Senior Thesis: Predicting a Comment's Popularity on Reddit

Zeno Barcutean
Department of Computer Science
University of Illinois

May 7, 2015

Contents

Chapter 1

Introduction

Becoming famous or popular has long been the goal of many people. Whether they think it's the closest thing to immortality, the most visible marker of high status or desire of social acceptance, only a select few have achieved this goal; however, today's technology gives any person just the right amount of exposure in order to gain their fame (even if only for 15 minutes).

The virtual social networks allow others to interact with various users who share their ideas or interests, even if they are geographically separated. The larger audience means anybody can broadcast their content and are able to get immediate feedback.

Some social media services give the possibility of rating a user's content, therefore giving them feedback on what they have posted. The desire to gather as much positive feedback as possible is nothing new - we have all seen artists, businesses or politicians following this concept. Aside from the status on the social network of choice, a user gathering positive feedback might get access to various perks, from increased visibility of their content to accessing exclusive areas. This reward system benefits both sides, as the owners of the social network make sure their users are active and the users enjoy the constant instantaneous gratification. Considering how much importance some users place on the feedback they receive and the popularity of algorithms that analyse their previous social media activity, I have no doubt my project would be relevant to many of them.

Chapter 2

The Task and Data

2.1 Reddit

Reddit is a social networking site where the users can submit content in the form of either text or links to other websites. The name of the site is a play-on-words of "read it", where the verb "read" is in the past-participle form (i.e. "You read it on a social media service"). Once a user is registered, they can maintain anonymity and offer feedback on the user content they come across. Even though this content is visible to everybody, only registered users can make posts and offer feedback.

2.1.1 Subreddits

The content of the site is categorized into "subreddits", according to the topic; any user can post an entry, whether text or link, on a subreddit. Users can subscribe to subreddits that coincide with their interests and the most popular posts will appear on the users' front page. A new subreddit can be created by any registered user, therefore the total number of subreddits as of 2014 is around 340,000. At the time of this writing, the most popular subreddit was r/AskReddit.

2.1.2 Voting

The registered users can up-vote or down-vote any post they come across. When a user makes a post, one up-vote is assigned to it by default. Popular comments will give the user comment karma, while popular links will give link karma; the score of a post is calculated as the number of up-votes minus the number of the down-votes. The amount of karma is a good indicator

of a user's popularity and time spent on reddit; although there are virtual "badges" given for high-karma posts, most of the time these badges are overlooked in the favor of the raw karma numbers. A post with a high-score will have a higher visibility and is more likely to be displayed, while comments with lower scores are usually collapsed; users can select whether they want to see 200 or 500 posts under each entry.

2.1.3 Reddit Gold

If a user finds content that they consider was of high-quality, they are able to give "reddit gold" to the author. Reddit gold allows access to the r/lounge subreddit, as well as the possibility to award reddit gold to another user. Although reddit allows users to "friend" one another, it doesn't have the same focus or functionality that a website like Facebook does.

2.2 The Dataset

The raw data was downloaded in a .csv file of over 300,000 comments from AskReddit, posted between February 27, 2015 and March 2, 2015. The comments were downloaded on March 3, since it usually takes about a day for popular entries to stop showing up on the front page of r/AskReddit and some users might still be voting them in that case. Reddit makes all this data available in JSON format so it is easy to sort and find the set of comments needed. Comments that had a score of NULL were removed, leaving 308,200 comments to work with.

2.2.1 Categories

Because AskReddit is the most active subreddit, I consider a comment which has a score between 1 and 5 to be mediocre. That means it was barely noticed by the multitude of users present on the site.

Similarly, a comment which has score 0 or lower is a bad comment, since it received more down-votes than up-votes (it got at least one downvote, since all comments have one upvote by default).

Next, I consider the good comments to have score over 5. Such comments were noticed by enough people to get a higher number of upvotes relative to the downvotes. They were marked as "interesting" and they are the ones that grant the user the karma-points they are after.

This breakdown is specific to r/AskReddit and my own perception of comment quality. Others might have a different notion of how high a score

means a comment is good, as well as different subreddits having different distributions. For example: r/uiuc, the subreddit of University of Illinois at Urbana-Champaign, is much smaller than r/AskReddit and the scores of comments are in a much smaller range.

Examples

In some cases, the questions asked are looking for an informative answer and posting an interesting, somewhat unknown fact will grant a large amount of upvotes, while posting something funny looking for a reaction (also known as "trolling") will not do as well.

Here is a good comment posted by u/Prufrock451, with a current score of 2500

> General James Wilkinson was the senior commander in the U.S. Army; he spent decades at the highest levels of the U.S. government. He was often suspected of double-dealing, but nothing was ever pinned on him - until decades later, when the United States uncovered Spanish secret archives during the invasion of Cuba in the Spanish-American War. The highest-ranking officer in the U.S. Army was a Spanish agent, who had tried to convince Kentucky to secede and ally with Spain before the Louisiana Purchase. He tried to give Lewis and Clark's position to the Spanish so they could "disappear" the expedition. He collaborated with Aaron Burr in trying to create a new empire in the Southwest. One of the worst traitors in American history.

A bad comment, with a current score of -3, posted by u/notaneggspert

> I'm a firm believer that the CIA killed Bob Marley with a radioactive nail planted in his shoe given to him as a gift. He died of toe cancer after getting pricked by the nail.

Both comments were posted on April 17th, 2015, under the entry "What conspiracy theories ended up being true?" - in this case, the longer comment provided a serious answer that covered most of the arising questions. The short comment was obviously a joke (the username is "notaneggspert") and even though it might have fared well in a different situation, it was deemed inappropriate here.

In a different situation, a question that starts as serious could evolve into a series of jokes in which the users try to outwit each other. For example,

on July 8, 2014, shortly after Germany faced Brazil in the World Cup semi-final, the user u/sliceeguy asked the question: "Brazilians of reddit, how is your country reacting to the semi-final in the world cup?" The original answers were mostly people from Brazil describing how they perceived the game, but as soon as the entry got popular, most of the users were making fun of Brazil's performance. In one instance, after a user jokingly speculated that the game will end up being the most-viewed video on an adult website, the notorious user vargas replied with

> At least the Brazilian classic "Two Girls, One Cup" finally got a sequel: "198 Million People, No Cup."

His quote received reddit gold and has a score of 3861.

Lastly, mistakes that cause hilarity are also appreciated. On an entry where married people were discussing things they found out about their significant other, the user u/zerocoke accidentally posted a comment which he intended to send as a text message to someone. As of right now, the comment

> We're closing. Come get your grape juice.

has 6374 points and has received reddit gold four times, despite being blatantly off-topic and derailing the conversation.

2.2.2 Dataset Breakdown

According to my categories for comments, this is the current breakdown:

Table 2.1: Dataset Breakdown

Good comments	37151
Mediocre comments	245030
Unpopular comments	26019

As usual, most of the comments had the score of 1, which is the default value, so the vast majority of my instances fall in the Mediocre category. As the score goes higher or lower, the number of comments found asymptotically approaches 1. Since we are focusing on the comments that can bring a significant change of karma points to the user, the mediocre comments will be removed from the dataset and I will look at what makes a good or a bad comment. The baseline accuracy, obtained by assigning the majority label to all instances, is approximately 58 percent.

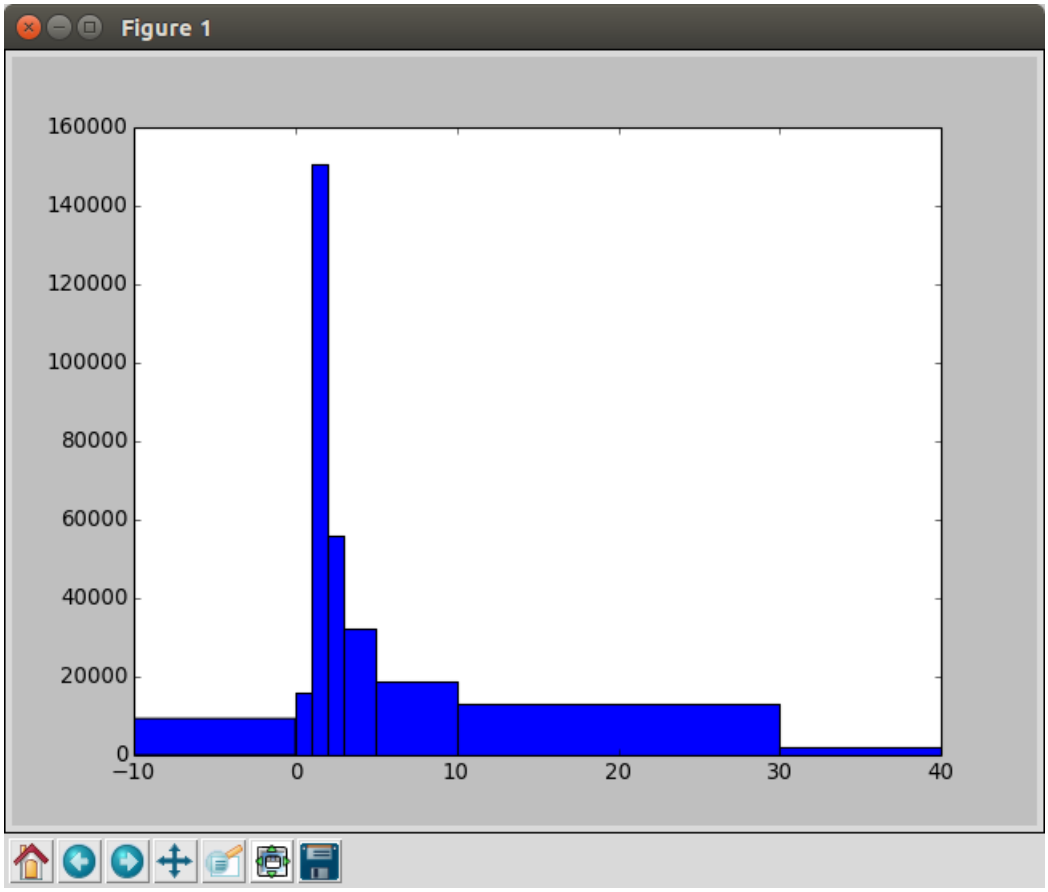

Figure 2.1: Distribution of the number of comments relative to their score

2.2.3 Processing the Dataset

The dataset will be converted to arff form, from its current csv form. As of right now, the raw data presents all the features reddit stores about the comments posted. I will use the comment's score to create the class label.

For training and testing, I am splitting the dataset into 95 percent for training and 5 percent for testing, giving me 60012 instances to train on. The data is shuffled each run and then split into training instances and testing instances.

2.3 The Task

This project will create a system which will predict how popular a comment will be once posted on a social media network. In this case, I will focus on reddit. A comment's popularity is determined by the amount of points it gets from the other users. The points of a comment are equal to its score.

Since reddit is split into areas of interest called subreddits, the content on the entries vary so the predictions of my model will be specific to a certain subreddit. For this project I decided to focus on the most popular subreddit (at the moment), AskReddit.

2.3.1 Motivation

The points for each comment accumulate as the user's karma. Generally, the more karma users have, the higher the quality of their comments. As always, there are many users openly declaring that they want as much karma as possible (works in a similar fashion to the Facebook 'likes') and generally a negative score is to be avoided as it will get subtracted from the user's karma. Since so many users are self-conscious about their karma, I imagine my project would be of interest to many of them.

2.3.2 Testing

Dataset Usage

When running the algorithm, the dataset is shuffled each time. The first 95 percent of the instances is used for training purposes in order to build the classifier. The other 5 percent will be the unseen instances and will be used for testing. If the prediction of a comment from the unseen examples is equal to the comment's category, then that example is considered correctly classified. Otherwise, it counts as an error.

Metrics

The accuracy is calculated as the sum of correctly classified instances divided by the number of instances in the test dataset, or

$$Accuracy = \frac{CorrectlyClassified}{UnseenExamples}$$

The breakdown for each category will be computed in terms of precision and recall.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

where TP are the instances which were correctly classified as belonging to the class, FP instances which were incorrectly classified as belonging to the class and FN are the instances incorrectly classified as not belonging to the class.

In binary classification, TP stands for True Positive, FP for False Positive and FN for False Negative.

2.4 Related Work

2.4.1 Predicting Comment Score

Lamberson et al. [2] chose to focus less on the content of the comment, but rather on the other characteristics. Their classification was also binary, with comments being marked as GOOD in case they had a positive score and BAD when they had a negative score.

The features used were split in four categories: timing, relevance to the entry, word counts and derived features such as sentiment analysis (positive/negative).

Three algorithms were used:

1) Naive Bayes using unigram features

2) Support Vector Machine with Sequential Minimal Optimization, using only the metadata and not the unigrams

3) Support Vector Machine with Sequential Minimal Optimization where the output of the Naive Bayes algorithm was an extra feature (a combination of the previous two algorithms)

The results indicate the second algorithm performed the best, although the third one was far better in terms of recall. Their conclusion was that short, profane comments that are replies to popular posts are the most likely to get upvoted.

2.4.2 Sentiment Analysis on the Reddit Comments

Past work involving classifying reddit comments based on their content dealt with sentiment analysis: looking at a user's comments, determining whether each comment was positive or negative and evaluating the user's overall happiness/angst from the results.

There are two more popular such models:

https://www.sentimentview.com - a reddit bot that takes requests and analyses each user's comment history

https://api.blockspring.com/robert-wett - this is a website where you can look up a certain reddit username and see a breakdown of the positive and negative comments along with the overall attitude of the user

2.4.3 Reddit Posts

Other related work deals with the popularity of the posts rather than the comments: they look up what features would make a successful entry on a subreddit.

Popularity of Posts on Various Subreddits

Segall et al. [5] worked on predicting the maximum score a post will reach in its lifetime. They dealt with target demographic, the type of entry and the location of the post while using two algorithms: Naive Bayes and Multi-Class SVM.

Their methods focused mostly on reducing overfitting and the classification error. One of the findings was that training and testing the classifier should be done on only one subreddit, as the error was very large when dealing with all the subreddits in the dataset. Another finding was that some subreddits are more predictable than others, such as those tagged NSFW (Not Safe For Work).

The one thing to note would be that while they used the words in the title as features, they noticed a significant improvement when they reduced the number of features by removing the stop words. This is because those stop words are less important than the key words, but were still picked up by the algorithm.

Their two algorithms performed very similar to one-another and they stated that although small differences appeared in regard to dataset size, they were so small they could be attributed to random sampling and no strong conclusions could be made.

Popularity of Reposts

Lakkaraju et al. [1] dealt mostly with reposts and analysing when they did well. Three crucial things are:

1) the relationship of the content with the theme of the subreddit - on some subreddits (like r/atheism), entries that are fairly similar to other entries do better; however, on subreddits like r/news, similarity to other entries is framed upon

2) time of the day - posting at noon seems to be more successful, while posting during the night doesn't go as well; the time is UTC, which indicates that the target demographic of reddit is the western world

3) originality of the content - as they kept reposting, they found out the popularity decreased. One workaround was to have some time delay between reposts, which helps bring the same content to a new audience

Chapter 3

The Models

3.1 Classifiers

The experiments were run using the classifiers provided by Weka. They were chosen based on memory used, running time and expected accuracy.

Weka (named after the flightless bird found in New Zeeland) is a collection of open-source software issued under the GNU General Public License by the University of Waikato. It contains learning algorithms for data mining and although it has its own Graphical User Interface, it can also be imported into Java code.

3.1.1 Naive Bayes Classifier

Naive Bayes Multinomial

The Naive Bayes classifier is a probabilistic classifier. It looks at all the instances and their class, then calculates the probability of each instance having a certain value for each of the attributes while going through all the possible values for the class. Each $p(x_i|C)$ is computed as follows:

$$p(x_i|C) = \frac{count(x_i, C) + 1}{count(C) + |V|}$$

where $|V|$ is the number of values the ith feature can take.

Once all the $p(x_i|C_j)$ are calculated from the training set (where x_i represents each attribute of our instances, and C_j represents any of our classes), we can iterate through the classes and choose the hypothesis that yields the greatest probability. For instance, given the features $x_1, x_2, ..., x_n$, the probability of this instance having class C_k is

$$p(C_k|x_1, x_2, ..., x_n) = argmax_k(p(c_k)p(x_1|C_k)p(x_2|C_k)...p(x_n|C_k))$$

As an example, if our attributes were Fever(yes/no), Sore Throat (yes/no) and the class attribute Flu(yes/no), we would look over all our given instances and find the probability that we have fever when we have the flu, we have fever when we don't have the flu, we don't have fever when we have the flu etc. until all possible combinations have a probability.

Because the algorithm calculates the probability of each attribute's value given the value of the class attribute, its running time is linear in the number of features.

Naive Bayes using the Bernoulli model

The WEKA classifier implementing this algorithm is the NaiveBayesMultinomial class. There is a variation on this, using the Bernoulli model. This model generates an indicator for each term of the vocabulary, using a 1 to indicate presence of the term in the document and 0 to indicate absence. Since this model ignores the number of occurrences and focuses on binary occurrence information, the multinomial model is usually expected to perform better on text classification [3].

My experiments deal with using the number of uni-grams and bi-grams in the body of the comment, hence the Bernoulli model will not be used.

3.1.2 Logistic Regression

The algorithm for logistic regression is a direct probability model used to predict an instance's class based on the value of the features. It is based on the linear regression model, where given an instance x and a weight vector w we compute the dot product $w^T x$ and use it to model a probability.

For example, considering $p(getsick = 1) = w_0 + wx$, we have the probability of getting sick increasing by w each time we add 1 to x. Since the probability of an event must be positive and less than 1, a sigmoid function (called the "logistic" function) is used to obtain the correct values, as follows:

- to make sure the value of the logistic function is positive, an exponential is used. Euler's number, e, is positive, therefore $exp(w_0 + wx)$ will always be positive - to make sure the function is always less than or equal to 1, we can divide the exponential by anything greater than itself. By convention, the

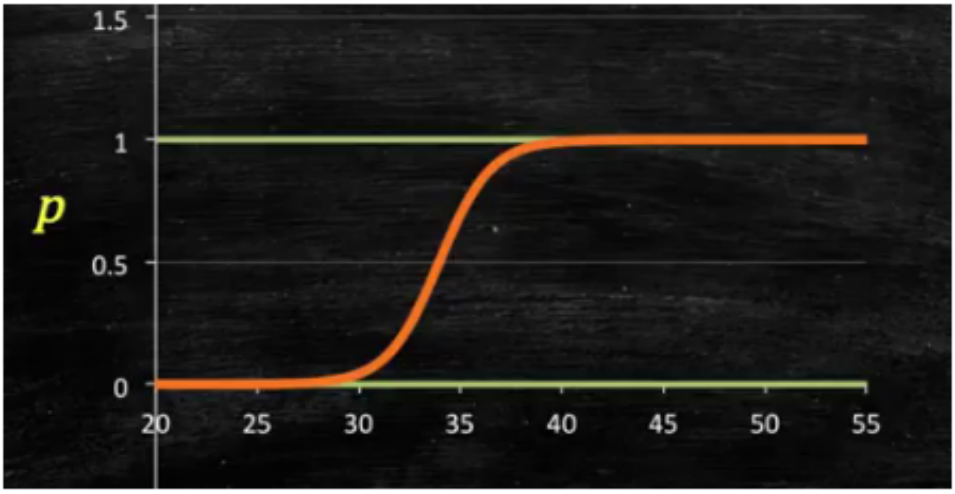

Figure 3.1: Image courtesy of INCAE Business School

number 1 is added to the denominator, resulting in the probability being

$$\frac{exp(w_0 + wx)}{exp(w_0 + wx) + 1}$$

. The new asymptotic function will look like the one in figure 3.1.

3.1.3 Discriminative Multinomial Naive Bayes

A middle point between Naive Bayes and Logistic Regression is the Discriminative Naive Bayes. It runs almost the same as Naive Bayes, except it also includes the posterior probability when updating the frequencies for words. The posterior is computed using the formula

$$p(c|d) = \frac{p(c)\Pi_{i=1}^{n} p(w_i|c)^f i}{p(d)}$$

where f_i is the number of occurrences of word w_i in the instance d and n is the number of unique words appearing in d. The probability $p(w_i|c)$ is computed by counting the occurrences of the word given the class, just like for the Naive Bayes Multinomial algorithm. In the posterior probability $p(c|d)$, c is the true class of d. The actual $p(c|d)$ isn't known in practice, so the algorithm assumes the probability to be 1 when d is in class c within the dataset. Therefore, the loss function on a training document d would be

the actual probability minus the posterior probability, or $L(d) = 1 - p(c|d)$, where $p(c|d)$ is computed using the above formula.

The loss function is then used in the update rule: when computing the frequencies, at each step the current classifier is applied on each instance. The frequencies are then updated based on how the current classifier performs on the instance: $f_{ic} = f_{ic} + L(d)$ where f_{ic} is the frequency of word w_i in the instance d. Therefore, if the classifier performed correctly on the instance, the loss will be zero and the corresponding frequencies do not have to be updated.

According to Su et al [6] the discriminative models, although less efficient, are generally more accurate than the generative models. I expect this model to give me a higher accuracy on my data.

3.2 Features

3.2.1 Content

The features I intend to use can be categorized in two sections: first, I will use the words from the comment's body and the link title as features; for my experiments, I am using both unigrams and bigrams, where the minimum frequency of a word is set to ten (in order to avoid any misspelled words). A bigram will not appear unless the frequency of both its words are at least 10 - for example, if our bigram is "rocket engine", where "rocket" appears 20 times and "engine" only 5 times, the bigram will not be a feature. Also, stop words are removed from the unigrams but not the bigrams, so we can have features like "not good". Using this configuration yields about 125,000 features for the model. The link title is important in determining whether the content of the comment was on topic - usually off-topic comments get low scores, although some (mostly the accidental ones) are upvoted because of the hilarity caused.

3.2.2 Metadata

Second, I will use the following as metadata:

- the word count; according to my distribution of comments, I opted for these categories - comments that have less than 20 words, comments that have between 20 and 30 words, comments that have between 30 and 53 words and comments that have over 53 words. Some of the comments are much longer than 53 words, but the bulk are usually short. Lamberson et

al [2] found that short, profane comments usually perform well. However, long, informative comments also have good scores.

- the sentence count - once again, the breakdown is skewed towards shorter comments, therefore the breakdown is as follows: comments with less than two sentences, comments that have between two and four sentences, comments that have between four and nine sentences and comments that have over nine sentences. Splitting the comments into sentences was done using a regular expression that looked at periods, question marks or exclamation marks, where having two or more of these characters counted as one - for example, "??!!!" counted as one delimiter between two sentences.

- a binary feature that indicates whether this is a top-level comment or a reply; since reddit collapses nested replies, top-level comments have more visibility and therefore get more votes. To compute this feature I looked at the id of the parent - if the parent was an entry then this was a top-level comment, otherwise it was a reply.

- the time of day the comment was posted; after analysing my comment distribution, I consider that having 24 values for this feature (corresponding to a 24-hour format) would work best - there were local maxima for the number of comments posted where the local maximum for the good comments was one hour ahead of the local maximum for the bad comments

- the author; certain authors automatically get more upvotes because they are recognized by the audience of the subreddit. Since most of WEKA's classifiers cannot work with String attributes, I am using the absolute value of the hash-code for this feature.

- a binary feature that indicates whether there is profanity or not in the body of the comment; usually short, profane comments tend to be quite popular since they are concise and transmit either a strong or a humorous message. This claim is also supported by Lamberson et al [2]. I used a list of profane words and then checked whether the body of the comment contained any of them.

Chapter 4

Experimental Results

4.1 Global Setup

The 63170 instances were split into a training set and a testing set, as follows: during each run, the instances were shuffled and then the first 95 percent would be used for training, while the remaining 5 percent used for testing.

The unigrams and bigrams had their values computed using TF-IDF values. TF-IDF stands for "term frequency - inverse document frequency"; it computes how important a word is to a certain part of text within the corpus - that is, the value increases proportionally with the number of times the word appears in the document but decreases proportionally with the number of times the word appears in the whole corpus. For example, even though a conjunction might appear many times in a document, it would also appear many times in the whole corpus so its TF-IDF score will be low. However, if the word "processor" appears many times in a certain document but not that often in the whole corpus, that means "processor" is important to that one document and can give us an idea of what class the document belongs to.

4.2 Naive Bayes Multinomial

4.2.1 Experimental Setup

The NaiveBayesMultinomial provided by WEKA was used for this run. The features included unigrams and bigrams collected from the title of the post and the body of the comment, as well as metadata. The tokenizer removed all non-alphanumeric characters from the text; the minimum term frequency

is 10, to eliminate most of the misspelled words.

4.2.2 Results

The accuracy was 62.01

Confusion Matrix - The column represents the actual class, the row represents what the algorithm classified it as. For example, out of all the good comments, 1049 were classified correctly and 783 classified as bad.

Table 4.1: Naive Bayes Multinomial Confusion Matrix

	Good	Bad
Good	1049	783
Bad	415	909

Table 4.2: Naive Bayes Multinomial Precision and Recall

	Precision	Recall
Good	0.7165	0.5725
Bad	0.5372	0.6855

4.3 Discriminative Multinomial Naive Bayes

4.3.1 Experimental Setup

The DMNBtext classifier provided by WEKA was used for this run. The features included unigrams and bigrams collected from the title of the post and the body of the comment, as well as metadata. The tokenizer removed all non-alphanumeric characters from the text; the minimum term frequency is 10, to eliminate most of the misspelled words.

Since discriminative models are expected to perform better than generative models [6], this classifier is expected to perform better than the other Naive Bayes classifier running on unigrams and bigrams.

4.3.2 Results

The accuracy was 72.07

Confusion Matrix - The column represents the actual class, the row represents what the algorithm classified it as. For example, out of all the good

comments, 1605 were classified correctly and 272 classified as bad.

Table 4.3: Discriminative Naive Bayes Confusion Matrix

	Good	Bad
Good	1605	272
Bad	610	671

Table 4.4: Discriminative Naive Bayes Precision and Recall

	Precision	Recall
Good	0.7246	0.8551
Bad	0.7115	0.5238

4.4 Logistic Regression

4.4.1 Experimental Setup

The Logistic classifier provided by WEKA implements the multinomial logistic regression algorithm. The features included only the metadata. I performed the test on metadata using multiple classifiers, they all obtained similar accuracies on the metadata, although the accuracy for logistic regression was negligibly higher.

Unlike the Naive Bayes classifiers, the logistic regression's running time is not linear in the size of the features. It also went constantly over 12 GB RAM when running, therefore it was not used with bigrams and unigrams.

4.4.2 Results

The accuracy was 59.75

Confusion Matrix - The column represents the actual class, the row represents what the algorithm classified it as. For example, out of all the good comments, 1885 were classified correctly and 0 classified as bad.

Table 4.5: Logistic Regression Confusion Matrix

	Good	Bad
Good	1885	0
Bad	1271	2

Table 4.6: Logistic Regression Precision and Recall

	Precision	Recall
Good	0.5972	1.0
Bad	0.9984	0.00157

4.5 Discussion

Table 4.7: Accuracies of all the experiments performed

Classifier	Accuracy	Setup
Baseline	58.81	N/A
Naive Bayes Multinomial	68.17	Unigrams and bigrams only
Discriminative Naive Bayes	69.85	Unigrams and bigrams only
Naive Bayes Multinomial	62.01	Unigrams, bigrams and metadata
Discriminative Naive Bayes	72.07	Unigrams, bigrams and metadata
Logistic Regression	59.75	Metadata only
Naive Bayes Multinomial	57.88	Metadata only

4.5.1 Metadata's Effect on Accuracy

Table 4.8: Accuracy using only metadata

Omitted Feature	Accuracy
N/A	59.75
Word Count	58.39
Author	59.15
Number of Sentences	57.41
Depth	57.85
Profanity	58.23

After removing the metadata features one at a time from the model, the accuracies obtained have negligible differences. The two most significant ones were the number of sentences and the depth, meaning that keeping track of longer/more fragmented comments or whether the comment is a top-level comment or a reply matter the most. However, the importance of using

metadata in the model is evident from the accuracy of the Discriminative Naive Bayes, which received a 3 percent bump as soon as the features were added.

4.5.2 Classifier Results

With a baseline accuracy of about 58 percent, the logistic regression and the naive bayes models did not perform too well. Surprisingly, the metadata scores were lower than the scores that had the unigrams and bigrams. Lamberson et al [2] achieved the best performance by running the classifiers on metadata only. However, all their tests had accuracies in the 50s, with the metadata-only test in high-50s. My accuracy using both metadata and unigrams and bigrams was in the 70s.

If we look at the confusion matrix of the logistic regression, we can see that it classified the examples almost the same way the baseline classifier would - the majority of the examples - the good comments, which were in greater number - were all correctly classified. Almost none of the instances for the bad comments were classifed correctly. Segall et all [5] had problems with their models overfitting the data; their dataset initially had comments from multiple subreddits, so they had to first focus on each particular subreddit then reduce the number of features to avoid overfitting. I started with r/AskReddit so I was already focused on a specific subreddit, but felt that just the metadata alone does not make a model expressive enough and the results I got confirmed my suspicions.

The Multinomial Naive Bayes classifier had an accuracy of 62.01 percent, which isn't much higher than the one obtained by the logistic regression relative to the baseline accuracy, however things change when we look at the confusion matrix. Both the majority of the good comments and bad comments were correctly classified, so this more expressive model including both bigrams and unigrams worked better for my task. However, the accuracy was still relatively low, but this was the generative model and the discriminative model was expected to behave better. The interesting result was the 6 percent bump in accuracy when the metadata was removed, suggesting that the model might have problems overfitting when the number of features is increased.

The Discriminative Naive Bayes classifier gave the best accuracy - 72.07 percent. The accuracy itself is over ten percent better than the generative model, but the precision, recall and confusion matrix are more informative in this case. The majority of both good and bad comments were correctly classified, but this time almost all of the good comments were classified

correctly - we can look at the recall for good comments: 0.8851 versus 0.5725. Although the recall for bad comments was not as high, the difference was not enough to offset the recall with which it classified the good comments, offering an overall better classification.

Chapter 5

Future Work and Conclusion

There are ways through which the results could be improved in the future. First, obtaining a larger dataset from which I can extract only the comments that also have their parents in the dataset; measuring their relevance to the parents would give me the opportunity to add features like keeping track of the depth rather than a binary feature that only tells me whether I am dealing with a top-level comment or not. Running the experiments on a system with more memory would give me an opportunity to experiment with other classifiers, that do not scale as well with a large number of instances or features. A decision tree, for example, has the running time exponential in the number of features and running it on bigrams and unigrams would take a lot of time. Similarly, linear classifiers like the logistic regression always run out of memory quickly when dealing with a large number of features. Lastly, experimenting with multiclass classification would allow for better precision on how many points the comment will get - right now a comment with a score of 10 and a comment with a score of 4000 are both considered as good. One challenge with multiclass classification will be a skewed dataset breakdown - since the bulk of the comments have the default value, there will be a high baseline accuracy and most classifiers will have a hard time beating it.

Ultimately, predicting how well a comment will do on r/AskReddit is a hard task and it is difficult to assess whether its popularity comes from its content or from other factors. Unfinished comments, off-topic references or posting mistakes can quickly gather numerous up-votes due to their wit, humor or sarcasm. However, detecting such emotions using libraries dedicated to sentiment analysis is impeded by slang or even terms specific to a certain part of reddit.

At the end of the day the average users will want their comments to be positively received by the community of the subreddit. My model will classify a good comment correctly most of the time which is what the users will want to know in the end. The precision for the experiment with the greatest accuracy was not the highest I obtained, but it still returned the most relevant results.

Bibliography

[1] Himabindu Lakkaraju, Julian McAuley, Jure Leskovec *What's in a name? Understanding the Interplay between Titles, Content and Communities in Social Media* Stanford University, 2013

[2] Daria Lamberson, Leo Martel, Simon Zheng *Hacking the Hivemind: Predicting Comment Karma on Internet Forums* Stanford University, 2014

[3] Vangelis Metsis, Ion Androutsopoulos, Georgios Paliouras *Spam Filtering with Naive Bayes - Which Naive Bayes?* CEAS 2006 - Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California USA

[4] Mrutyunjaya Panda, Ajith Abraham, Manas Ranjan Patra *Discriminative Multinomial Nave Bayes for Network Intrusion Detection* In Information Assurance and Security (IAS), 2010 Sixth International Conference on (pp. 5-10). IEEE.

[5] Jordan Segall and Alex Zamoshchin *Predicting Reddit Post Popularity* Stanford University, 2012

[6] Jiang Su, Harry Zhang, Charles X Ling and Stan Matwin *Discriminative Parameter Learning for Bayesian Models* University of Ottawa, 2005