given that most college classes have returned to in-person instruction, this data is likely more suitable to analyze reviews in 2022.

He’s dataset has 18 variables. This study was a text analysis and only concerned two: comment and student star. I created a copy of the dataset with only comment and student star columns. In RateMyProfessor, a student star score of 3.5-5 is good, 2.5-3.4 is average, and 1-2.4 is poor. I created a column named “quality” and labeled each review as good, average, or poor based on the student star. The algorithms were then trained to classify each comment into one of those three categories.

He’s full dataset contained over 9.5 million rows of data. That would have been unnecessarily large and difficult to work with. Instead, I looked at the smaller sample dataset that contained 20,000 rows for training and testing. I split the dataset into 25% testing data and 75% training data.

I decided to study three different models: random forest, logistic regression, and multinomial Naive Bayes. All three of these models are commonly used for text classification and appeared in sentiment analysis literature.

Before building my models, I preprocessed the data by setting parameters in TfidVectorizer. I converted the words to lowercase and removed English stop words to focus only on the relevant information. I then constructed a pipeline for all three models that included the TfidVectorizer and fit each pipeline to the training data.

I next set hyperparameters for each of the three models. I wanted to achieve the most accurate results possible and the fastest way to do that was by performing a grid search using GridSearchCV. The Naive Bayesian model was tested for the best alpha, maximum features, and ngram range. The random forest model was tested for the best maximum depth and bootstrapping, and maximum leaf nodes was set to none. The logistic regression model was tested for the best penalty and solver.

**Results**

After running the grid search and fitting each original model on the training data, I found that logistic regression and Naive Bayesian performed the best, and random forest performed poorly.

Fig. 1: Grid Search Scores

|  |  |  |
| --- | --- | --- |
|  | Grid Search Training Score | Grid Search Testing Score |
| Multinomial Naive Bayes | 0.793 | 0.7638 |
| Logistic Regression | 0.84286 | 0.7778 |
| Random Forest | 0.97546 | 0.752 |

Figure 1 shows each model’s score on the training and testing data. Multinomial Naive Bayes and logistic regression performed similarly well. Logistic regression had a slightly more accurate training and testing score. However, the Naive Bayesian model’s scores were closer to each other, indicating that the model was more consistent. The random forest model performed very well on the training data but slightly worse than the other two models on the testing data, which indicates that it was prone to overfitting, even with the best hyperparameters.

The best scores for both the Naive Bayesian and logistic regression models showed that these methods again outperformed the random forest model. The best parameters may help explain why. The best parameters for the Naive Bayesian model were an alpha level of 0.1, 2,000 maximum features, and a ngram range of (1,2). The alpha level was somewhat surprising. The alpha level in text classification models is often 1 to avoid the zero-probability issue; however, the best alpha level was lower than 1, so it seems that was not an issue for this dataset. The best score using this model was 0.77153. The best parameters for the logistic regression model were a penalty of l2 and a newton-cg solver. The l2 penalty is ridge regularization, which reduces the complexity of the model; this may have helped the logistic regression model avoid the overfitting present in the random forest model. The best score using this model was 0.7784. The best parameters for the random forest model were to not use bootstrapping and use 1,000 as the maximum depth. It makes sense that bootstrapping was not used, as random forest models already include regularization techniques. However, removing bootstrapping was not enough to prevent overfitting. The best score using this model was 0.75746.

In addition to looking at the best scores for each model, I created classification reports examine other performance measures to determine the best model:

Fig. 2: Classification report Naive Bayesian model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| Average | 0.53 | 0.03 | 0.06 | 600 |
| Good | 0.77 | 0.97 | 0.86 | 3202 |
| Poor | 0.76 | 0.59 | 0.66 | 1198 |
|  |  |  |  |  |
| Accuracy |  |  | 0.76 | 5000 |
| Macro Average | 0.68 | 0.53 | 0.53 | 5000 |
| Weighted Average | 0.74 | 0.76 | 0.71 | 5000 |

Fig. 3: Classification report logistic regression model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| Average | 0.33 | 0.05 | 0.08 | 600 |
| Good | 0.80 | 0.95 | 0.87 | 3202 |
| Poor | 0.75 | 0.67 | 0.71 | 1198 |
|  |  |  |  |  |
| Accuracy |  |  | 0.78 | 5000 |
| Macro Average | 0.63 | 0.56 | 0.55 | 5000 |
| Weighted Average | 0.73 | 0.78 | 0.74 | 5000 |

Fig. 4: Classification report random forest model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| Average | 0.24 | 0.01 | 0.01 | 600 |
| Good | 0.76 | 0.96 | 0.85 | 3202 |
| Poor | 0.74 | 0.56 | 0.64 | 1198 |
|  |  |  |  |  |
| Accuracy |  |  | 0.75 | 5000 |
| Macro Average | 0.58 | 0.51 | 0.50 | 5000 |
| Weighted Average | 0.69 | 0.75 | 0.70 | 5000 |

The logistic regression model had more consistently high precision, recall, and f1-scores than the Naive Bayesian or the random forest model. It also had the highest f1-score accuracy and outperformed the random forest model on all averages but was outperformed by the Naive Bayesian model for both weighted and macro averages for precision and recall. Overall, it still appears that the Naive Bayesian and logistic regression models are the best methods, and the logistic regression model may slightly outperform the Naive Bayesian model overall.

The one area in which none of the models performed well was in categorizing the reviews as average. In every model, the f1-score was low, and the precision score was higher than the recall score, though still not great. The higher precision than recall demonstrates that the model strict when it came to assigning reviews as average. The overall poor performance is likely due to the small support. Only 600 out of 5,000 testing cases were classified average, so the model had a more difficult time training for and then labeling that case.

I identified the top 20 most important words used in classification based on the random forest model’s parameter feature\_importances\_. Some of the words were self-explanatory, such as “worst” and “best.” Others indicated specific reasons why a review may have been classified as positive or negative, such as “easy” or “rude.” Other words such as “teacher” or “professor” were labeled as having a high importance, but in the context of the reviews are neutral and should behave almost as stop words.

Fig. 5: Top 20 most important words

|  |  |
| --- | --- |
| Word | Importance |
| worst | 0.019213 |
| great | 0.017398 |
| best | 0.011288 |
| horrible | 0.009349 |
| easy | 0.009188 |
| class | 0.008594 |
| avoid | 0.008261 |
| teacher | 0.006855 |
| awesome | 0.006579 |
| boring | 0.006428 |
| teach | 0.006174 |
| rude | 0.006037 |
| terrible | 0.006016 |
| hard | 0.005966 |
| professor | 0.005892 |
| good | 0.005646 |
| really | 0.005477 |
| take | 0.005421 |
| helpful | 0.005140 |
| fun | 0.004785 |

**Discussion**

Overall, I was able to create models which accurately classified different student reviews. The Naive Bayesian and logistic regression models were the best models for evaluating student reviews of teacher performance. This was like the Amazon study, which indicated that the random forest classifier would work poorly on vectorized reviews (Fang & Zhan, 2015). This differed from the movie review study, which said that the random classifier performed second best with 78.65% accuracy (Baid et al., 2017). This may have differed from my finding because the movie dataset did not have a category for neutral reviews, whereas my dataset and the Amazon reviews both dealt with neutral data.

**Limitations**

The origin of the data somewhat limits the scope. The reviews were pulled from RateMyProfessors; however, professors would ideally be using an algorithm to look at formal student reviews submitted to them directly. RateMyProfessor suffers from a degree of self-reporting bias, which means that students who report were more likely to have strong positive or negative feelings towards the professor. This was further supported by the low support for average reviews observed in the classification reports. Future version of the model should attempt to access formal student feedback to train a more balanced model.

The paper also used original models. In the future, it would be interesting to see how a pre-written algorithm such as word2vec would perform on the data. It is possible that there are better models for classification. Additionally, it would be interesting to ascertain the weight of each word based on the model type. I was able to pull the word weight from the random forest classifier, but that was also the worst performing model. Comparing the weights between models could be informative of their behavior as well as the most important student impressions which lead to the categorization of the professor’s performance. Based on this table, it would also be beneficial to remove other words such as “teacher,” “professor,” or “class” from the data before training the model, as they had a high importance but within the context of the data are not as useful for determining sentiment.

**Literature Cited**

American University. (2022). *Student Evaluation of Teaching*. https://www.american.edu/provost/oira/set/

Fang, X. and Zhan, J. (2015). *Sentiment analysis using product review data*. Journal of Big Data 2, 5. https://doi.org/10.1186/s40537-015-0015-2

He, Jibo. (2020, March 4). *Big Data Set from RateMyProfessor.com for Professors’ Teaching Evaluation*. Mendeley Data, V2, doi: 10.17632/fvtfjyvw7d.2

Baid, Palak et al. (2017, December). *Sentiment Analysis of Movie Reviews using Machine*

*Learning Techniques*. International Journal of Computer Applications (0975 – 8887)

Volume 179 – No.7. https://www.researchgate.net/profile/Neelam-Chaplot/publication/321843804\_Sentiment\_Analysis\_of\_Movie\_Reviews\_using\_Machine\_Learning\_Techniques/links/5b6cc3b6299bf14c6d97d82a/Sentiment-Analysis-of-Movie-Reviews-using-Machine-Learning-Techniques.pdf