Professor Quality Classification using Data from Rate My Professors (RMP)

Casper Guo, Jack Yu, Manheng Wang, Jiawei Hao, and Steve Fan

1 Project Description

In this project, we are investigating the relationship between review texts and numerical ratings on Rate My Professors (RMP) (rat), which will be expressed as either "hot" or "not hot" with a custom cutoff boundary. Specifically, we will be investigating a dataset we scraped from RMP containing meta-information and ratings of professors at the University of Michigan (UofM). We will combine and train multiple machine learning models on various text encodings using this dataset to study the sentiment conveyed by the comments (used interchangeably with reviews in this project) students gave to professors.

Some interesting questions we want to investigate:

- To what extent does the language of a positive review differ from that of a negative review?
- What are some strong predictors (particular phrases or words) that strongly predict the overall sentiments of reviews?
- To what extent does the overall frequency of positive and negative words affect the average rating?
- How much does an overly positive or negative review affect the professor's overall assessment?
- In the cases where the student also provides tags (up to 3) for the professor, can we effectively predict the tags based on the review?

1.1 Motivation

As students at the University of Michigan, we want to be better informed about the educational opportunities available at the University and ensure that we use them effectively. To that end, when students are choosing instructors for their courses, many turn to RMP, where students provide comments and ratings about their experience, to understand the instructional style of professors. Normally a student would use the average rating and student comments to determine if a professor is great (hot in our context). Thus we will focus on professor ratings and comments in the context of this project.

However, it is not always the easiest to obtain information from a website like RMP. Here are some examples of poor information quality from our experiences with the website:

- A professor may have very few comments, making the validity of the average rating questionable.
- Most comments are short and give average ratings, but a small group of comments are long and overwhelmingly positive. Who should we trust?
- Comments that contain a lot of negative words or phrases (e.g. hard, difficult, complicated, harder than expected) but have an overall positive rating. How do we evaluate the tradeoff between a good professor and a difficult course?

The goal of our project is to leverage machine learning techniques to perform sentiment analysis on student comments. We will use supervised training methods with the student comments as the inputs, and student scores as labels, to predict the general sentiment of the comments and predict their associated ratings.

This project will hopefully allow us to be able to make better choices when choosing courses and professors in the future without having to sacrifice a lot of time reading through each comment for a professor, which can be especially cumbersome for professors with many ratings.

2 Related Work

2.1 Support Vector Machines

Support Vector Machine (SVM) (Cortes and Vapnik, 1995) is a supervised learning algorithm that maps input vectors to a higher dimensional space in order to construct a hyperplane that separates all mapped vectors to achieve binary classification. Each side of the hyperplane represents one class of the binary label. This algorithm is widely used in many disciplines including text classification.

Past attempts at using SVM to classify texts include determining news categories using an SVM-based classifier (Saigal and Khanna, 2020). This study first converted news articles to a Term Frequency-Inverse Document Frequency (TF-IDF) matrix. The encoding is then fed to several SVM models like Least-squares Support Vector Machines. The One-Against-All approach is used to achieve multi-categories classification. Their best model achieved 92.96% accuracy on a dataset containing 20 news categories and proved robust generalization ability.

Other prior research using SVM to process texts dove deeper into extracting sentiment clues from texts. The so-called sentiment analysis is a field of natural language processing that determines the polarity (positive, negative, neutral) of a text.

One group of researchers used SVM to conduct sentiment analysis on movie review data on Epinions.com and analyzed whether a piece of review is favorable or not towards the associated movie (Mullen and Collier, 2004). The team's model extracted unigram and lemmatized unigram features from the review texts and experimented with multiple SVM models. They achieved 86% of accuracy using an ensemble of SVM models.

In our own research, we also want to apply SVM to conduct sentiment analysis on the student reviews for a professor.

2.2 Logistic Regression

Logistic regression is a type of regression analysis used to predict the probability of a binary outcome (such as whether a customer will purchase a product or not) based on one or more predictor variables. It works by estimating the coefficients of a linear equation to model the relationship between the predictor variables and the outcome.

In text classification, logistic regression is a commonly used method. It involves representing text data as numerical features and then fitting a logistic regression model to predict the probability of a document belonging to a particular category (Genkin et al., 2005). The model is trained on a labelled dataset and can then be used to classify new, unseen documents into the appropriate category. Logistic regression can also be extended to multi-class text classification problems.

Researchers have used logistic regression to predict students' academic performance. In "Predicting Success in College: A Study of Placement Exams and First-Year Performance" (Belfield and Crosta, 2012), researchers used logistic regression to analyze the relationship between college placement exam scores and first-year academic success. The researchers used text classification to categorize essay responses to placement exams based on the essay content, and then used the resulting categories as predictor variables in the logistic regression analysis. As a result, the study found that the essay categories were significant predictors of first-year academic success. Specifically, the "Integration of Knowledge and Ideas" category had a positive effect on academic success (odds ratio = 1.18, $p < .01$), while the "Comprehension and Collaboration" category had a negative effect on academic success (odds ratio = 0.89, $p < .01$).

2.3 Random Forest

Random forest (Ho, 1995) is an ensemble learning method for either classification or regression that uses a multitude of decision trees. Random forest classifier involves each decision tree contributing a single vote towards the predicted class. The votes may or may not be weighted depending on the specific implementation of random forest. Random forest regressor uses the average prediction as the predicted value.

Researchers have used random forest to perform clinical text classification (Wang et al., 2019), where machine learning techniques were used to classify clinical narratives automatically, with the hope that human efforts could be directed elsewhere as a result. The study found that random forest, while unable to achieve the highest levels of precision and recall within the study, consistently scored between 0.85 and 0.95 on both metrics, indicating a high level of success.

In our case, we will be using both random forest classifier and random forest regressor. Whereas the classifier will predict the hotness of each student comment, the regressor will directly predict the rating associated with said comment.

2.4 Rate My Professors API

We would also like to acknowledge API written by Nobel Zhou, Christopher Bryan and Oze that made data extraction easier for users (Zhou et al., 2022). Instead of having to parse thousands of HTML documents, we used the GraphQL endpoints and query formats outlined in their repository, which enabled us to quickly and reliably gather information on a large number of professors.

3 Data

The entirety of our dataset originates from the individual professor pages from Rate My Professors (University of Michigan - all professors) (rat). This section lists the steps of data collection, processing, and initial exploratory data analysis conducted on our retrieved data.

3.1 Data Crawling

The crawling step utilizes the GraphQL endpoints implemented by RateMyProfssorAPI (Zhou et al., 2022). The required input for the endpoints is the unique pid (professor id) for each professor at UofM. The RMP page for UofM conveniently links to a listing of all professors associated with the University, totalling 5000+ results. However, only the first 8 records (ordered by pid ascending) are shown when the page is first loaded. A "Show More" button located at the bottom of the page must be clicked to load the next 8 records. To solve this issue, We used WebDriver from Selenium (**?**) to automate the process of finding and clicking the button until all professor names are listed. As we were rapidly sending a large number of requests to the website server, the scraping was interrupted a few times as the server stopped responding. Despite the difficulties, we were able to obtain 4600+ unique records. We note that this likely accounts for the vast majority of available information about UofM professors on RMP as professor records that were scraped last or were not scraped at all are created more recently and thus tend to contain fewer ratings.

3.2 Data Cleaning

After running the selenium script, we downloaded the full page HTML containing all 4600+ records and used BeautifulSoup package (Richardson, 2007) to extract the professor IDs. We then

Column Name	Note
profID	
firstName	
lastName	
department	
numRatings	Number of student ratings
wouldTakeAgainPct	Percentage of students that answered yes to "Would you take this professor again?"
avgDifficulty	Average of difficulty ratings provided on a scale of 1 to 5
avgRating	Average of quality ratings provided on a scale of 1 to 5

Table 1: *raw_prof_info.csv* schema

made POST requests to RMP GraphQL endpoints with each professor ID to acquire various information about that professor in JSON format in *data_acquisition.py*. Limited processing was performed before the JSON responses are parsed into dictionary objects, which are then converted to Pandas (pandas development team, 2020) DataFrames and stored as CSV files. The resulting two CSV files store data about professor information and rating information, which constitute our raw dataset. The detailed schema of each file is given in Table 1 and 2.

Extensive data cleaning and pre-processings are then performed in *data_cleaning.ipynb*. Some notable steps include:

1. Duplicate professor records, as indicated by identical profID, are dropped.

2. The *wouldTakeAgain* column is given in JSON as an integer (either 0 or 1), it is converted to a boolean column.

3. A *fullName* column is added for easier interpretation.

4. The *attendanceMandatory* column is given in JSON as one of the following values "mandatory", "non-mandatory", "Y", "N". It is processed into a boolean column.

5. The *date* column is given in UTC format and cannot be directly interpreted by Pandas as

Column Name	Note
profID	
class	
attendanceMandatory	
comment	Comment left by the student, if any
date	
difficultyRating	Difficulty rating provided on a scale of 1 to 5
grade	Letter grade received in the class, if the student chooses to disclose it
helpfulRating	Helpfulness ratings provided on a scale of 1 to 5
isForCredit	
isForOnlineClass	
ratingTags	Up to 3 tags given in the format tag1–tag2–tag3
wouldTakeAgain	Whether the student answered yest to "Would you take this professor again?"

Table 2: *raw_ratings.csv* schema

a datetime object. Transformations are performed so the column can be casted to the datetime type.

6. The *ratingTags* column, whose format is described above, is processed into lists of tags.

The schema of the cleaned dataset will not be repeated here. You can find it in the README file.

3.3 Interesting Data Sample

3.3.1 Data Sample I

The first sample we examine utilizes data from the top 10 departments with the highest number of ratings (Figure 1) because they provide the most diverse data. The ratings given by the students range from 1 to 5 in 0.5-point increments.

The median of the ratings is 3.9, which can be used as a general cutoff for determining whether a comment is positive, negative, or neutral. However, such arbitrary cutoffs may be misleading in some cases as there are confounding variables unique to each department. For example, one department may be less funded than another, resulting in fewer teaching resources and poorer ratings.

Another interesting observation can be made from the box plot. If we group the departments by STEM (subject related to science, technology, engineering, and mathematics) and non-STEM (in this case Psychology, English, Social Work, Spanish, Political Science, and History), we notice that the STEM group has a noticeably lower rating on average compared to the non-STEM group (median of 3.6 comparing to a median of 4.0).

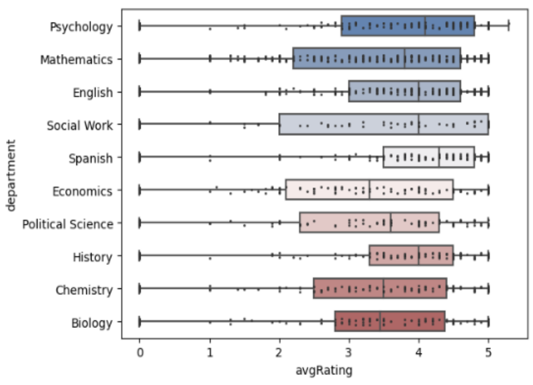

Figure 1: Top 10 departments with the most ratings

3.3.2 Data Sample II

The second sample examines the relationship between the average comment length for a professor with the professor's average rating (Figure 2). One common observation in marketing is that when people write good reviews they tend to write less for high ratings, but when they give bad reviews they write more as it's easier to find reasons to complain. We want to verify if this is true for our professor ratings.

From the graph, it's clear that there is no apparent relationship between average comment length and average rating. At all average rating levels, the average comment length is between 20-60 words. Looking at ratings of 5, we can find average student comments as short as 0 words in length and as long as over 70 words. It's interesting that when students decide to give the highest rating for a professor, some of them still write long comments to explain why they enjoy the professor's class. We think this indicates that the students have true reasons to like the hot professors which are reflected in their comments, as compared to typical evaluation where they just give high scores without reasons. We can thus say that even though ratings on RateMyProfessor.com are subjective, they are considerably authentic in reflecting student opinions.

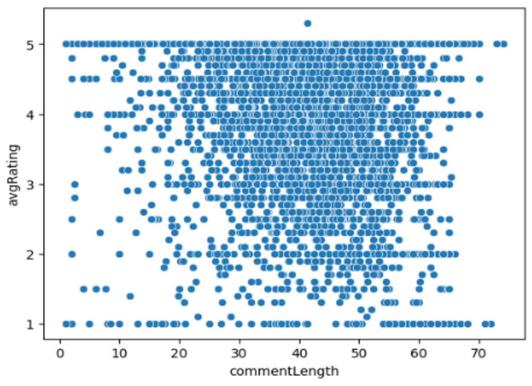

Figure 2: Average comment length versus ratings

4 Method

This section details the preprocessing and text encoding used to prepare the corpus as well as the training regimen and models used for the sentiment analysis task.

4.1 Hotness

We decided to define all comments with an associated rating of 4 or higher as "hot" and "not hot" otherwise.

4.2 Preprocessing

Before the comments are encoded, the following processing steps are performed:

1. All punctuations are removed from the comment text. While a more nuanced approach may be preferred, an examination of a random sample of 50 comments revealed that this effort has minimal undesirable effects.

2. The comment is tokenized, using the *word_tokenize* method provided by the Python Natural Language Processing Kit (NLTK). (Bird et al., 2009).

3. Stopwords, as defined in the NLTK English stopwords list, are removed.

4. While reviewing the sample of comments, we noticed that most comments contained the professor's name, the course number, or both. These tokens carry low information and are unique to each professor and class. Consequently, including these tokens will drastically increase the vocabulary size to the detriment of model performance. Therefore, we removed the first and last name of the professor (we are able to determine this as each comment is associated with an unique profID) and all numerical tokens.

5. All the tokens are converted to lower case.

6. All tokens are lemmatized, using the *WordNetLemmatizer* class provided by NLTK.

For example, the following comments are in their unprocessed forms:

- Fricke is the man. Entire class probably took five hours of studying for the entire semester. I think he may be retiring, but if not, take this class if you can fit it in.

- Tom Fricke is one of those professors you will never, ever forget. I found myself coming to every single lecture even though none of them were mandatory simply because he is quite possibly the most entertaining person on the planet. Tom cares so much about his students and I became quite good friends with him through office hours. 100/10 recommend

And the following are their processed forms:

- man entire class probably took five hour studying entire semester i think may retiring take class fit

- tom one professor never ever forget i found coming every single lecture even though none mandatory simply quite possibly entertaining person planet tom care much student i became quite good friend office hour recommend

4.3 Word Encoding

The following word encoding methods are experimented with:

1. Encoding the comments with a simple frequency dictionary. Each comment is turned into a vector whose length is equal to that of the vocabulary size and each entry is the number of occurrences of the associated word in the corresponding comment. The encoding is done with the *CountVectorizer* class of the scikit-learn library. (Pedregosa et al., 2011).

2. Encoding the comments using the standard term frequency inverse document frequency (Tfidf) score. The encoding is done with the *TfidfVectorizer* class of the scikit-learn library.

3. Encoding the comments using a vector-based word representation approach, specifically, the global vectors for word representation (GloVe) developed by researchers at Stanford (Pennington et al., 2014). We chose to use the smallest model trained on six billion tokens from Wikipedia and Gigawords 5. The model contains the vector encodings of 400 thousand unique words and offer 50-, 100-, 200-, and 300-dimensional vectors. We chose to use the 100-dimensional vectors for a balance of performance and scalability. The downloaded model is prepared with the Geosim library (Řehůřek and Sojka, 2010). We implemented a simple vectorizer that attempts to retrieve the encoding for each token in a comment and then take the average of all retrieved vectors as the representation for the entire comment. Alternatively, the sum of the token vectors or the sum of the token vectors weighted by the Tfidf score may be used.

One potential improvement to the text encodings, specifically the frequency and tfidf encoding, is to reduce their dimensionality. We included all tokens that appeared at least once in the corpus, resulting in a vocabulary size exceeding 17 thousand. This made later model fitting time-consuming. A better approach is to only add tokens that appear in at least 5 comments to the vocabulary. Doing so will result in a much smaller vocabulary size of around 5500.

4.4 Model training & Evaluation

We trialed the following types of machine learning models, using their scikit-learn implementations:

- Random Forest Classifer (RFC)
- Gradient Boosting Classifier (XGBoost)
- Logistics Regression (LR)
- Multinomial Naive Bayes (NB)
- Linear Support Vector Machine (SVM)
- K-nearest Neighbor (limited success) (KNN)
- Random Forest Regressor (RFR)

The 51,650 comments in our corpus are divided into training and testing set at 3:1 ratio. For each combination of machine learning model and text encoding, a grid search is conducted over a small search space to identify the optimal hyperparameters. The candidate models are evaluated based on the average F1-score achieved on five-fold cross-validation.

Model	**Frequency**	**TfIdf**	**Word2Vec**
RFC	0.83	0.83	0.79
XGBoost	0.83	0.83	0.79
LR	0.85	0.85	0.80
NB	0.84	0.80	
SVM	0.83	0.85	0.80
KNN	0.72		
RFR	0.78	0.78	0.68

Table 3: Accuracy

Model	**Frequency**	**TfIdf**	**Word2Vec**
RFC	0.84	0.83	0.79
XGBoost	0.83	0.83	0.79
LR	0.85	0.85	0.80
NB	0.84	0.80	
SVM	0.83	0.85	0.80
KNN	0.72		
RFR	0.76	0.75	0.59

Table 4: Recall

The best candidate is then applied to the testing set. For the classification models, we noted down their success on predicting the hotness of comments as measured by the weighted average of accuracy, recall, precision, and F1-score. For the regression models, we calculated their mean absolute error (MAE) as well as root mean squared error (RMSE). Additionally, we inferred the hotness from the predicted ratings (which are numerical and continuous) based on the cutoff of four. This enables us to also find the accuracy, recall, precision, and F1-score for the predictions made by the regression models.

5 Results

5.1 Benchmarks

We additionally note that the random forest regressor trained on Tfidf encoding achieved a MAE of 0.75 and an RMSE of 1.04.

5.2 Discussion

As the benchmarks clearly show, all models except the regression model achieved very similar performance when measured by all metrics. The best performance, as measured by F1-score, is achieved

Model	Frequency	TfIdf	Word2Vec
RFC	0.83	0.83	0.78
XGBoost	0.82	0.82	0.79
LR	0.84	0.85	0.80
NB	0.84	0.82	
SVM	0.83	0.84	0.80
KNN	0.71		
RFR	0.9	0.9	0.9

Table 5: Precision

Model	Frequency	TfIdf	Word2Vec
RFC	0.83	0.83	0.79
XGBoost	0.83	0.83	0.79
LR	0.85	0.85	0.80
NB	0.84	0.80	
SVM	0.83	0.85	0.80
KNN	0.72		
RFR	0.82	0.82	0.72

Table 6: F1-Score

by the logistic regression and SVM model trained on the Tfidf encoding.

We had originally expected the word2vec encoding, which is state-of-art, to perform the more traditional Tfidf encoding, which will in turn outperform the rudimentary frequency encoding. However, we find that the frequency encoding achieved comparable performance to the Tfidf encoding and both in turn outperformed word2vec. The reason for this is unclear.

Using the frequency and Tfidf encoding over the word2vec approach does mean accepting diminished scalability. While the word2vec encoding is 100-dimensional and constant, the other two encodings' dimensionality is equal to the vocabulary size, which is orders of magnitudes bigger than 100 and will grow logarithmically with the corpus size. In other words, as the corpus size grows, training models with frequency and Tfidf encodings will become extremely demanding time and computation-wise. On other hand, the rate of growth for the fitting time needed to train a model on Tfidf encoding only grows linearly.

6 Conclusion

In this project, we compiled a dataset of UofM professors' metainformation and student comments available on RMP. The dataset is then cleaned and we proceeded to use it to benchmark three text-encoding methodologies and seven machine learning models' effectiveness at sentiment analysis. We find that on this specific dataset, traditional word embedding approaches outperform their modern counterparts and the choice of machine learning model doesn't have much effect on the achieved performance.

7 Individual Contribution

Jack Yu: Brainstormed with the group regarding project scope and data collection. Worked on writing scripts to crawl every UofM professor from RMP and automatically hit the "load more" buttons. Conducted preliminary analysis on the raw data file and explored the relationship between review length and rating. Researched similar studies in sentimental analysis and methods using SVM and logistic regression. Worked on poster presentation Method section.

Jiawei Hao: Suggested the initial approach of using SVM to classify RMP data. Coordinated with team members to come up with alternative model ideas. Found several classification-related research work and Medium articles and wrote about their relevance in the Related Work section. Contributed to the Data section by analyzing interesting data patterns and producing visualizations with Matplotlib (Hunter, 2007) and Seaborn (Waskom, 2021). Gave suggestions in Method section by introducing feature engineering methods such as stemming and lemmatizing. Worked on the poster.

Steve Fan: Determined project goals and questions of interest with the group. Wrote the project description and fleshed out the project motivation to better reflect how it would be useful to students at UofM. Contributed to the related work section by researching and writing about random forests, as well as how we plan on using them. Brainstormed alternate data collection method to collect small random samples from a variety of institutions, not just UofM, to generate the overall dataset. Found additional supporting references for the methods described in related work.

Manheng Wang: Contributed to the investigation of RMP webscrapping options including beautifulsoup and RMP's internal API. Summarized and presented related work. Contributed to the exploratory analysis and visualization of the raw datasets. Led the design and production of the expo poster and coordinated the writeup of the checkpoints and the final report.

Casper Guo: Developed the overall vision of the project. Investigated the application of RMPAPI towards the project and contacted the project author to request relevant details. Conducted data cleaning and feature engineering to turn the raw datasets into the cleaned datasets that are ready for experimenting. Brainstormed text encoding methodologies and machine learning models. Implemented the text encoders, trained the machine learning models, designed the evaluation procedures, and reported the results.

8 Code Source

The project implementation may be found on our public Github Repo.

References

Rate my professors. Accessed: 2023-03-15.

C.R. Belfield and P.M. Crosta. 2012. Predicting success in college: The importance of placement tests and high school transcripts. page 41.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20:273–297.

Alexander Genkin, David D. Lewis, and David Madigan. 2005. Sparse logistic regression for text categorization. pages 1–8.

Tin Kam Ho. 1995. Random decision forests. pages 1–5.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 412–418.

The pandas development team. 2020. pandas-dev/pandas: Pandas.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Leonard Richardson. 2007. Beautiful soup documentation. *April*.

Pooja Saigal and Vaibhav Khanna. 2020. Multi-category news classification using support vector machine based classifiers. *SN Applied Sciences*, 2(3):458.

Yanshan Wang, Sunghwan Sohn, Sijia Liu, Feichen Shen, Liwei Wang, Elizabeth J. Atkinson, Shreyasee Amin, and Hongfang Liu. 2019. A clinical text classification paradigm using weak supervision and deep representation. pages 1–13.

Michael L. Waskom. 2021. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.

Nobel Zhou, Christopher Bryan, and Oze. 2022. Ratemyprofessorapi. `https://github.com/Nobelz/RateMyProfessorAPI`.