The Ethics of Reddit and an Artificial Moral Compass

Presented to the S. Daniel Abraham Honors Program

in Partial Fulfillment of the

Requirements for Completion of the Program

Stern College for Women

Yeshiva University

May 6th, 2020

Ayliana Teitelbaum

Mentor: Professor Joshua Waxman, Computer Science

Introduction

There is an inherent lack of clarity to moral lines. This clouding of clear margins is the cause of constant ethical debates. People may disagree with their opponent, or they may not know themselves what the correct decision is. There is no black and white in morality; there are only shades of gray. Besides being of interest to the involved parties, these gray area situations are a source of fascination to many other people. In order to benefit both the conflicted and fascinated, a question and answer community to discuss these situations was created. This community allows people to post their questions and get multiple responses by various users giving their personal opinion on who, if anyone, is correct. These questions tend to have a lot of nuance and subtlety to them and can involve many different people, which human readers can understand and rule on using their moral compass. While humans have this moral compass to guide them, be it inherent or societally taught, machines do not. However, the same way machines can "learn" different skills, such as how to identify the subject in pictures or conduct a basic conversation based on previous data, they may be able to "learn" a set of ethics based on previous situations that have already been decided on by people. This paper will explain and analyze an attempt to use the moral question and answer community history to "teach" a computer a set of moral standards to use in judging new situations. Additionally, this paper will explore other analyses of this question and answer community to understand what ethical considerations and ideas are prioritized.

This question and answer community is found on Reddit, which is a discussion platform for various interests and questions. Reddit is based around various communities called "subreddits". Subreddits can be focused on various topics such as politics, headphone types, different TV shows, or funny stories. People who post on Reddit, called redditors, post

questions or statements relating to the subreddit's topic under that specific subreddit. Then, other redditors can comment on that specific submission. Comments can also be made on other comments, all focusing on the topic of that specific subreddit.

There is a subreddit which focuses on moral or ethical dilemmas. This subreddit is called "Am I the Asshole," or AITA. Redditors post situations that they have encountered and ask who was in the right and who was in the wrong. An example could be something along the lines of:

> "I (29, M) have 2 younger half siblings (21 and 18). We are biologically related through my mom, but my dad adopted me when I was very young. When one of my brothers was really young, he was kind of a jerk, so my mom said it wasn't a good idea to tell him I was adopted on my dad's side. Now he's older so I wanted to clear the air and have an open relationship with my brothers, but my parents still didn't want me to tell them. I went ahead and told them, and explained that my parents wanted me to lie to them. They understand why I kept it a secret, but they are furious at my parents for lying to them this whole time."

After the submission is posted, different redditors comment on the submission and the redditor who posted the submission responds to any questions asked in the comments.

In the comments the redditors give a judgement on who they think is correct. However, in many cases there is more than one party who is right or more than one who is wrong. Therefore, there are five different options for judgements: YTA, NTA, ESH, NAH, and INFO. The first is YTA, which stands for you're the A-hole, and the other person/people are not, which in this case would mean the writer was wrong. The second is NTA, which stands for not the A-hole and the other person/people are, which in this case would mean the writer did nothing wrong, but the parents did. The third is ESH, which stands for everybody sucks here, both the writer and the parents. The fourth is NAH, which stands for no A-holes here, conveying no one did anything wrong. The fifth is INFO which means the writer did not

provide enough information to make a judgement. The INFO ruling was not used as a judgement option in the classifier, as it doesn't convey an ethical opinion.

The different rulings given by commenters are used to assign a final ruling to the submission. Only "top-level" comments, which are comments directly on the submission and not on other comments, are eligible to be assigned as the final ruling. After the submission is posted, comments are posted and "upvoted" by other redditors. If a redditor upvotes a comment, it means they agree with the comment. 18 hours after the submission is posted, the top comment of the submission is selected as the ruling on the submission. The top comment is the comment with the most upvotes by the other redditors. The final ruling is assigned by the reddit bot, which takes the judgement in the top comment (YTA, NTA, ESH, NAH, INFO) and assigns it as the "flair" on the submission. A "flair" on a submission is a small banner or tag added to the top of the submission that lets readers know the final judgement on the submission.

Data Collection

There are two main application programming interfaces (APIs) used in gathering data from Reddit. An API is a set of pre-built tools that can be used without having to know the details of how it was implemented. The two main APIs used for reddit data collection are Pushshift and PRAW. Each of these APIs have benefits and drawbacks to them. PRAW stands for Python Reddit API wrapper. PRAW contains many different python methods that can be used to easily retrieve data based on the user's specifications. For example, it can return submissions or comments from a specific subreddit submitted during a certain time period. However, PRAW requires the user to have a two second delay between each call of at most

100 items. This means that in a minute, one can make about 30 calls of 100 items each, which is 3,000 items total per minute.

As opposed to the PRAW API, the Pushshift API was designed by redditors, and does not have a set of pre-built Python methods. It is accessed directly through the API endpoint, which is a URL that data requests are sent to. Similar to PRAW, it can be used to search for submissions and comments filtered by attributes such as subreddit, date and redditor. However, it is different from the PRAW API in the maximum requests per minute allowed. Through Pushshift, the user can make 200 requests per minute, with each request having 500 items. Therefore, using Pushshift one can get 10,000 items in a minute, which is more than three times the 3,000 item per minute limit using PRAW. However, Pushshift is not as user-friendly as the PRAW API because it is accessed directly through the API endpoint, and not through a wrapper. Therefore, to retrieve large amounts of data, a combination of Pushshift and PRAW was used to maximize the allowed requests per minute. If only small amounts of data were needed the PRAW API was used because it has a more user-friendly way of retrieving data.

There were two sets of data needed to generate the features to train and test the classifiers. The first of these sets was a list of the rulings of prolific redditors on the AITA subreddit. In order to determine who were prolific redditors, I first took a subset of the top submissions on the subreddit from the PRAW API and gathered a list of all redditors who commented on those submissions. This was based on the assumption that most redditors who frequently commented would have commented on the top submissions. Then, for each of those redditors, using the Pushshift API, I retrieved a list of their top- level comments in the AITA subreddit and the submissions that those top-level comments were on. I then made a list of the redditors and the number of submissions that they commented on, and sorted it. This gave me

a list of the most prolific redditors, along with the number of times they had commented throughout the subreddit. The redditors with the highest number of comments had commented on thousands of submissions.

The second set of data that needed to be gathered was a list of the submissions along with their top ruling, which is assigned as that submission's flair. Originally, I started by using Pushshift, because I wanted to retrieve all the submissions in the subreddit. However, when I tried to retrieve the "flair" on each submission, it was empty on a large amount of the submissions. This is because Pushshift ingests the data from Reddit at certain periods of time, and then doesn't update that data when it changes. Therefore, if the data on that particular submission was pulled by Pushshift into their database before the flair was assigned, the flair of that submission would not be in the data. To get around this issue, I pulled the submission ids and submission text from Pushshift and retrieved the flair for each submission using the PRAW API. This way I was able to leverage the fast retrieval of the submission text and id list from Pushshift while also getting the flair submissions of each post from PRAW using those ids. Additionally, by alternating the retrieval of data from each, there was a break between calls to each API, allowing for larger amounts of data to be retrieved without hitting the request limit. However, from the compiled list, not all the submissions were used. In order to not have a bias in the machine learning models, an equal number of submissions receiving each ruling was under sampled to use in the training and testing of the different models.

Feature Selection

After gathering the data from Reddit, features were extracted from the text. Features are quantifiable aspects of data that are fed into different machine learning techniques. These

features were used to produce machine learning models that would predict the rulings of different submissions based on aspects of the submissions. There were a few methods used to break down the text: bag-of-words, TF-IDF and Doc2Vec. Each of these methods have different uses and strengths to them, while sacrificing other aspects that are more represented in the other methods.

However, before the features could be produced, text processing had to be done to remove issues in the text of the submissions that would affect representation of the text. The first problem is that certain words that appear many times throughout a sentence, such as “a” or “the”, can confuse the representation of the text because these words don’t add to the meaning of the sentences. To solve this issue the text was filtered to get rid of any “stop words” which are common words in the English language that are often filtered out before doing any text processing.

After filtering out stop words, there was still more text processing that had to be done on the data. An additional problem within the text is that there are multiple versions of the same word within the English language. For example, the words “work”, “worked” and “working” all have the same basic meaning, but are viewed as three separate words. In order to make these words be viewed as the same word, lemmatization was used. The “lemma” of a word is the root dictionary meaning of the word, and lemmatization is the process of converting a word to its lemma. Thus, “working” would be lemmatized to “work”.

After text processing, the features of the text could then be selected. The first method of feature selection, bag-of-words, is a very simple way of quantifying a text based on the frequency of occurrence of different words throughout the text. In computer science, the term “bag” refers to a multiset, which allows duplicates but does not have order. To create the bag-

of-words, a single document, which in this case is a submission, is made into a count of words that appear in the document. The number of times each word appears in a single document is counted and stored. The document is then represented as a list of the number of times each word appears in that document. However, not all the words across submissions can be included as features. This is because there are many words in the English language, and therefore many words across submissions. If all of the words were included, there would be too many features, and the models would take a very long time to run. Additionally, there would be many words with only a handful of occurrences, which would not really affect the model produced. Therefore, a certain amount of words with the top frequency were selected as features. Because of the simplicity of this method for feature selection, it has many problems. One of the shortcomings of this method is that the method does not maintain or represent the original order of the text. To somewhat represent this order, groups of words, called ngrams, were added to the bag-of-words in addition to the individual words. An additional shortcoming is that the frequency of word occurrence is used as a measure of importance, which can lead to certain words used very often in the English language being used as features, even if they are not necessarily indicative of a specific ruling or content of a submission.

The next method used was TF-IDF. This method is used to try and get around some of the shortcomings of the simple bag-of-words model. As stated, one of the problems with the bag-of-words model is that frequently appearing English words which do not add to the meaning of the document, but are not necessarily classified as stop words, can end up being included as features. TF-IDF attempts to address this problem. TF stands for term frequency, and is the number of times a term appears in a text, which is the way the bag-of-words method approaches the analysis. However, there is also an IDF aspect, which stands for inverse

document frequency. Document frequency is the number of documents that have the specific term in it. Inverse document frequency is the log of the number of documents in the corpus divided by the document frequency of that particular term. TF-IDF for a particular word is the term frequency for that particular word multiplied by the inverse document frequency for that word (Havrlant & Kreinovich, 2017). The TF-IDF of each word within the corpus is computed and is a measure of the significance of that particular word within the corpus.

The main idea of this approach is that if a term appears throughout many documents in the corpus, then it is probably not a word that is related to a specific ruling. Rather, it is a frequently used term in the English language that has no effect on the determination. On the other hand, if the term only appears in a few documents, it is more likely to be a word that has an effect on the ruling of the particular submission, as it is a more unique word. The number of appearances of the word in a document is also indicative of the importance of that particular word. The more times a word appears in a document, the more relevance that term has to the document. By combining these two aspects, word frequency and inverse document frequency, a list of relevant words in the corpus can be created.

This can be illustrated by a toy example. For example, let's say a particular corpus has ten documents, and the word "house" appears in two of them ten times each, while the word "said" appears in eight of them five times each. The TF for "house" is 20, while the TF for "said" is 40. The IDF for "house" is ln(10/2)=1.61, which the IDF for "said" is ln(10/8) = .22. Therefore, the TF-IDF of "house" will be 20*1.61= 32.2, while the TF-IDF of "said" will be 40*.22= 8.8. Therefore, even though "said" appears many more times in the corpus than the word "house", "house" has a higher TDF-IDF because it appears many times within a few documents, rather than many times across the whole corpus. After assigning TF-IDF scores to

the terms in the corpus of submissions, the term frequencies of the terms with the highest TF-IDF scores were used as features to then feed into the machine learning algorithms.

One of the problems with both bag-of-words and TF-IDF is that neither of them account for all of the context within a submission. They are taking each word as a distinct feature, and not taking into account the similarities between different words, or the words surrounding each of the words. Doc2Vec is an approach that tries to solve this problem, by turning documents into vectors that are representative of the document as a whole. It is based on earlier work, called Word2Vec, that created a method to convert words into vectors through analyzing surrounding words. In Tomas Mikolov et al. (2013) they explained Word2Vec as being similar to a feedforward neural net language model, or NNLM. A NNLM is a neural network used to predict the next word in a document based on the previous words. As a very basic overview, the NNLM converts each word into a feature vector, and creates a probability function, which can be a separate neural network. The probability function takes the feature vector for previous words and outputs a vector that in position i contains the probability that the word w_i is the next word given the previous words. The network then goes through training to change the parameters associated with the feature vector and the probability function to minimize error, along the way also creating a very complex hidden layer as part of the prediction method (Bengio et al., 2000). Word2Vec does not include this hidden layer, and also does not care about the final model result that will do the prediction. Rather, it cares about the feature vector for each word, because the assumption is that two words that are similar in meaning will have similar words around them, and therefore will have similar feature vectors. Therefore, the feature vectors represent the semantic meaning for a word, taking into account its context (Mikolov et al., 2013).

The Doc2Vec method expanded on Word2Vec and was first proposed by Quoc Le and Tomas Milolov in 2004. Instead of producing only feature vectors for words, it also produces a feature vector for a specific paragraph, or document, and then either concatenates the two vectors or averages them to produce a guess for the next word. At the end of the training of the neural network, besides the neural network there is a matrix containing the feature vectors for each word. This matrix is shared between the various paragraphs. In addition, there is also a paragraph vector for each paragraph. When a new paragraph is fed into the model, the paragraph feature vector for that paragraph is generated by the neural network. The neural network does this by choosing the parameters for the paragraph feature vector that, in conjunction with the word vector matrix, will minimize the error in predicting the next word in the paragraph. The paragraph feature vector can then be used as input to different classifiers (Le & Mikolov, 2014).

Classifiers

After using bag-of-words, TF-IDF and Doc2Vec to select features from the submissions, the features were then used by different classifiers to try and predict the rulings for new submissions. Each set of features generated was used to train each of the classifiers and then the classifiers were tested on a set of new submissions. The classifiers used were Logistic Regression, Multinomial Naive Bayes, K-Nearest Neighbors, Linear Support Vector Classifier and Random Forest.

The first classifier used was multivariable Logistic Regression. Binary Logistic Regression, Logistic Regression with only two classes, is a classifier that is similar to Linear Regression. The main difference between them is that while linear regression produces a

continuous spectrum of results without bounds, Logistic Regression outputs a result between zero and one representing the probability of the input belonging to a specific class. To make this model produce values between zero and one, a linear function is generated representing the log odds of the probability of an input belonging to one of the two classes, based on its features. Each feature has a parameter, a number which is multiplied by the value of the feature in the equation and which indicates how that feature affects the final result, produced by the maximum likelihood method. The value outputted by the linear equation is used as input into the sigmoid function, which produces a value between zero and one (DeMaris, 2003).

However, while this technique works for problems where there are only two possibilities, it does not work for situations where there are multiple possible classes. For those cases, there are multiple variations on Logistic Regression, one of which is a one-vs-rest, or ovr. In this technique multiple equations are produced, each computing the probability of the input being one particular class and not any of the others. For example, in a case where there are three classes, A, B and C, one equation will calculate the likelihood of the input being class A and not class B or C, the second equation will output the likelihood of the input being class B and not A or C, and the third will produce the likelihood of the input being class C, and not class A or class B. When all the probabilities are calculated, the class with the highest probability is used as the result.

The next classifier used was Multinomial Naive Bayes. This classifier is called “naive” because of the assumption it makes that each of the features are independent of one another, and therefore have no effect on each other. This assumption, while not true in many contexts including this one, can still produce accurate results. While the function may not be correct in estimating the exact probability of a piece of data belonging to a certain class, it can still have

a high accuracy in predicting the correct class. Naive Bayes classifiers calculate the probability for a specific document belonging to a specific class by multiplying the probabilities that each feature, or word, in the document would be present for that specific class.

There are multiple variations on Naive Bayes classifiers, and the classifier used here was Multinomial Naive Bayes. For a specific document, Multinomial Naive Bayes classifiers are different from other Naive Bayes classifiers, such as Bernoulli Naive Bayes, in that they take into account the number of times each word appears in a specific document, and not only the presence or absence of those words. The Multinomial Naive Bayes classifiers calculate the probability that a word in the document would be present for a specific class by taking the number of times a specific word appeared in documents of that class, divided by the total number of words in documents of that specific class. When a new document is passed in as a feature vector of the count of the number of times specific words appeared in the document, the class that has the highest probability of containing that document is given as the class. That probability is calculated by

$$l_{\mathrm{MNB}}(d) = \mathrm{argmax}_c \left[\log \hat{p}(\theta_c) + \sum_i f_i \log \frac{N_{ci} + \alpha_i}{N_c + \alpha} \right]$$

with f_i being the frequency of the word in the new document, N_{ci} being the number of occurrences of a specific word in a specific class, N_c being the number of words appearing across all documents in the class, α being a smoothing factor, and $p(\theta_c)$ being the probability of occurrence of a class. The class with the maximum probability of the document belonging to it, the argmax, is given as the result of the Multinomial Naive Bayes classifier (Rennie et al., 2003).

In example, let's say we have a corpus with three documents. Document A is the sentence "The fire truck pulled up to the blue house on fire after speeding down the street" and is in the "news" category. Document B is "I saw red birds swimming in the blue ocean under the blue sky" and has the category "colors". Document C contains the sentence "The truck is fantastic, with excellent safety features, and a better engine than all similar trucks", which is in the "advertisement" category. Then, we see a new sentence "My truck is red and blue" and need to determine the category. The words "My", "is" and "and" are filtered out as stop words, leaving the words "truck", "red" and "blue". To compute the probability of the class being in the news category, we apply the formula above, with the smoothing factor being one in both the numerator and denominator for simplicity. The word "truck" appears once in the new sentence, once in the "news" class and after removing stop words there are 11 words in that class. Therefore the log probability for "truck" in the news class is 1*log((1+1)/(11+1)) =-.78. The probability for "red" is 1*log((0+1)/(11+1))= -1.08 and for "blue" is also -.78. Adding these together, you get -2.64. Then the probability of the "news" class occurring is 1/3, and log(⅓) +-2.64 = -3.12, which is the Naive Bayes value of the new sentence for the "news" class. For the "colors" class, the value is -2.82, and for the "advertisement" class, the value is -3.24. Therefore, the "colors" class has the highest value, and is the class chosen by the Naive Bayes classifier, which is consistent with the most likely topic of the new sentence.

The next classifier trained and tested on the material was the K-Nearest Neighbors classifier. This classifier works by finding the most similar documents in the training dataset to the new document and giving the new document the class of the most similar documents. The classifier is set to be based on some number K nearest neighbors, which are the most similar documents to the new document. The class with the most neighbors belonging to it,

weighted by the similarity of the different neighbors to the new document, will be assigned to the new document. The number K is chosen based on training to reduce the number of errors, while still being able to classify new documents accurately (Manning & Hinrich, 1999). This classifier is used for many different cases such as facial recognition, song and movie recommendation, and text analysis.

After using the K Nearest Neighbors classifier, a LinearSVC model was trained and tested on the submission and ruling data. LinearSVC stands for linear support vector classifier. A two class linear support vector classifier assumes that if all the data is plotted, a hyperplane, which is a plane that has one less dimension than the space that it is drawn in, can be drawn between the different points on the plot to separate between the two classes. The support vectors are the points on the graph that are close to the plane which separates between the two classes. The purpose of the linear support vector classifier is to create the plane so there is the furthest distance possible between the plane and the support vectors, and therefore a clearer distinction between classes.

However, in many cases, the different classes cannot be separated completely by a hyperplane, and then there are outliers on either side of the plane belonging to the wrong class. In this case, the model works to maximize the distance between the plane and the support vectors, while also minimizing the penalty on the outliers. The penalty on any outliers increases as they move further from the hyperplane. When there are multiple classes, the Linear Support Vector classifier takes a one-vs-all approach, similar to multivariable Logistic Regression (Fletcher, 2008).

The final machine learning model generated was the Random Forest classifier. Random Forests work by building a group of decision trees and then those decision trees vote on the

final class classification. Decision trees are like a binary tree with a question at each level to determine which way to traverse. One of the problems with decision trees is that they can "overfit" the training data, which means the set of rules will be overly specific to the training data, while not generalizing well to new pieces of data. To solve this problem, Random Forest classifiers introduce an element of randomness to individual decision trees, and then take the consensus of the decision tree results to prevent overfitting.

There are two aspects of randomness introduced in Random Forest classifiers. The first is that for each decision tree in a Random Forest classifier, a random sample of the training input is taken with replacement and used as input. This means that from the submissions in the training set, only a certain amount of them are used in training a specific decision tree. The second aspect of randomness is that only a subset of the training features is used to build an individual decision tree. The features for the submissions are either represented by word frequencies, in bag-of-words and TF-IDF, or by the Doc2Vec vector representation. For a specific decision tree only a sample of words or part of the Doc2Vec vector from each of the submissions in the random sample are used to make the tree. By introducing these elements of randomness, and then taking the consensus of the decision trees to generate the result for a specific piece of data, overfitting is avoided, and the classifier's accuracy is improved (Breiman, 2001).

Results

	Bag-of-Words	TF-IDF	Doc2Vec
Logistic Regression	.39	.39	.31
Multinomial Naive Bayes	.39	.38	.29
K-Nearest Neighbors	.30	.28	.26
Linear Support Vector Classifier	.37	.36	.32
Random Forest	.30	.30	.27

Table 1: Top Ruling Prediction Accuracy

The above chart shows the accuracy for each of the machine learning models in predicting the top ruling, using the features selected by each of the feature selection methods. These results are for the most upvoted ruling on the submission, which is also the one that was assigned as the submission flair after eighteen hours. There were four options of rulings to choose from for each submission, and therefore if for every submission a random ruling had been chosen, the accuracy rate would have been close to .25. Performing at or below this benchmark would indicate a failure in using these features to predict the rulings. The accuracy results were higher than the random guessing benchmark across all selection methods and machine learning models. The most accurate classifier was Logistic Regression, with .39 accuracy for both bag-of-words and TF-IDF. The least accurate classifier was K-Nearest Neighbors, with .26 accuracy for Doc2Vec, barely above the random guessing baseline. Bag-

of-words and TF-IDF had similar accuracy for the same classifiers. Doc2Vec had lower accuracy for all of the classifiers, but still all above the random guessing baseline.

	Bag-of-Words	TF-IDF	Doc2Vec
Logistic Regression	.28	.27	.24
Multinomial Naive Bayes	.26	.26	.28
K-Nearest Neighbors	.27	.26	.25
Linear Support Vector Classifier	.30	.28	.26
Random Forest	.27	.27	.24

Table 2: Top Redditors Median Prediction Accuracy

The above chart shows the results for the median accuracy of the different machine learning models in predicting the rulings of the twenty most frequent commenters based on the features generated by the different feature selection methods. This analysis was done because if the rulings of specific redditors could be predicted, then the most popular predicted ruling among a group of redditors on a submission could be used to predict the top ruling. However, since the classifier performed around the random guessing benchmark, I did not pursue this further. The classifier with the highest accuracy was Logistic Regression on features created by the bag-of-words method and Multinomial Naive Bayes on Doc2Vec. However, while these results were the median, there were certain redditors for which higher accuracy was achieved. For example, one redditor had an accuracy rate in the .3-.4 range for the different classifiers.

Additionally, another redditor had .4 accuracy with Logistic Regression on TF-IDF selected features.

Discussion

Looking at the different feature selection methods, the performance of the models was better with bag-of-words and TF-IDF than with Doc2Vec. This could be because the document vector representation of Doc2Vec, even while better representing the context of different texts, does not capture the totality of the problems explained in the submissions. Bag-of-words and TF-IDF, while not having the context aspect of Doc2Vec, do include what they judge to be the important words from the submission, which better represents the text. Bag-of-words and TF-IDF have similar accuracy, possibly showing that the most frequent words across submissions are similar to those chosen by the TF-IDF calculation.

Across the different feature selection methods, the models that performed the best in predicting the top rulings were Logistic Regression, Multinomial Naive Bayes and Linear Support Vector Classifier, while K-Nearest Neighbors and Random Forest performed worse. K-Nearest Neighbors and Random Forest are both simpler algorithms and techniques; K-Nearest Neighbors looks at the similarity between feature vectors to decide what class a new submission belongs to, while Random Forest turns the submissions into a set of questions used to decide which class the new submission belongs to. Both of these methods are relatively simplistic. On the other hand, Logistic Regression, Multinomial Naive Bayes and Linear Support Vector Classifiers are all more complicated, looking at the results as a whole and not merely comparing each feature to features in past examples, like K Nearest Neighbors, or feeding the features into questions like the Random Forest classifier. These three classifiers

look at the features as a whole through mathematical methods, possibly having more of the nuance necessary to accurately predict the ruling on submissions.

Even though the different models were able to accurately predict the ruling for a certain amount of the test data, they were not able to correctly predict the ruling for the majority of the test submissions. This is most likely due to the complexity and subtlety of the classification problem. Deciding the appropriate ruling for a particular submission is not an objective classification, rather it is a matter of debate. The various feature selections would take into account the words of the different submissions, and then the machine learning models would try to find a pattern based on the different words. The method of feature selection and models generated were not able to correctly interpret the nuance of many of the moral questions that the commenters were asking.

However, the models were better able to predict the top ruling than the rulings of individual redditors. They were able to learn that certain features meant specific rulings. These features could be topic related, that certain words are more likely to point to a topic that is more common to a specific ruling. Additionally, it could be some sort of sentiment analysis on the different submissions, where language with specific sentiment, either positive or negative could point to a specific ruling. These features, or whichever the models actually used to get that accuracy rate for the top rulings, were learned and more consistent in the top rulings than in those for the individual redditors. One reason could be that there were more submissions for it to learn from. Additionally, it is possible that while a specific person might be inconsistent in their rulings, the overall consensus of many redditors may be more consistent.

Topic Analysis

While reading through many of the submissions, it became clear that there were a few common topics that many of the submissions focused on. To see what these topics were, a Latent Dirichlet Allocation (LDA) model was created from all the submissions. LDA models group documents of similar topics together, given a specific number of topics. It does this through a set of assumptions. The first is that each document was generated by assigning it certain percentages of different topics, for example document A is 90% about family and 10% about money. Then, within each of those topics there is a distribution of word probabilities. For example, the topic family has the words father, mother, spouse, house, and salary with different probabilities of appearing in the topic. The LDA method then assumes that each word in the document is generated by first choosing a topic based on the topic probabilities of the document, and then choosing a word from within that topic based on the word probabilities. The LDA method generates the word probabilities for each topic, and calculates the probability of each document belonging to each topic based on these assumptions. However, it does not label the topics. If the model works well, the topic will be obvious based on the words with the high probabilities in that topic (Blei et al., 2003). Below is a chart showing the top ten words for each of the seven topics the LDA analysis generated.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7
food	school	friend	work	money	room	mom
gift	class	time	time	month	dog	family

cat	time	thing	day	year	house	year
thing	game	people	car	job	roommate	sister
stuff	thing	year	hour	time	door	dad
dish	year	day	job	wedding	time	parent
clothe(s)	student	girl	people	week	night	kid
dinner	people	guy	week	trip	apartement	brother
table	college	group	minute	car	bed	time
meal	teacher	boyfriend	phone	day	day	mother

Table 3: LDA Analysis- Top 10 Words per Topic

Based on these words, the different topics can be inferred. Topic one would appear to be about items. Topic two is about school. Topic three seems to be about different people who are not family, from the words: *people*, *guy*, *girl*, *boyfriend*. Topics four and five have some overlap, both being about work. However, topic four focuses more on the work aspect, from the words *work*, *job* and maybe *phone*, also being related to work. Topic five focuses more on the monetary aspect of the job with monetary considerations such as *money*, *wedding*, *trip* and *job*. Topic six has words about living space such as *house*, *roommate*, *apartment* and *bed*. Topic seven has words referring to family such as *mom*, *family*, *sister*, and *dad*.

After generating the topics from the LDA model, I then looked at the distributions of rulings within each topic, to see if some topics had more of a leaning towards one ruling more

than the others. For a submission, I assigned the topic with the highest probability as that submission's topic. Then, I gathered the distributions of top rulings on submissions within each topic. The distribution within each topic is shown in the pie charts below.

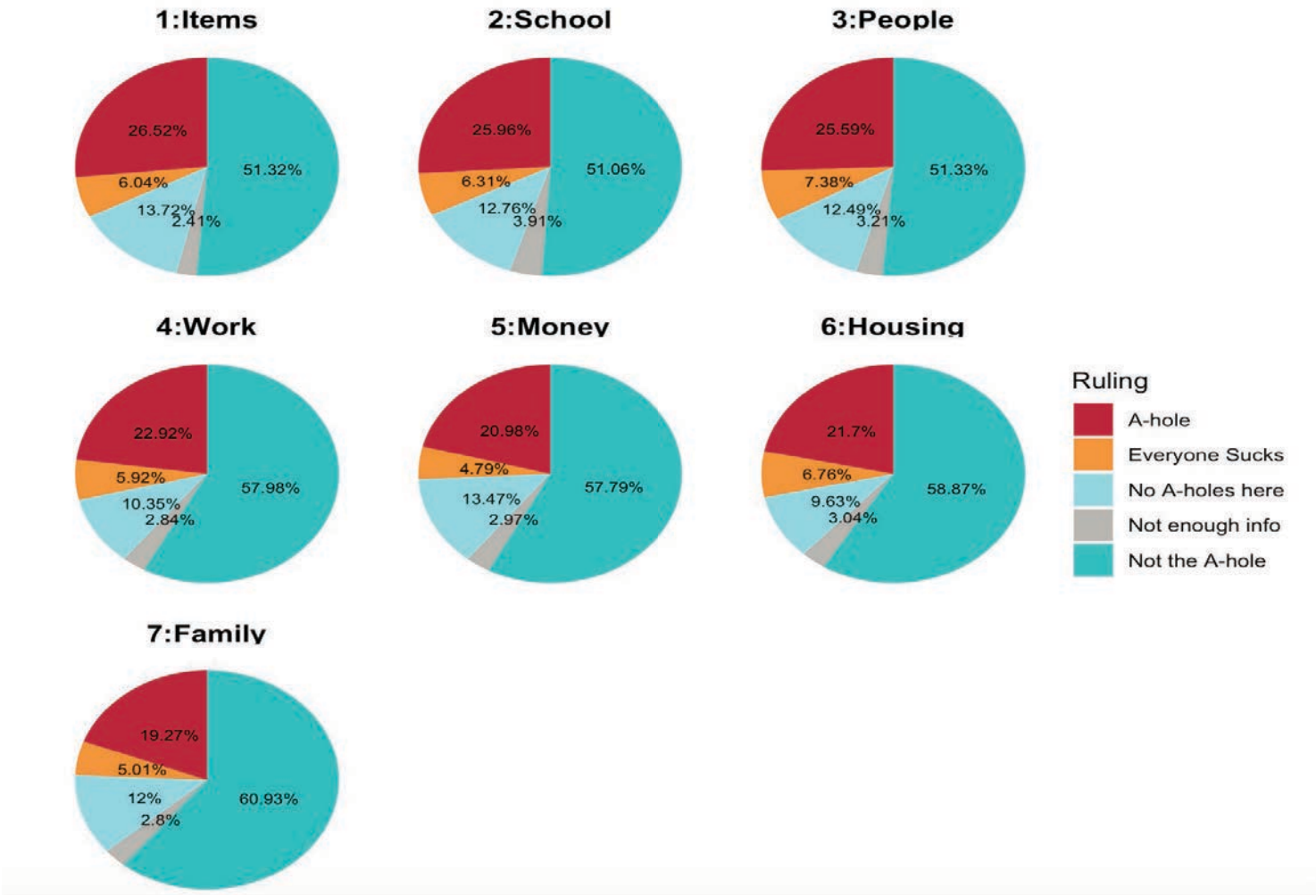

Figure 1: Distributions of Rulings within Topics

When comparing the percentages within the pie charts, it becomes apparent that certain topics have higher percentages of certain rulings. The topic with the highest percentage of A-hole rulings was *items*, with 26.52%, then *school*, *people*, *work*, *housing*, and *money*. The topic with the lowest percentages of A-hole rulings was *family*, with only 19.27%. With a slightly

different order, the topic with the highest percentage of Not the A-hole (NTA) rulings was *family*, with 60.93%, then *housing*, *work*, *money*, *people*, and *items*. *School* had the lowest percentage of NTA rulings, with 51.06%. This shows that people tend to be more sympathetic towards issues involving family, and are least sympathetic towards issues involving school and items. This may be because many people experience issues involving family, and the issues are often complicated, with multiple people involved and a big gray area, leading people to be more sympathetic and side with the writer of the submission.

On the other hand, issues involving school are usually more straightforward, and those writing submissions about school issues tend to be younger leading to less mature writing and perspective. This could cause people to have less sympathy towards the writer if they feel the writer is immature and do not have a developed moral compass yet. Issues involving items are also less sympathetic, as there are usually two basic scenarios involving items, either someone took or broke the writer's items, or the writer took or broke someone else's items. If the writer's items were taken or broken, usually they know they are in the right, and don't post in the group, leading most of the posts involving items to be about the writer taking someone else's items, which is usually wrong. Therefore, most of the rulings involving items are YTA. As a result of these different distributions, I tried adding the topic as a feature in the classifier to see if it would increase accuracy. It did not increase accuracy, which could be because the topic is already represented in the models based on the other features.

Another interesting point is that overall, NTA was the ruling for over half of all submissions, with YTA as a trailing second, with 19% - 26% of submissions across topics. The NTA ruling being the highest by far can be attributed to the story being told only by the writer in most cases, who might leave out certain parts that paint them in a negative light. Then, YTA

is the second most frequent. NAH and ESH are both infrequently used rulings, possibly because in most scenarios there is one party who is more in the right than the others. INFO is a rarely used ruling because after being used a few times in the comments, the writer of the submission will often edit the submission to include more details so that a ruling can be determined.

Age and Gender Analysis

Besides looking at the content of the submissions, another interesting point to consider is the age and gender of the individuals writing the submissions. These characteristics may change the type of problems encountered. It can also provide insight into the people commenting on the subreddit, if the assumption can be made that there is a similar age and gender distribution for both the writers of the submissions and the writers of the comments on the submissions. The age and gender of the writer of the submission can sometimes be extracted from the text of the submission. The writer will include the information in the format "first_person_word(age gender)", such as I(33F), or in a few other format variations. Using those formats, the age and gender of the writer was able to be extracted by applying a regex on the submissions. About 10% of the submissions included the age and gender. The age distribution was as follows:

Age Range	Under 10	10-19	20-29	30-39	40-49	50-59	60-69
Percent	2%	27%	58%	11%	2%	<1%	<1%

Table 4: Age Distribution Among Submission Authors

As would be expected, the majority of submission writers are in their twenties, followed by writers in their teens. When looking at the gender of the writers, about 64% were females,

while 36% were male. That means there were almost double the number of submissions by females as males. However, the actual distribution could be different if more females include their gender in the text of submission than males. Interestingly, the distribution of rulings among male and female writers were different.

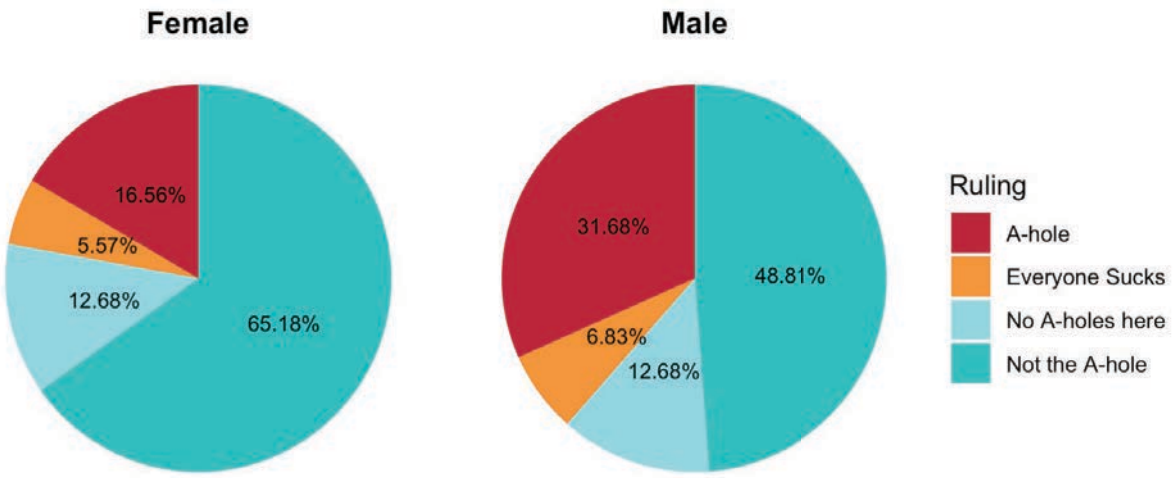

Figure 2: Distribution of Rulings for Female and Male Submission Authors

As can be seen from the pie charts, the distribution of rulings among male and female writers differ. Male writers received a YTA ruling for 31.68% of submissions, while female writers received a YTA ruling for only 16.56% of submissions. Similarly, female writers received a NTA ruling for 65.18% of submissions, while male writers received a NTA ruling for only 48.81% of rulings. It is not clear if this is due to the difference in type of question men and women are likely to post, the writing style, who is more sympathetic, or who has a better developed moral compass, but it is interesting to see the difference in percentage of rulings for each. Though there are these differences, when the age and gender were added as features in predicting the rulings it did not affect the model accuracy. This could be due to only a small percentage of submissions containing this information.

Conclusion

In this paper, an analysis was done on the AITA subreddit, looking at the submissions posted and the rulings they received. Using different feature selection methods and machine learning algorithms, the models attempted to learn to predict the ruling for a new submission. For learning to predict the ruling given by specific prolific redditors, the models had an accuracy of slightly above a random guessing baseline. However, to predict the top ruling on a submission, the models achieved an accuracy well above the random guessing baseline. Additionally, analysis was done on the topics of different submissions, and how the distributions of rulings were different among different topics. A similar analysis was then done on the difference in the distributions of rulings given to male and female writers of submissions.

There is more analysis that can be done on the data in this subreddit. The subreddit is a very interesting data source, giving a large amount of data on different moral situations and how people would judge them. More complex text analysis on the submissions to create features that are a more accurate representation of the data could be helpful in teaching the models to better predict the rulings on different submissions. However, the moral questions asked are complex and have nuance and subtlety that often leave people arguing among themselves. Therefore, with the text processing algorithms available now, it is unclear if it is possible to create models to predict the ruling given more accurately. As more complex and precise text processing methods are created, perhaps it will be possible, using this data, to create models that machines can use to answer moral questions, giving them an artificial moral compass.

Works Cited

Bengio, Y. & Ducharme, Réjean & Vincent, Pascal. (2000). A Neural Probabilistic Language Model. Journal of Machine Learning Research. 3. 932-938. 10.1162/153244303322533223.

Blei, David & Ng, Andrew & Jordan, Michael. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research. 3. 993-1022. 10.1162/jmlr.2003.3.4-5.993.

Breiman, Leo (2001) "Random Forests" Machine Learning, 45, 5-32. Retrieved from https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf

DeMaris, A. (2003). Logistic Regression. In Handbook of Psychology, I.B. Weiner (Ed.). doi:10.1002/0471264385.wei0220

Fletcher, Tristan. (2008). Support Vector Machines Explained. Retrieved from https://cling.csd.uwo.ca/cs860/papers/SVM_Explained.pdf

Havrlant, L., & Kreinovich, V. (2017). A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation). *International Journal of General Systems, 46*, 27 - 36.

Le, Q. & Mikolov, T.. (2014). Distributed Representations of Sentences and Documents. Proceedings of the 31st International Conference on Machine Learning, in PMLR 32(2):1188-1196

Manning, Christopher D. & Schütze, Hinrich. 1999. Foundations of statistical natural language processing. MIT Press, Cambridge, MA, USA., 295-296

Mikolov, Tomas & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of Workshop at ICLR. 2013.

Rennie, Jason & Shih, Lawrence & Teevan, Jaime & Karger, David. (2003). Tackling the Poor Assumptions of Naive Bayes Text Classifiers. Proceedings of the Twentieth International Conference on Machine Learning. 41.