PSTAT 135 Final Project

Jeopardy Category Classification

By:
Micheal Alexander
Nick O'Dea
Sriharsha Addepalli

Abstract

This report examines the process our group used in predicting a Jeopardy category given the content of a question. We first explain the general features of the dataset when we first imported it and the steps we took to prepare the data for model fitting. Next, we discuss the different models we constructed: Multinomial Logistic Regression and Naive Bayes with computer-defined labels and human-defined labels determined by the category of a question. Lastly, we explain our choice of the champion model and opportunities for future research, as well as potential uses for classification algorithms outside the scope of our project.

Data and Methods

1. Data Import

The dataset we used for our project was found on Reddit[1], where a user crawled Jeopardy questions, answers, categories, and other metadata from www.j-archive.com, a website with all jeopardy questions that have been on air. When the data were first imported, there were 216,930 observations with 7 attributes: air date, round, value, question, answer, category, and show_number. However, since our goal was to predict a question's category based on the words in the question, the only attributes we used for our models were the question and the category.

2. Exploratory Data Analysis and Data Preprocessing

Before we started modeling, we needed to become more familiar with our dataset. Some exploratory analysis yielded interesting results. There were 27,995 distinct categories, with the most populated category having 520 observations and the least populated categories having just one observation. Because of the unbalance in our dataset, we decided to trim it to increase the ratio of categories and questions for more accurate modeling and shorter run times.

To increase the ratio between distinct categories and the number of observations within each category, we decided to include only the categories and their corresponding questions where the categories have 100 or more observations. We chose this number because it gave us a good distribution of counts of observations within the categories and made it easily interpretable when we later went on to creating a new set of human-defined labels. This resulted in 145 categories with a range from 100 to 520 observations in each category, as we can see in figure 1.1.

Figure 1.2 shows us a closer look at the top 20 categories from figure 1.1, where we expect to see some common categories such as *Science*, *Literature*, *History*, *Sports*, etc.

[1] https://www.reddit.com/r/datasets/comments/1uyd0t/200000_jeopardy_questions_in_a_json_file/

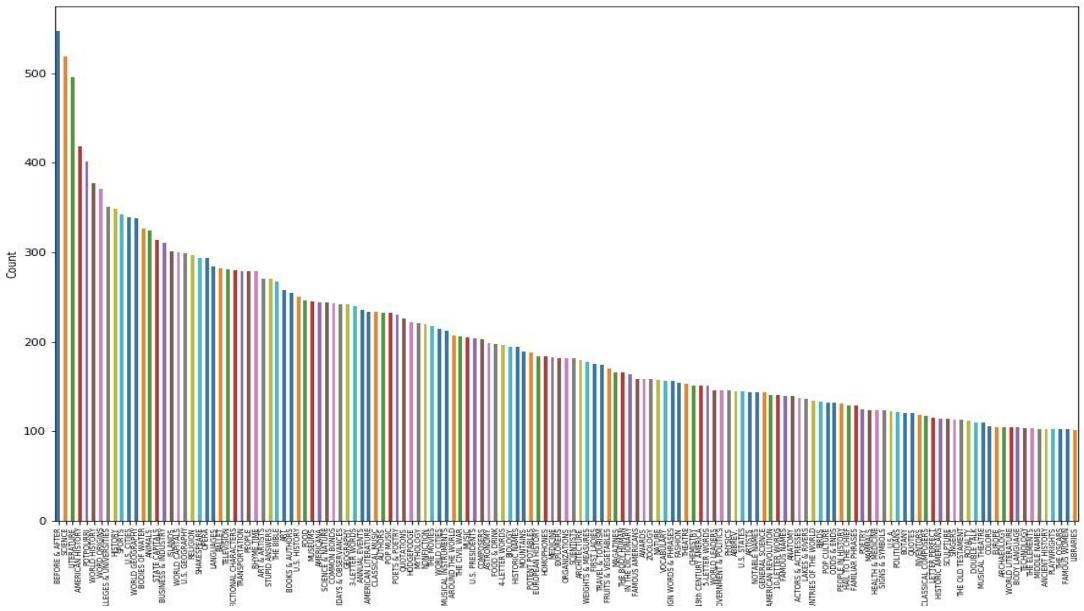

Figure 1.1

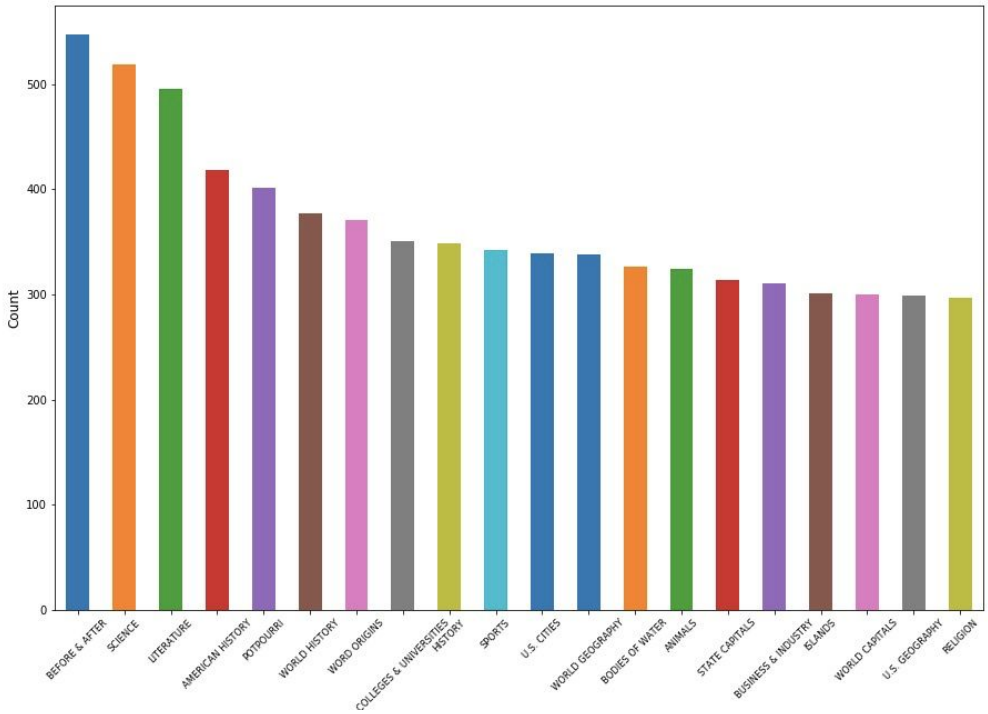

Figure 1.2

We then prepared this modified dataset using several steps. We used the questions as our predictors, so we needed a way to process the text of the questions. We did so by initially taking the questions and removing all the punctuation within them. We then tokenized the remaining

words and removed any stop words (i.e. “the”, “a”, “for”) as they do not add any meaningful information. We then stemmed[2] the words (i.e. “years” becomes “year”, “theory” becomes “theori”) so any tense of a word would be categorized as the same word.

3. Aggregating Labels

We found that many of the Jeopardy categories were rather similar. For example, there exist categories like *Literature*, *World Literature*, *Poetry*, *Poets and Poetry*, *Shakespeare*, and *Books + Authors* that could all reasonably be put under some larger *Literature* category. The similarity between categories will make correctly predicting such categories particularly difficult, as the observations are very similar between such categories. Thus if we could aggregate observations or categories together to yield a smaller number of more reasonable categories, we could have better predictive success without trimming our number of observations further. We felt that potentially losing the ability to differentiate between e.g. *Literature* and *World Literature* was a small price to pay.

We also found that we could run Naive Bayes on our 145 largest categories, but that Logistic Regression strained our computational resources to the limit. Given that predicting one of a large number of labels can be computationally challenging, we realized that reducing the number of categories could allow us to run and compare more machine learning algorithms.

Since we did not wish to remove categories’ worth of observation, we decided to investigate aggregating observations or categories to improve our predictions and allow us to compare previously prohibitive machine-learning techniques.

4. Computer-Defined Labels

We first used Bisecting K-Means to aggregate observations together into ten computer-defined labels. We knew we wanted to develop human-defined labels as a comparison to these computer-defined labels, and we knew that it would become increasingly difficult for us to sort observations into more and more categories, so 50 labels at the end would be prohibitively difficult for us to sort by hand. In addition, looking at the dataset we could see about ten natural labels based off the category names. Thus we chose k=10 computer-defined labels for ready comparison to our human-defined labels discussed below and because from data exploration that appeared to be a reasonable number.

We also found that Bisecting K-Means outperformed the regular K-Means algorithm in terms of runtime, likely because the number of observations requiring distance calculations shrinks as the clusters are bisected. Since the Bisecting K-Means algorithm gave us reasonable word clouds (below), we settled on that algorithm for generating our computer-defined labels.

---

[2] https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3923635548890252/1357850364289680/4930913221861820/latest.html

We used Python word clouds[3] to interpret our computer-defined labels, as such labels do not have intrinsic meaning to them. Below in figures 4.1 and 4.2 are two representative word clouds. The first, featuring words like “wrote”, “book”, and “critic” appears to be about literature and authors, while the second, featuring words like “state”, “capital”, and “name” appears to be about geography and maps.

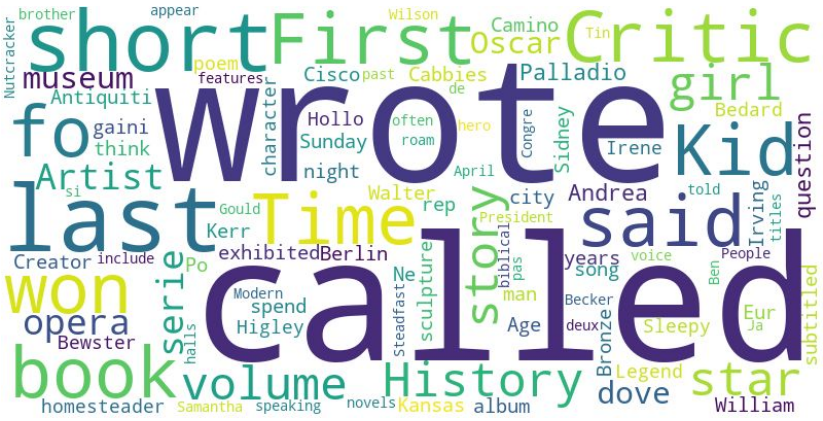

Figure 4.1

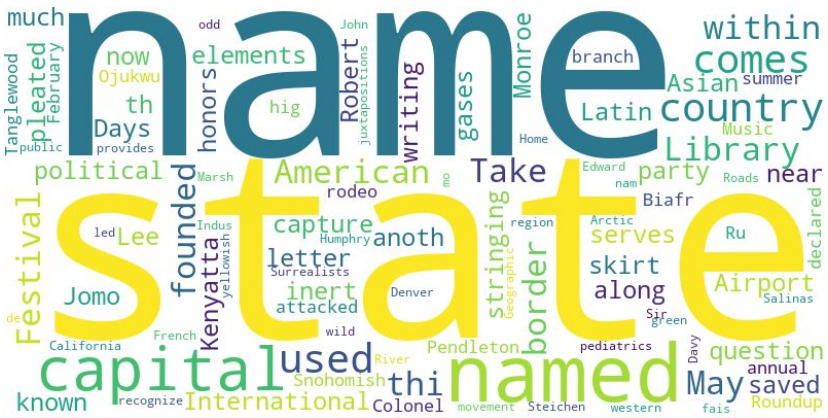

Figure 4.2

Though we can approximate the meaning of the computer-defined labels by looking at word clouds, we felt that lacking intrinsic meaning was a shortcoming of the computer-defined labels.

In addition, we recognized that attempting to predict the computer-defined labels would give us an artificially high accuracy. That’s because we generated the labels by aggregating observations based on the words within them; that is, *we generated our labels from the*

[3] https://www.datacamp.com/community/tutorials/wordcloud-python

predictors, so the predictors would have nigh-perfect predictive power. We discuss this phenomenon in results section 2 with Logistic Regression and Naive Bayes analyses.

5. Human Defined Labels

As a counterpart to the computer-defined labels created by clustering observations together, we decided to create human-defined labels created by aggregating categories together by their names. These labels, like *Literature* and *Water Geography*, would have intrinsic meaning. Moreover, we would not look at the question contents; that is, we would *generate our labels without looking at the predictors*. Thus we would avoid the artificially high accuracy and uncertain meaning affecting our computer-defined labels.

We decided on the labels by first individually reading through the 145 categories and creating ten labels under which we sorted the categories. We then shared our labels and came to a consensus with the ten labels *Science*, *Life and Food*, *Human Geography*, *Water Geography*, *People*, *Literature*, *Music/Art*, *History*, *Entertainment*, and *Wordplay*. Some of our choices for categories were strongly influenced by the frequent topics on Jeopardy. For example, *Wordplay* served to hold punning categories like *Before and After* which merge two answers in one. Similar to the computer-defined labels, we were able to run Logistic Regression in addition to Naive Bayes on this smaller set of labels which we discuss under results part 3.

Results

1. Naive Bayes Classification on 145 Categories

We decided to run a Naive Bayes model on all 145 categories of our modified dataset as a benchmark. We began with a 80% training and 20% test split. We then ran a pipeline that went through Hashing TF with the number of features set at 50,000 features, IDF, Indexing to assign numeric labels to the categories, and the Naive Bayes algorithm where we tested 5 smoothing parameters (0.1, 1.0, 5.0, 10.0, 50.0). We also did five-fold cross validation where we ended up getting that the best smoothing parameter was 10.0. When it comes to sensitivity, all parameters of smoothing gave us a range from 0.242 to 0.265 for the average metrics of the training set. This shows us that the sensitivity is low for this model and thus our model is robust. For this model, we ended up getting an accuracy of 0.2934 and an f1 score of 0.2573. We were surprised by this high accuracy given that many labels are similar in type such as *Science* and *Science & Nature*. We were able to improve this metric when we ran models on the computer-defined labels and human-defined labels.

2. Multinomial Logistic Regression and Naive Bayes Classification on Computer-Defined Labels

By reducing the number of labels to just ten by using clustering, we were now able to run Multinomial Logistic Regression in addition to Naive Bayes on these labels. Prior to shrinking

the number of labels, Multinomil Logistic Regression failed to run so this was a major step forward. Similarly to results section 1, we made two pipelines that began with Hashing TF, IDF, and Indexing. The pipelines and then ended with respectively Multinomial Logistic Regression and Naive Bayes.

For Multinomial Logistic Regression, we tried smoothing parameters of (0.001, 0.0025, 0.005, 0.0075, .01) before settling on .005 via cross-validation.

For Naive Bayes, we found that only extremely large values of the smoothing parameter yielded the best results, with many of our first sets of smoothing parameters consistently choosing the largest parameter via cross-validation. Our final net of smoothing parameters was (5000, 20000, 30000, 40000, 50000) with 30000 chosen via cross-validation. We viewed this complete change in the best hyperparameter as a sign that it didn't make sense to run prediction algorithms on our computer-defined labels. That is, since we generated our labels using our predictions, our predictors had perfect predictive power and led our models to have edge-case behavior.

As anticipated, our accuracy was inflated by generating labels based on the predictors. We found that Naive Bayes had an accuracy of 84% and Logistic Regression had an accuracy of 91%. Given the difficulty in interpreting both the computer clusters and the predictive power of the algorithms, we stopped here with the computer clusters and decided to focus our energies on the human-defined clusters which enjoyed intrinsic meaning and non-pathological dependence of the category labels on the predictors.

3. Multinomial Logistic Regression and Naive Bayes Classification on Human-Defined Labels

We decided to run a Logistic Regression model on the human-defined labels. We did so by first doing a 80% training and 20% test split. We then ran a pipeline that went through Hashing TF, IDF, Indexing, and the Multinomial Logistic Regression algorithm. We used multiple regularization parameters (0.001, 0.0025, 0.005, 0.0075, .01) and through five fold cross-validation, we found that 0.0075 gave us the best results. Additionally, average metrics for all five parameters of smoothing were all close to one other, so sensitivity in this model to hyperparameter choice is low. We obtained a test set accuracy of 0.4951 and f1 score of 0.4962.

We also ran a Naive Bayes model on the human-defined labels. We did so by first doing a 80% training and 20% test split. We then ran a pipeline that went through Hashing TF, IDF, Indexing, and the Naive Bayes algorithm. We ran the Naive Bayes model with smoothing parameters: 0.1, 1.0, 5.0, 10.0, 50.0, 100, and found that using five fold cross-validation that smoothing at 50.0 provided us the best results. Our metrics did not change much given these smoothing parameters, which means that there is low sensitivity to hyperparameter choice. We obtained a test set accuracy of 0.4956 and an f1 score of 0.4828.

We found that both models result in accuracy and f1 scores that are much higher than what we found in the Naive Bayes model on all 145 categories, showing that the aggregating of

labels using human judges was effective. Additionally, we found that the metrics were almost similar in both the Multinomial Logistic Regression and Naive Bayes models for human-defined labels.

Conclusion

To conclude, the goal of our project was to test whether we could accurately predict a category of a Jeopardy question based on the words of that question. After preprocessing the data to only include the most informative categories, we thought about how the computer would cluster the data and deduced that the computer-defined clusters would lack intrinsic meaning. To address this, we aggregated extremely similar Jeopardy categories into more broad categories without looking at the predictors. We called these our human-defined clusters, and would compare these to the computer clusters which were obtained using Bisecting K-Means.

For our first benchmark model, we used Naive Bayes on all 145 of our categories without computer or human labels. Next, we fitted a Multinomial Logistic Regression and a Naive Bayes model with computer-defined clusters. The final models we looked at were again Logistic Regression and Naive Bayes, but now on our human-defined labels. The accuracy rate of these models was lower than the computer-refined labels, but allowed us to generate more meaningful clusters. We decided that our champion model is the Naive Bayes model on the human-defined labels because that did provide us a high accuracy which was similar to the Logistic Regression model on the human-defined labels, but it had a much shorter run time than the Logistic Regression model.

If we had more time, we would consider adding predictors to ours models, such as the value of the question or the round of jeopardy that the question was on. We would also explore different methods to deal with the issue of sparsity in our dataset. Additionally, we would attempt to find a less sparse dataset with more observations per category to give us more data to construct our models with.

Our project was confined to the context of Jeopardy classification, but category prediction based on text has many uses, such as creating a database of questions and answers, or classifying comments of products into separate groups. With this project, we were able to gain a first hand look at the many possible uses of classification and clustering and will be able to apply this knowledge to future work.

References

https://www.reddit.com/r/datasets/comments/1uyd0t/200000_jeopardy_questions_in_a_json_file/

https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3923635548890252/1357850364289680/4930913221861820/latest.html
https://www.datacamp.com/community/tutorials/wordcloud-python