# What Kind of Joke?: Classification and Categorization of Short Text

* COMP331 Final Project

Amy Lee
*dept. of History*
*Occidental College*
Los Angeles, United States
alee5@oxy.edu

Olivia Baldwin
*dept. of Computer Science*
*Occidental College*
Los Angeles, United States
obaldwingeil@oxy.edu

***Abstract*—Humor is arbitrary and idiosyncratic as it appeals both to general audiences but differs on an individual bases. Jokes can be considered a condensed and compact version of humor. In this paper, we compare different models of text classification of jokes, all based off the same data set of jokes. Compiling the categories of both data sets, all jokes were labelled with one of the 21 categories. Preprocessing and implementing K fold has shown to increase both the precision and recall of our models. Furthermore, we include a joke recommendation system, where if the user inputs their own joke, one of similar content will be printed out. We discuss the implications of our research in sentiment analysis.**

## I. Introduction

Sentences, depending on the word choice, will create different effects and impressions. Causing an intended reaction can be particularly difficult when it comes to humor. Humor contains added nuances, some which become obsolete when delivery of comedy is limited to only written text. For example, without being back to rely on timing or conduct, a greater emphasis is placed on meaning. Much work has been done on computational humor towards distinguishing the difference between a joke and non-joke. However, not so much work has been done in analyzing in comparing one joke to another despite the wide variety of humor.

Originally, our project was intended to analyze and compare the components of short stories. By taking the elements of well-known and popular short stories, we had hoped to ultimately reproduce our own short story by dividing each file in our dataset into plot sections (introduction, middle, climax, and conclusion) and train models for each section. Our focus was to be on both the structure of the story as well as the contents.

However, we found research heavily revolves around Artificial Intelligence, such as TALE-SPIN, to help create plots and characters rather than Natural Language Processing (Meehan). For the sake of scope and feasibility, we shifted our attention from short stories to jokes. The short length of jokes as well as the vast variety in our dataset has given us enough data to analyze the differences between categories of jokes and the features of each category. We further limited our scope from analyzing the structure to the content of the jokes, because our dataset was labelled according to what it pertained to rather than the type of jokes. Due to the immense flexibility and diversity of both content and structure, we found it more prudent to focus on one of the two aspects.

Our paper is on the text classification of jokes. Given their content, how well can a model determine what kind of joke it is given the categories? Additionally, which model performs the best? We will be comparing the performance of a Naive Bayes model, Logistic Regression model, a Random Tree Classifier model. On top of answering these questions, with our text classification system, we were able to create a "Joke Recommender." Given an inputted joke, our model first labels what category it most likely belongs to and then returns a similar joke from our dataset.

The dataset was downloaded from a github repository by user "taivop." It contains three different files from three different sources. Each joke object in each file has a `body` field with additional fields that vary depending on the dataset. The `reddit_jokes.json` file was scraped from /r/jokes and includes all the submission as of February, 13, 2017. The `stupidstuff.json` file was scraped from stupidstuff.org and the `wocka.json` was scraped from wocka.com. Out of the three files, we use the `stupidstuff.json` and the `wocka.json` files because these files had the `category` field, unlike the textttreddit_jokes.json file. We compiled and simplified the combined categories and reassigned their `id` to match the revised categories. In total, we have 20 categories with a (category_code) attached to them

In this work, our three models performed differently, with Logistic Regression performing the best and Naive Bayes performing the worst. All models did improve with the implementation of K Folding. While the general trends of Precision and Recall of each category were similar across all three text classifiers, Logistic Regression has the best F1-score.

In the remainder of the paper, we first discuss related work,

. A special thanks to Dr. Celia Chen of the Computer Science of Occidental College, without whom this project would not have been possible.

outline our approach and results, and analyze our results. We conclude with discussion and future work.

## II. Related Work

To create a humor-processing program, it would need to have knowledge of the world and reasoning abilities, and for this reason (Ritchie, 2001.) For this reason, research on understanding humor has concentrated more towards using features to recognize jokes from nonjokes.

Cai and Ehrhardt explain, in their work, how using learned semantic word vectors in a neural network can create a humor metric that can determine whether a text is a joke or not, specifically if their model could differentiate between a pun and a sentence. They use the Brown corpus and POS tagger from python's nltk library to model a word's contextual information. Cai and Ehrhardt use a Hidden Neural Network and train their model using data collected from twitter. While their model performed at a reasonable level with a precision score of .650, a recall score of .822, and a F1-score of .726, they do conclude saying it is difficult to make progress due to the nature of hidden neural network.

Likewise, Jing, Talekar, and Rayz were interested if they could calculate how much a joke is a joke by calculating joke text similarity using doc2vec. They conclude that while they can test for text similarity, there is very little correlation between the cosine similarity and the knowledge resource-annotated joke similarity. The KR-annotated joke similarity was obtained from the General Theory of Verbal Humour, and rather than simply taking into consideration the similarity of the words themselves, the KR-annotated joke similarity takes into account six different features: Script Overlap/Oppositeness, Logical Mechanism, Situation, Target, Narrative Strategy, and Language. For this paper, however, we will be comparing jokes to other jokes. Instead of trying to estimate the percentage a text is to a joke, we seek to identify what kind of joke it is. We believe not enough work has been done in classifying jokes themselves before rules and features can be made to solve a larger problem: jokes vs nonjokes.

Text classification is a common topic in Natural Language Processing. One of the most famous of these models being Naive Bayes. Naive Bayes uses "bag-of-words" and is useful in discovering basic features quickly. Ngyuen, Smith, and Rosé present a model that predicts age based on language use. They gathered data from blogs, transcribed telephone speech, and posts from an online forum on breast cancer. Features of older people included words such as husband, retired, and work; while features of younger people were fun, school, and definitely (Ngyuen et al, 2011). By using Naive Bayes, we are able to have a baseline of performance for our data.

However, due to its low performance, Logistic Regression is a popular method of text classification. In Celine, Dominic, and Devi's research, they were able to predict the employability of an applicant by training their model on aptitude, communication, technical, and personality. If the calculated average is greater than the set threshold, the candidate is then assigned as "Employable."

Unlike Celine et al, we have twenty target classes and are limiting our factor to one variable, the `body`, to predict the `category`. The nature of our data allows us to compare naive bayes and logistic regression. We will also compare using a Random Forest Classifier. Random Forest has been shown to perform with good results; Liparas et al having trained using Random Forest using N-gram textual features to categorize articles and it performed with .844 accuracy.

Each model has benefits that suit our purpose, but because of the repercussions of each model, we will be comparing them. Each text classification model informs us about the data in a different light that can help further work on computational humor and text classification.

## III. Methodology

We first read the data of `wocka.json` and `stupidstuff.json` into a pandas Dataframe, with columns `body` and `category`. We dropped the fields `title` and `rating`. because the `stupidstuff.json` data has jokes that are categorized into over 40 different categories, we dropped all of the categories that were not included in the `wocka.json` data. Then, we added the two Dataframes together. Furthermore, we deleted the "Miscellaneous" category as it offset data distribution because it contained a sample much larger than any other category and it did not provide any meaning in its classification. Then we simplified the `category` by creating a dictionary that maps each category code to a numerical value and saved these values as a new column, `category_code`.

| | body | category | category_code |
|---|---|---|---|
| 0 | What do you call a cow with no legs?\r\n\r\nGr... | Animal | 0 |
| 1 | What do you call a cow jumping over a barbed w... | Animal | 0 |
| 2 | So, this guy walks into a bar.\r\n\r\nAnd says... | Bar | 2 |
| 3 | If the opposite of pro is con, isn't the oppos... | One Liners | 14 |
| 4 | I went to a wedding the other day. Two antenn... | Puns | 15 |

Fig. 1. A snapshot of our Dataframe

Once the full Dataframe had been assembled, we sent in the `body` column, which contains the raw text of each joke to be cleaned. The data is cleaned through the following process: tokenization → lowercase → remove punctuation → remove non-alphabetical characters → remove stop words → stemming → lemmatization → join back to string.

Once the data has been cleaned, the `body` and `category_code` are split into training (**X_train**, **y_train**) and testing sets (**X_test**, **y_test**) using the *sklearn train_test_split* method and a test size of 0.15. This means that 85 percent of the data will be used for training and 15 percent will be reserved for testing.

The **X_train** and **X_test** are what we take to train our term frequency-inverse document frequency model. Term Frequency counts the number of occurrence of each word. However, each transcript has a different length and it may not necessarily be true that document A, where word X occurs one time, is less relevant than document B, where word X occurs ten times, because the numbers are not normalized. So, we multiply the number of occurrence with the inverse document

frequency and doing this will also lessen the weight of words more frequently used. Inverse document frequency looks at how many documents also have the same word. The TF-IDF value of each word is the product of the term frequency and the inverse document frequency. In short, for a word to have high TF-IDf in a document, it must appear a lot of times in said document and must be absent in the other documents. This is useful when reporting which words weigh more given a certain point category.

- 0 : Animal
- 1 : Work
- 2 : Bar
- 3 : Blonde
- 4 : Children
- 5 : College
- 6 : Gross
- 7 : Insults
- 8 : Knock Knock
- 9 : Lawyer
- 10 : Lightbulb
- 11 : Medical
- 12 : Men/Women
- 13 : News/Politics
- 14 : One-liners
- 15 : Puns
- 16 : Redneck
- 17 : Religious
- 18 : Sports
- 19 : Technology
- 20 : Yo Mama

The TF-IDF model vectorizes each of the joke descriptions in **X_train** and **X_test** so that they can be entered into our different text classification models for training and testing. Using the TF-IDF, we are able to determine which category the input is most closely related to using logistic regression. The model predicts which category the joke is part of. Within the given category, we use cosine similarity once more to compare the input with the jokes in our training set. We output the most relevant joke, and because our training set and testing set is randomly split each time the code runs, even if the user inputs the same joke, there is a chance a different joke will be outputted.

### A. Naive Bayes

The Naive Bayes Classifier is a probabilistic classifier that uses a 'bag of words' approach, that is, it assumes a feature independence and focuses only on word frequency. To classify a sentence as positive or negative, the classifier enacts Baye's Rule: For a document d and a class c,

$$P(c|d) = \frac{P(d|c) * P(c)}{P(d)}$$

The probability of a class, given a document is the probability of the document given the class multiplied by the probability of the class and all divided by the probability of the document. Because our data set has more than two possible classifications, we are training the *sklearn* Multinomial Naive Bayes Classifier, which is different from a normal Naive Bayes Classifier only in the fact that the class has more than a binary option. In the case of this paper, the classes are the different joke categories and the document is the text of each individual joke.

### B. Logistic Regression

For this data set we use the *sklearn* Logistic Regression Classifier to perform Multinomial Logistic Regression, which is the use of the logistic regression algorithm to predict more than two classes. The logistic regression algorithm is used to predict the probability of a certain class based on independent and dependent variables. The dependent variables are the classes and the independent variables are the features used to predict the class. The equation for logistic regression is as follows:

$$p = \frac{1}{1 + e^{-}b_0 + b_1 x} \tag{1}$$

In the case of Multinomial Logistic Regression, similarly to Multinomial Naive Bayes, the dependent variables are categorical rather than binary (they have more than two possible values), but other than that it is the same as regular logistic regression.

### C. Random Forest Classier

Random Forest is based on a group of decision trees. Each decision tree is built by determining split nodes, which are features that calculate and predict the class. Using one tree will result in overfitting, which is why we use multiple trees in random forest. Random forest uses two concepts: the sampling of training data is randomly select when building trees and the subset of features when splitting nodes is also randomly selected. Ultimately, The final prediction of the random forest is the average of all the predictions of each decision tree.

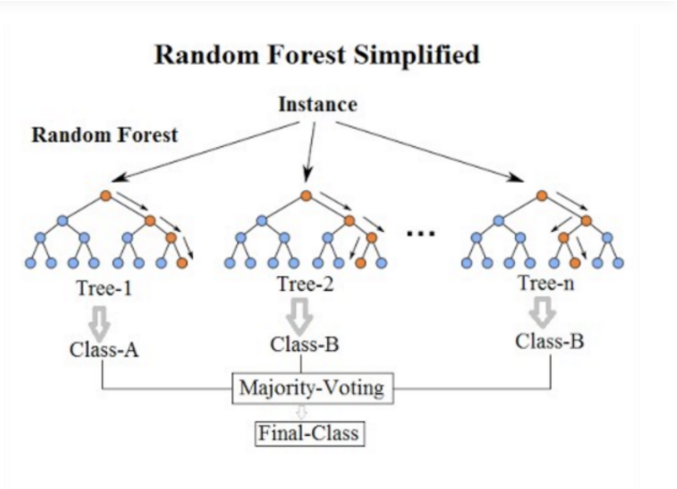


Fig. 2. Visualization of Random Forest from "Random Forest Simple Explanation" (Koehrsen, 2017)

Each of these models takes in **Features_train** (the vectorized **X_train** output) and **labels_train** (the vectorized **y_train** output) to train the model to identify each category. Then, the trained model is given **Features_test** (the vectorized **X_test** output) so that it can make predictions on the testing data that it has not seen before. The predictions that the model makes are then compared to the true classifications using the *sklearn classification_report* method and the precision, recall, and F-1 measures of each model for each category are shown.

## IV. Result and Analysis

This section will be analyzing and comparing the three different text classification models described in the earlier parts of our paper. Before we decided on which text classification model to use for our recommendation system, we used different methods to determine the best model. We will begin by discussing the features of each `category_code`, before presenting the performance measures of each model. Lastly, we will also be examine the performance after implementing K-fold cross-validation for each model.

### A. Features

By using TF-IDF, we can see the most correlated unigrams and bigrams of each `category_code`. The table below displays the unigrams and bigrams with the greatest weight of each `category` in their lemma forms, due to them being preprocessed for the sake of a simplifed vocabularly list, and shows us what kind of words are most closely related to what kind of joke.

| category | unigram | bigram |
|---|---|---|
| Animal | cat, eleph | cross road, deciph phrase |
| At Work | employe, bos | get work, bos ask |
| Bar | bar, batend | bartend say, walk bar |
| Blonde | brunett, blond | call blond, blond say |
| Children | teacher, johnni | littl boy, littl johnni |
| College | student, professor | ask student, look forward |
| Gross | gay, diarrhea | michael jackson, go bathroom |
| Insults | stupid, ugli | black man, black peopl |
| Knock-Knock | knockknock, knock | answer door, knock knock |
| Lawyer | clinet, lawyer | thousand dollar, lawyer said |
| Lightbulb | light, blub | take chang, light blub |
| Medical | patient, doctor | doctor say, doctor doctor |
| Men/Women | wife, husband | wife repli, wife say |
| News/Politics | presid, bush | presid bush, georg bush |
| One Liners | blonde, yo | last word, nt succeed |
| Puns | pirat, maker | know old, get littl |
| Redneck | might, rednekc | knowk redneck, might redneck |
| Religious | rabbi, priest | st peter, priest say |
| Sports | coach, golf | golf ball, play golf |
| Tech | viru, comput | oper system, tech support |
| Yo Mama | momma, yo | yo mama, yo momma |

Due to the nature of TFIDF, these words are not the ones most commonly use in each category, but the ones that uniquely appear the most in each category. We note that there are no outstanding abnormalities in our data, rather the unigrams and bigrams provide information for each `category`. For example, we can tell most jokes related to News/Politics are about President Bush and most Children jokes use the name "johnny." Judging by the TFIDF features of each category, using TFIDF for text classification is a acceptable method.

### B. Precision and Recall

In the following graphs, we compare the performance of each model; the red line representing Naive Bayes, the blue line representing Logistic Regression, and the black line representing Random Forest. Precision is the number of true positives over the number of true positive plus the number of false positives. This is the percentage of jokes categorized correctly out of all jokes given a category. The general trend of precision are parallel, such as high precision

for `category_code: 3` and a dip at `category_code: 4`.

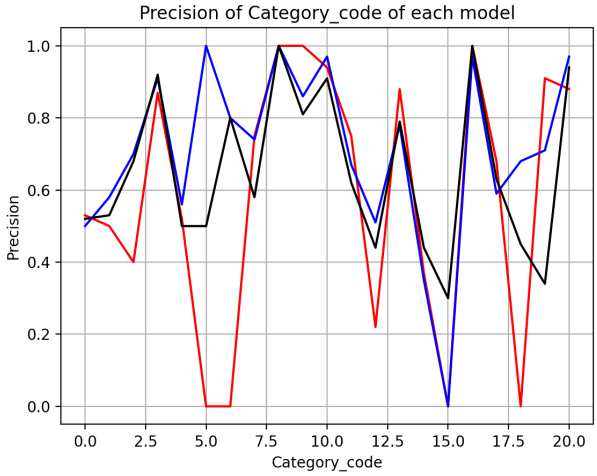


Fig. 3. Precision of Category_code of each model

Recall is the number of true positives over the number of true positives plus the number of false negatives. This is the percentage of jokes categorized correctly out of the actual number of these type of jokes. It will tell us the proportion of actual "Blonde" jokes out of all "Blonde" jokes classified. While logistic regression and random forest perform similarly, naive Baye's recall is less than that of the other two models.

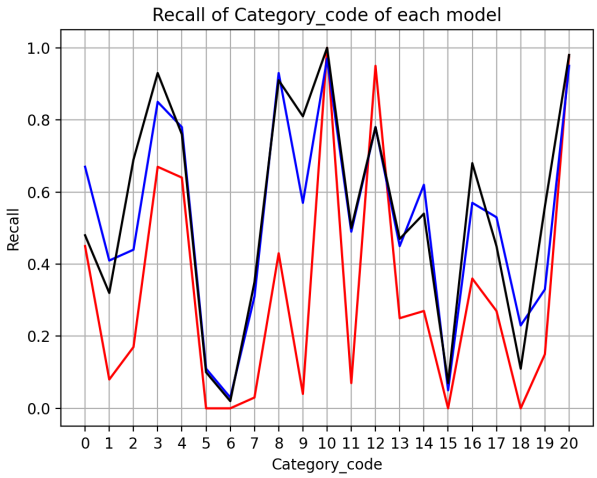


Fig. 4. Recall of Category_code of each model

### C. F1-Score

However, maximizing one metric will be at the expense of the other metric. Therefore, examining the F1-score, the harmonic mean of precision and recall, gives us a better understanding of performance. It gives equal weight to both measures. Logistic regression and random forest perform even more similarly when comparing F1-score than comparing precision or recall. There is one instance when naive Bayes does perform better than logistic regression and random forest at category_code 12, which is also the `categorycode` with the highest support.

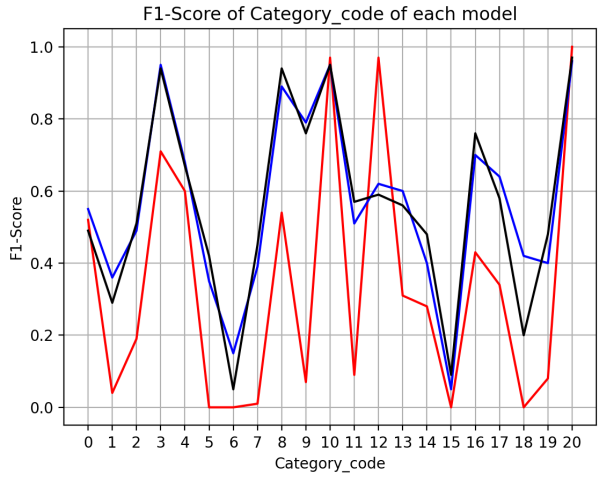


Fig. 5. F1-Score of Category_code of each model

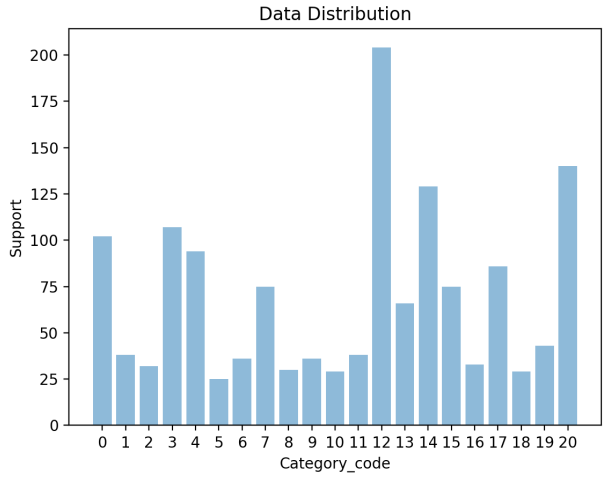


Fig. 6. Data Distribution by Category_code

While logistic regression and random forest perform the best, there are limitations for the model. Logistic regression works best for binary values, and is not entirely suitable to use to classify into 21 classes. Random forest can be too slow and ineffective for real-time predictions. While it trains quickly, it is slow when creating predictions. This is not a huge problem, since we do not have a huge dataset. So despite naive Bayes poorer performance, we included it to be used as a baseline.

### D. K-fold

After examining the data, we considered what ways we could improve the performance of our models. We considered if the performance could be improved by balancing our data,

due to the variance in data sample in category_code. However, there is no strong correlation between the data support and F1-score. Fig. 7, 8, and 9 display a scatter plot of data size vs. F1-score of naive Bayes, logistic regression, and random forest, in that order.

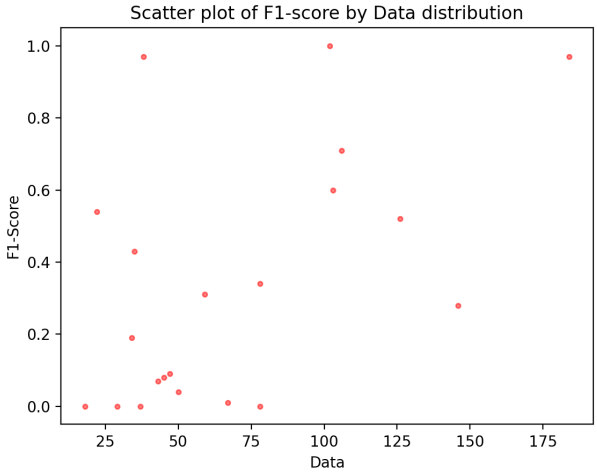


Fig. 7. Scatterplot of F1-score of Naive Bayes vs. Number of Data

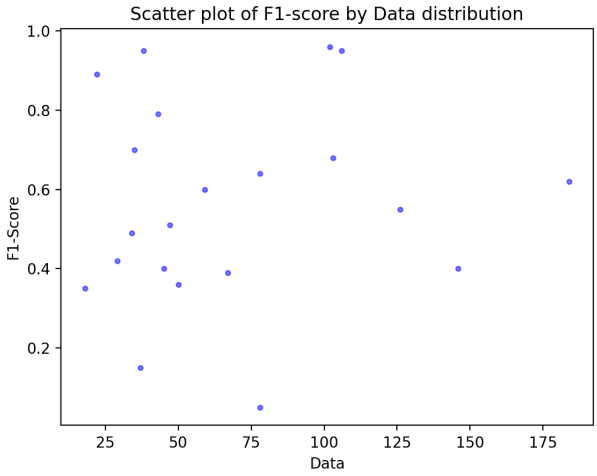


Fig. 8. Scatterplot of F1-score of Logistic Regression vs. Number of Data

Rather than the F1-score depending on the data size, we suspect the F1-score is dependent on the scope of each `category_code`. Despite seeming equal, the category_code vary in variety; for example, knock knock jokes will always contain the words ”knock knock” while jokes in the ”News/Politics” category could range from jokes about the president to ones about voter participation.

So in order to better the performance of the models, especially naive Bayes that has a F1-score less than 0.5, we decided to test the models using K-fold cross-validation. K-fold cross-validation estimates the skill of a model on unseen data. It

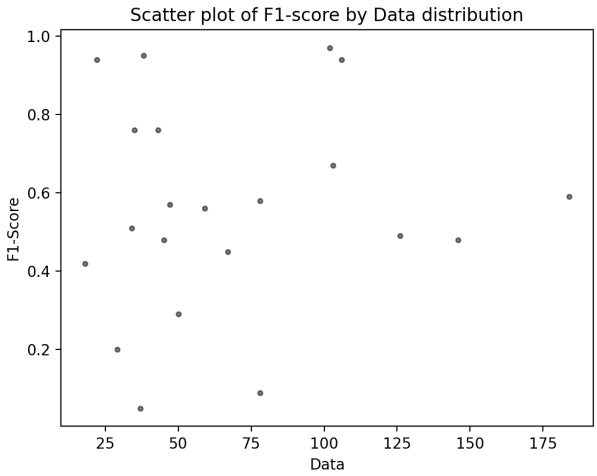


Fig. 9. Scatterplot of F1-score of Random Forests vs. Number of Data

randomly shuffles the dataset, splits the dataset into k groups, and within each group, split the group into training data and testing data. After fitting the model using the training data, all the evaluation scores are collected and used to summarize the performance. This technique prevents overfitting, especially with the data sets that have a smaller data sample. We initially chose a 10-fold because it is known to have low bias and modest variance.

Using a `test_size` of 0.15 and 10-fold, the naive Bayes model's mean performance is 0.43 with a standard deviation of 0.04, logistic regression's mean performance is 0.59 with a standard deviation of 0.08, and random forest's mean performance is 0.57 with a standard deviation of 0.06. This is depicted in the fig. 10.

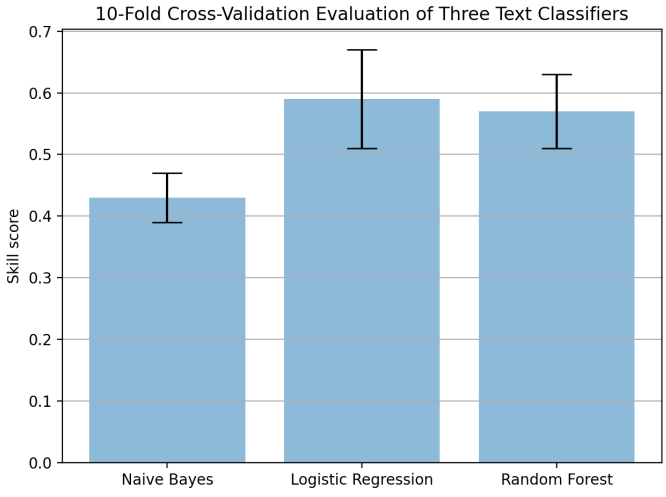


Fig. 10. Bar graph of each model using 10-fold cross-validation with error bars

When comparing our three models with 10-fold to a 5-fold, the performance decreases. The mean score of naive Bayes

is 0.42 with a standard deviation of 0.03, the mean score logistic regression is 0.58 with a standard deviation of 0.05, and the mean score of random forest is 0.56 with a standard deviation of 0.04. While the variance decreased, so did the general performance of the model.

## V. Threats to Validity

The greatest threat to the validity of our model is the data set and the categorical distribution of the data. As seen in Fig. 6. There is a sizable difference between the number of jokes in each category. This means that each category does not have an equal amount of data for the models to train on and create features. In some cases, this means that the model does not have enough data to create distinguishing, accurate features, and in other, this means that the category and its features are too generalized. As was stated in the Methodology section, we removed the "Miscellaneous" category from the data because it both offset data distribution and was not complementary to text classification due to its ambiguity. This decision improved the accuracy of the model, however, it deleted a large portion of the data set.

While doing research to locate a data set for this paper, we were unable to find any other large open source joke data sets that were labeled by category, so we were unable to fix this issue. In future work, we would hope for the creation of another data set like this one that illuminates the "Miscellaneous" category before the data is assembled and works to retrieve greater amounts of data for the other categories.

An equally vexing problem that cannot be overcome with more data is the broadness of certain categories, as discussed previously. The models are based on TFIDF, which weights unique words more in a given category. However, in more general and broad categories such as "Puns," there could be a lack of defining features to train our models on. More specified rules could result in a lower recall.

In our model, we also used the built-in functions of Scikit-learn, a machine learning library, to calculate the TF-IDF values, and to implement each of the models (Naive Bayes, Logistic Regression, and Random Forest). Scikit-learn, however, is one of the most popular machine learning libraries on GitHub, so we can safely assume the formulas used to calculate the TF-IDF values and to implement each of the models are correct.

## VI. Conclusion

In this paper, we proposed three different text classification models, Naive Bayes, Logistic Regression, and Random Forest to classify jokes into categories based on their textural context. Through the implementation of preprocessing the raw text, TF-IDF vectorization, and K-Fold cross validation, we have concluded that logistic regression, narrowly, has the best performance among these classifiers. We have also created a joke recommendation system that allows the user to enter a joke, and provides them with a new joke from our database that is similar to the one that they entered. Further work can be done to increase the performance of each of these models through the creation of a larger data set. Another facet worth exploring is the racial and gender bias within jokes. One interesting observation is the TFIDF bigram of Insults, black man and black peopl. This raises questions as to whether insulting jokes revolve around calling people "black," disclosing social stigmas and racist tendencies.

## References


[1] Celine, Dominic, and M. Savitha Devi. "Logistic Regression for Employability Prediction." *International Journal of Innovative Technology and Exploring Engineering* 9, no. 3 (January 2020): 2471-2478.

[2] Dong Nguyen, Noah A. Smith, and Carolyn P. Rosé. (2011). Author age prediction from text using linear regression. In Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH '11). Association for Computational Linguistics, USA, 115–123.

[3] Jing X., Talekar C., Taylor Rayz J. (2018) Comparing Jokes with NLP: How Far Can Joke Vectors Take Us?. In: Streitz N., Konomi S. (eds) Distributed, Ambient and Pervasive Interactions: Technologies and Contexts. DAPI 2018. Lecture Notes in Computer Science, vol 10922. Springer, Cham

[4] Liparas D., HaCohen-Kerner Y., Moumtzidou A., Vrochidis S., Kompatsiaris I. (2014) News Articles Classification Using Random Forests and Weighted Multimodal Features. In: Lamas D., Buitelaar P. (eds) Multidisciplinary Information Retrieval. IRFC 2014. Lecture Notes in Computer Science, vol 8849. Springer, Cham

[5] Koehrsen, Will. (2017). "Random Forest Simple Explanation." https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

[6] Meehan, J.R. (1976). The Metanovel: Writing Stories by Computer. Outstanding Dissertations in the Computer Sciences.

[7] Ritchie, G.: Current directions in computational humor. Artif. Intell. Rev. 16(2), 119-135 (2001)

[8] Pungas, Taivo. (2017). "A dataset of English plaintext jokes." GitHub repository. https://github.com/taivop/joke-dataset