Compsci 349 final
Alex Cates Atc8943

Expanding Course Evaluations with Natural Language Processing

Course evaluations are common across different institutions. Every year thousands of students across the country rate the course they just completed and the professor you taught it. While these evaluations have been shown to have inherent biases (Felton et al. 2003, Reid 2010, Rosen 2017), they are still commonly used because of the ease of implementation and review. By rating professors on a simple scale (such as the 6 point scale used in Northwestern CTECs), it is easy to aggregate these ratings across students, producing mean scores which are used to inform decisions such as new teaching practices and whether or not to award tenure. However, most course evaluations often provide free response space for students to write in their own comments. Student comments provide important nuance, and while they are often read over, methods of aggregating the information contained in these comments are lacking, normally boiling down to the professor's own intuition. Natural language processing offers a method to computationally analyze comments to better understand the information contained in the comment and provide important depth and nuance to the simple course rating number.

The present work uses rating and comment data from RateMyProfessor.com, a third-party professor rating site, which has been shown to be an accurate representation of university led course evaluations (Rosen 2017, Clayson 2014). First, I try to predict whether a professor has a positive or negative rating based on their comments. Second, I compare the accuracy of this prediction compared to the general polarity of the comment. Finally, I explore how the unsupervised classifier Latent Dirichlet Allocation (LDA) could be used to provide more nuance to a star rating.

Methods

Dataset

The present analysis uses the sample data provided by (He 2020) which contains 20,000 professor ratings from ratemyproffesor.com. This data contains rating for 1413 professors from 554 different universities. In particular. All comparisons are to the star rating provided with each review.

Text Processing Pipeline

To extract and analyze the comment texts, the following steps were followed. First, all punctuation was removed, then the text was converted to lowercase and tokenized (each comment is split into a list of words). Following tokenization, stop words ("and", "it", "a", etc.) were removed. Additionally, all professor and department names were replaced with "prof_name" or "dept_name", respectively. This was done to: preserve anonymity of all professors, preserve the generality of the results, and because the mentioning of a proper name likely implies a more personal connection to the class (compared to the general professor or class). Next, bigrams (groupings of 2 words that are commonly found together such as "extra_credit") were created. Finally, the data was lemmatized, which converts each word to its base form, such as "studying", "studies", and "studied", all being stored as the word "study".

Predicting Positive or negative Star Rating

For the features, I used the lemmatized text of each comment. I then expanded the dataset into a binary sparse matrix where each column is a unique word, each row is a comment, with the binary 1 if that row's comment contained the column's word. For targets, I split all the star ratings at the median (3.7 stars), assigning a 1 to star ratings above the median and 0 to star ratings below the median. Split the data into 75% test and 25% train sets and trained a Bernoulli Naïve Bayes classifier to predict whether the comment was positive or negative.

Polarity calculation

Again, starting with the lemmatized text, I calculated the polarity of each comment. Polarity is defined between -1 and 1 and is a measure of how positive (1) or negative (-1) a comment is. I used an off the shelf polarity algorithm (TextBlob, Loria 2020) but which has shown usefulness in a wide variety of natural language processing contexts.

For comparison with the predicted positive or negative star rating from above, the confidence of the Bernoilli Naïve Bayes' classifier was manipulated to be the same scale as the comment polarity. Specifically, the confidence of the prediction (ranging from 0, definitely a negative rating, to 1, definitely a positive rating) was first multiplied by 2 (so that scores ranged from 0-2). Then 1 was subtracted from each confidence (creating confidence scores that range from -1 - 1). This manipulated score was then correlated with the polarity of the comment using a Pearson correlation.

Topic modeling

For topic modeling, I used a tf-idf weighted LDA unsupervised algorithm to identify topics. Starting with the lemmatized texts, I created a bag of words dictionary which assigns a unique id number to each word in the entire dataset, then creates tuples in each comment which contain the word id and the frequency of the word in that comment. This dictionary was fed into a term frequency inverse document frequency (tf-idf) algorithm (equation 1). Tf-IDF compares the frequency of words in a comment (or document) vs the number of documents which contain that word. Doing so down weights words that may not be stop words but are common across this specific dataset (such as "class" or "course"). The tf-idf weighted dictionary is then fed into the LDA algorithm. Similar to the KNN clustering algorithm LDA searches for groups of words that are near each other or often contained in the same comment. The algorithm outputs a weight for every word in the dataset which describes how much weight that word has for a given "topic". Because LDA is an unsupervised categorization algorithm, I did not split the data into training and testing data, instead clustering all the data based on the primary topic expressed in the comment.

Equation 1 $$W_{i,j} = tf_{i,j}\ x \log\left(\frac{N}{df_i}\right)$$

i = the current term (word)
j = the current document
$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Results

Star Rating prediction

When predicting the star rating, the overall model had train accuracy of 87% and a test accuracy of 66%. This is better than chance of 50% but not necessarily reliable. Interestingly, the model was near perfect at predicting negative comments, with most of the mistakes coming from false negatives (table 1).

	Predicted Positive (1)	Predicted Negative (0)
Actual Positive (1)	877 (17%)	1408 (28%)
Actual Negative (0)	293 (6%)	2421 (48%)

Table 1. Confusion matrix of the accuracy positive or negative star rating predictor on the test dataset.

Polarity vs positivity of star rating

When comparing the text trained prediction of a star rating to the off the shelf polarity model (figure 1), there is a weak positive correlation ($r^2 = .12$). The weakness is likely due to the issues with the model mentioned above (namely a large false negative rate).

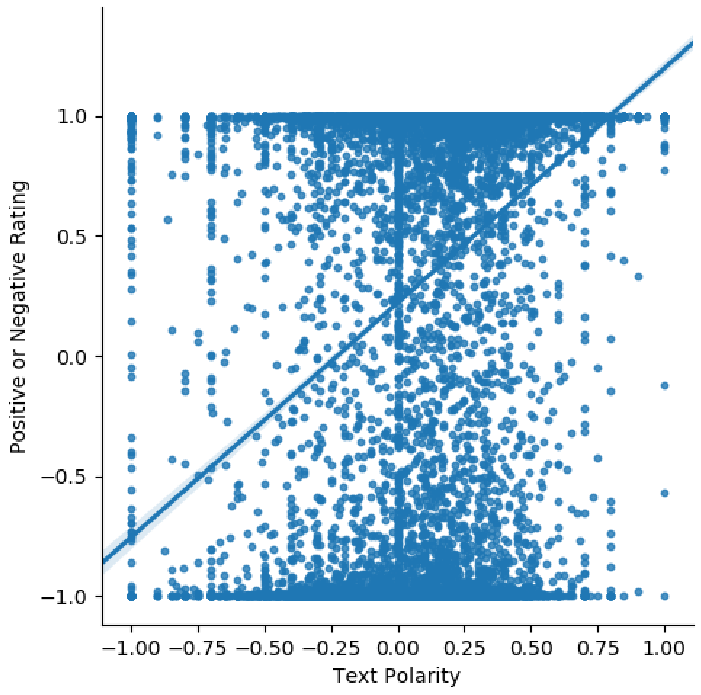


Figure 1. Correlation between text polarity and predicted rating

Text polarity vs true star rating

The relationship between the general polarity of a comment and its star rating (figure 2) is not much better ($r^2 = 0.15$). this may be due to the nature of the comments, such that they are commenting on specific aspects, but is nonetheless surprising.

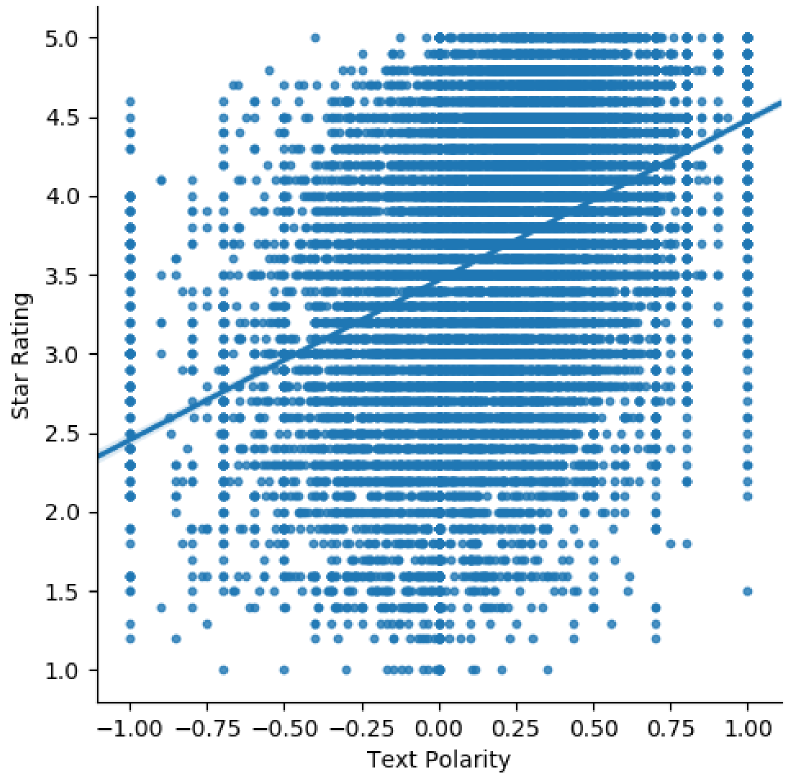
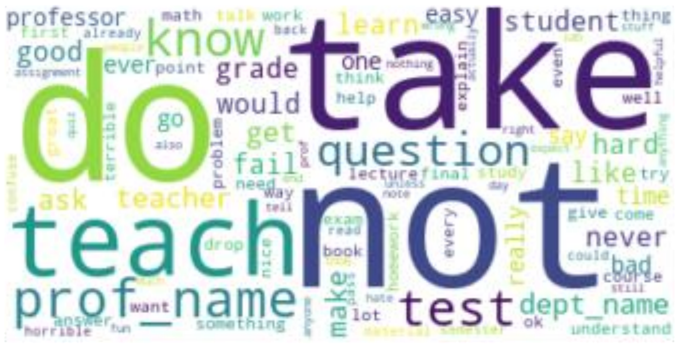
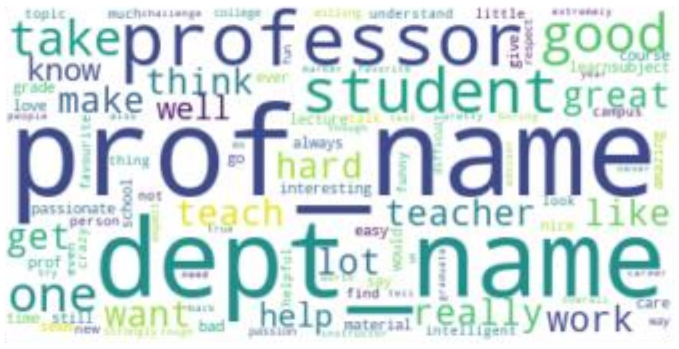


Figure 2. Correlation between text polarity and Star Rating

a. Topic 2



b. Topic 5



c. Topic 8

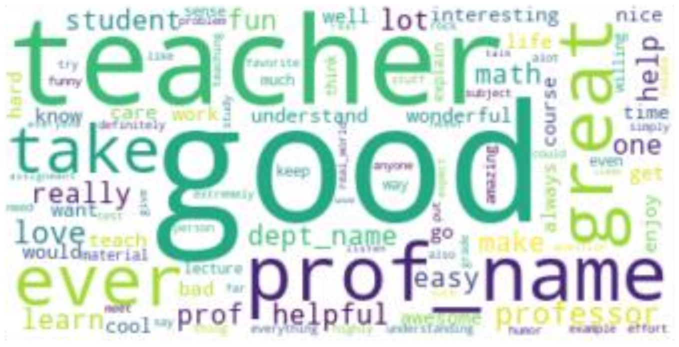


Figure 3. Word clouds of 3 representative topics. Specifically topic 2 (a), 5(b), and 8(c)

Topics

Based off the mean coherence of all possible topics, it was determined that 10 topics would be appropriate for the current set of topics. Ten topics produced the largest coherence (example Word clouds of three of the topics (topics 2, 5, and 8) can be seen in figure 3). Additionally, the primary topic of a comment is related to the overall rating (figure 4), although only 2 topics seem to significantly differ from the other 8 topics (topic 8 positively and topic 2 negatively).

Discussion

Overall, the present project displays both the promise and the limitations of using natural language processing to better understand course evaluations. First, it is possible to train a model that predicts the overall rating based on the comment. This model can likely be refined with additional data (rate my professor has millions of ratings available) and with tuning the parameters to improve the accuracy (beyond 66% reported here). That said, it is interesting that most of the mistakes were due to false negatives. This suggests that many of the comments likely focused on problems in the class or with the professor, aspects they would like to see fixed. Such details are important for professors reading comments in that there may be a disconnect between the emphasis of the comment and the students overall feeling about the class. This idea is emphasized by the weak correlations between the text polarity and the real and predicted star ratings. All of these points add up to the idea that there can be a disconnect between what the comments are saying and star rating of the professor or course.

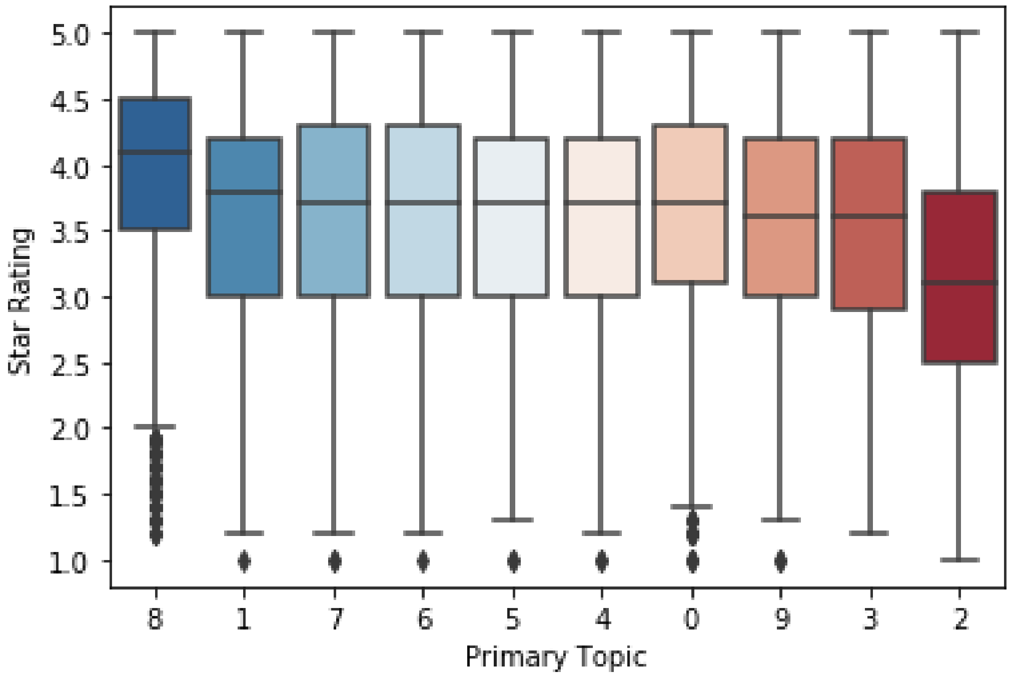


Figure 4. Boxplots of the star ratings associated with comment topics. Topics are arranged in descending order of median star rating

The hope was that the LDA model may be able to provide additional context and nuance to a comment, however, here again the results are disappointing. The overall coherence is rather low (less than 50%) suggesting that the topics are ill defined. Here the lack of data is probably the main issue. Despite this, 2 topics seem to warrant significant attention. Topic 8, which contains many positive words with large weights (e.g. “good”, “great”, “helpful”) along with prominently using the professor’s name (coded as prof_name) is associated with significantly higher star ratings than the other topics. Conversely, topic 2, who’s defining phrase looks to be “do not take”, seems to be associated with significantly lower star ratings. These may seem obvious, but it is important to remember that these topics were organically found using an unsupervised learning model. As such it is reasonable to expect that with more data to better define the topics, one could extract additional insights.

The present project provides evidence of both the promises and limitations of using natural language processing to better understand course evaluations. Models can easily be trained to predict a course rating based off the comment, which can be useful for identifying times when the comment and the reported rating do not align. Similar explorations can be made based on the polarity of a comment with a generic, off-the-shelf, sentiment analysis model. Finally, unsupervised clustering algorithms, such as LDA, can be used to identify topics in comments, however more data and parameter tuning is needed on all of these before any conclusions can be made.

References

Clayson, Dennis E. 2014. “What Does Ratemyprofessors.com Actually Rate?” *Assessment & Evaluation in Higher Education* 39 (6): 678–98.

Felton, James, John B. Mitchell, and Michael Stinson. 2003. “Web-Based Student Evaluations of Professors: The Relations between Perceived Quality, Easiness, and Sexiness.” https://doi.org/10.2139/ssrn.426763.

He, Jibo (2020), “Big Data Set from RateMyProfessor.com for Professors' Teaching Evaluation”, Mendeley Data, v2http://dx.doi.org/10.17632/fvtfjyvw7d.2

Loria, Steven. 2020, “TextBlob: Simple, Pythonic text processing. Sentiment analysis, part-of-speech tagging, noun phrase parsing, and more.” https://textblob.readthedocs.io/en/dev/index.html

Reid, Landon D. 2010. “The Role of Perceived Race and Gender in the Evaluation of College Teaching on RateMyProfessors.Com.” *Journal of Diversity in Higher Education* 3 (3): 137–52.

Rosen, Andrew S. 2017 “Correlations, Trends, and Potential Biases among Publicly Accessible Web-Based Student Evaluations of Teaching: A Large-Scale Study of RateMyProfessors.com Data.”