The Lowest Form of Wit: Identifying Sarcasm in Social Media

Saachi Jain, Vivian Hsu

***Abstract*—Sarcasm detection is an important problem in text classification and has many applications in areas such as security, health, and sales. Previous works have tried various features such as sentiment, capitalization, and n-grams to classify sarcastic texts. Downloading a corpus of tweets through Twitter Archiver, we used multinomial naive Bayes, logistic regression, and support vector machine to classify tweets as sarcastic or not sarcastic. We found that unigrams and bigrams were the most indicative features of sarcastic tweets, and we achieved an accuracy of 80.27% using logistic regression on a data set with oversampling on sarcastic tweets.**

I. Introduction

Defined by Merriam Webster as the use of words that mean the opposite of what you really want to say, sarcasm is a counterintuitive social construct that befuddles programs and humans alike. Often cited as an exception in sentiment classification strategies, it reverses the intention of a concept or idea by relying on grammatical structure, hyperbolical vocabulary, and context.

Sarcasm can be difficult to spot, especially when expressed through written text. Humans often mistake the true sentiment that others convey in emails, messages, or posts on social media. This mistake proves to be a problem in many different contexts. For example, security agencies have trouble differentiating false comments about planning terrorist attacks from real ones. People also need to distinguish sarcastic ratings and reviews from non-sarcastic ones on websites such as Amazon and Yelp. Furthermore, some find it hard to distinguish tweets and posts that joke about depression from those that are cries for help. A sarcasm detector can not only help people interpret others' writings, but can also assist the writers themselves in avoiding being misunderstood.

An efficient sarcasm detector has proved to be difficult to implement, as many companies and research groups have tried to develop algorithms to detect sarcasm with varying success. In this paper, we use a corpus of sarcastic and non-sarcastic tweets, isolating features such as n-grams, capitalization, sentiment split, and subjectivity, to classify a tweet as sarcastic or not sarcastic using multinomial naive Bayes, logistic regression, and linear support vector machine (SVM) classifiers.

II. Related Work

Due to the significance of sarcasm in text classification, several groups have already explored using machine learning techniques to detect sarcasm in text. Throughout many of these studies, social media, especially Twitter, is often the primary data source for sarcastic and non-sarcastic texts.

A significant challenge in using supervised learning on sarcastic texts is annotating the corpus as sarcastic or not sarcastic beforehand. Dr. Mathieu Cliche from Cornell University separated tweets as sarcastic or not sarcastic according to the presence of the hashtag #sarcasm, arguing that tweets with #sarcasm are likely to be truly sarcastic tweets, and tweets without the tag, although they may contain sarcastic tweets, have a large enough corpus of regular tweets that the existent sarcastic samples in the set can be considered noise. Liebrecht et. al from Radboud University Nijmegen and Dr. David Bamman and Dr. Noah A. Smith from Carnegie Mellon University employed similar techniques to create their datasets. Gonzalez-Ibanez et. al from Rutgers University also used #sarcasm to identify sarcastic tweets, but rather than choosing non-sarcastic tweets as tweets lacking sarcastic hashtags, he used tweets presenting positive or negative tags (#happy, #sadness, #angry, etc) under the hypothesis that tweets with tags representing pure emotions are less likely to be sarcastic. Although this approach caused the non-sarcastic dataset to be less representative of general tweets, we thought that it is a better choice because it reduces the noise associated with the non-sarcastic set if it were obtained by simply choosing tweets without "#sarcasm."

Dr. Cliche, Liebrecht et. al, Bammam et. al, and Gonzalez-Ibanez et. al all had n-grams as a critical feature in their classifiers. In addition, Bammam et. al counted the number of words in all caps in a tweet as a feature. Riloff et. al from the University of Utah explored another feature involving a split in sentiment using a bootstrapping algorithm. Specifically, they found that sarcastic tweets were likely to have a positive verb phrase juxtaposed to a negative activity or state (i.e. "I love taking exams"). Their algorithm learned positive sentiment phrases and negative activity or situation phrases to recognize sarcastic tweets.

Several of the groups focused on logistic regression as a promising classifier for sarcastic tweets (Cliche, Gonzalez-Ibanez et. al, Bammam et. al). Dr. Cliche also employed multinomial naive Bayes and a linear SVM, and achieved an F-score of 0.60. Liebrecht et. al from Radboud University Nijmegen used a Balanced Winnow Classifier, based off of the perceptron algorithm, and obtained an accuracy of 75%. Gonzalez-Ibanez et. al had a 66% accuracy with logistic regression and a 71% accuracy with sequential minimal optimization algorithm (SMO). Bammam et. al used logistic regression to get an accuracy rate of 85.1%. Overall, it seems that logistic regression was the most popular and consistent classifier, while unigrams and bigrams were the more effective features.

III. Data and Feature Extraction

Our dataset consisted of English tweets obtained from Twitter through Twitter Archiver, a Google add-on that downloads tweets into a Google Spreadsheet based on filters such as hashtags and language. We downloaded tweets from November 10th to December 3rd, 2015. Based on the assumption that the writers of the tweets are the best people to judge whether their tweets are sarcastic or not, we used hashtags to annotate sarcastic tweets. We obtained sarcastic tweets by getting tweets with the hashtag #sarcastic. We further assumed that any tweets with "emotional" hashtags such as "happy," "joy," "lucky," "sad," "angry," and "disappointed" were non-sarcastic tweets expressing positive or negative sentiment.

To clean the data, we filtered out symbols and strings that did not contribute to the overall meaning of the tweets. We took out all words followed by a hashtag ("#"), all links to other websites (tokens beginning with "http"), and all tags to other accounts (tokens beginning with "@"). If a tweet has fewer than three tokens left after cleaning, we took it out of our data set. We were left with 26,206 sarcastic tweets and 101,361 non-sarcastic tweets in our dataset.

Our feature set consisted of unigrams, bigrams, capitalization, sentiment-split, and subjectivity. Each unigram and bigram was its own feature. We created the unigrams by parsing the tweets into lemmatized individual words and punctuation marks ("!", "?", and ","), and counted the overall occurrences of each in sarcastic texts and non-sarcastic texts. We created the bigrams by parsing the tweets into pairs of consecutive words, and also counted their frequencies in sarcastic and non-sarcastic texts. However, we took out punctuation marks in the bigrams because we wanted to analyze the impact that two adjacent words have on whether or not a tweet is sarcastic, rather than whether or not a punctuation mark precedes or follows a word. We then further reduced our data set by eliminating unigrams and bigrams that appear fewer than 10 times. In total, we had 8,223 unigrams and 20,790 bigrams.

Aside from unigrams and bigrams, we extrapolated three more features based on the overall context of the tweets. One of the features was the number of words in all caps (greater than one letter), as suggested by Bammam et. al. Another was sentiment-split, which captures the difference in the sentiment between the part of the tweet before the verb phrase and the part of the tweet after the verb phrase. For example, if the tweet was "I hate Christmas presents," we used the library *pattern.en* to split the tweet into two chunks: "I hate" and "Christmas presents". Using the NLP library *TextBlob*, we calculated the sentiment score for both chunks (a score of from -1 to 1, where -1 is very negative and 1 is very positive), and found the difference between the two scores as the tweet's sentiment-split score. Finally, our last feature was the subjectivity score (fact or opinion) of the entire tweet, which we also calculated using *TextBlob*.

IV. Methods

After collecting the data and extracting the features, we used three classifiers (all from the library *scikit-learn*): multinomial naive Bayes, logistic regression, and SVM, to predict whether a tweet was sarcastic or not sarcastic.

A. Multinomial Naive Bayes

Bayes' Theorem states that, for feature vector $(x_1, x_2, ..., x_m)$ and resulting class y, the following relationship holds:

$$P(y|x_1, x_2, ..., x_m) = \frac{P(y)P(x_1, x_2, ...x_m|y)}{P(x_1, x_2, ..., x_m)} \tag{1}$$

Under a naive Bayes classifier, we model $P(x_1, x_2, ..., x_m|y)$ with the assumption that each x_i is conditionally independent on y. Thus, we can simplify (1) to be:

$$P(y|x_1, x_2, ..., x_m) = \frac{P(y)\prod_{i=1}^{m} p(x_i|y)}{P(x_1, x_2, ..., x_m)} \tag{2}$$

After finding the prior distributions using maximum likelihood estimates, we simply choose the class $\hat{y}$ that gives the higher posterior probability in (2).

Multinomial naive Bayes, a variation on the naive Bayes algorithm above, is commonly used for text classification. It is parameterized by $(\theta_{y1}, \theta_{y2}, \theta_{y3}, ..., \theta_{yn})$ where θ_{yi} is the probability of feature i appearing in the class y. Specifically, *scikit-learn's* version of multinomial naive Bayes that we used further employed laplace smoothing, such that the parameters were calculated as:

$$\hat{\theta}_{yi} = \frac{N_{yi} + 1}{N_y + n} \tag{3}$$

where N_{yi} is the number of times feature i appeared in class y and N_y is the total count of features in y.

B. Logistic Regression

Under logistic regression, given a feature vector x, we use the following hypothesis function:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{4}$$

where g is the sigmoid function.

Following most linear classifiers, we set the hypothesis function as:

$$p(y|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y} \tag{5}$$

We then find the likelihood of the parameters θ as:

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}} \end{aligned} \tag{6}$$

We can then maximize the likelihood by maximizing the log likelihood:

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))\end{aligned} \tag{7}$$

In order to maximize the log likelihood, we use gradient ascent over a period of updates. Our gradient ascent update rule will then be given by:

$$\theta := \theta + \alpha \nabla_\theta \ell(\theta) \tag{8}$$

$\nabla_\theta \ell(\theta)$ simiplifies to:

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})) x^{(i)} \tag{9}$$

Then gradient ascent is performed until convergence to create the parameter vector θ. After training, tests can be performed by calculating the value of the hypothesis $h_\theta(x) = g(\theta^T x)$

C. Support Vector Machine (SVM)

We used *scikit-learn's* linear support vector classification algorithm, which is an implementation of linear support vector machine that scales better to large numbers of samples.

In support vector machine, we denote the class that each point x_i belongs to by y_i, which is either 1 or -1. SVM works by finding the maximum-margin hyperplane that divides the x_i's for which $y_i = 1$ from the x_i's for which $y_i = -1$. A hyperplane is a set of points x such that $w \cdot x - b = 0$, in which $w = [\theta_1 ... \theta_n]^T$ and $b = \theta_0$. We need to solve the dual optimization problem:

$$\max_\alpha W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \tag{10}$$

s.t. $0 \leq \alpha_i \leq C$, $i = 1, ..., m$ and $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$, where the a_i's are Lagrange multipliers and C controls the relative weighing between the goals of making the $||w||^2$ small and of ensuring that most examples have functional margins at least 1 in ℓ_1 regularization. In our case, $C = 1$, the default set by *scikit-learn*.

We then use the sequential minimal optimization (SMO) algorithm to solve the dual problem:

Repeat until convergence {
1) Select some pair α_i and α_j to update.
2) Reoptimize $W(\alpha)$ with respect to α_i and α_j, while holding the other α_k's fixed.

}

The decision function used by *scikit-learn* is:

$$\text{sgn}(\sum_{i=1}^{m} y_i \alpha_i \langle x_i, x \rangle + \rho) \tag{11}$$

where ρ is the intercept term.

V. Results and Discussion

A. Original Runs: Unbalanced Data Set

We partitioned 70% of our tweets into a training set, and 30% into a testing set. We then ran each of the three models on the two sets. To see that our models have learned from the features, we also ran them under the condition that each tweet had only one feature, randomly assigned to be 0 or 1.

Model Name	Accuracy	Random
Multinomial Naive Bayes	0.8250	0.7843
Logistic Regression	0.8247	0.7843
Linear SVM	0.7986	0.7843

Table 1: *Model ccuracy when splitting 70% of the samples into the training set and 30% into the test set*

Here multinomial naive Bayes and logistic regression were more effective than SVM. However, because the dataset was heavily unbalanced with far more non-sarcastic samples than sarcastic samples, the random classification rate was close to the model classification rates. To more closely evaluate the differences in the three models, we found the confusion matrix to report the false positive and false negative rates.

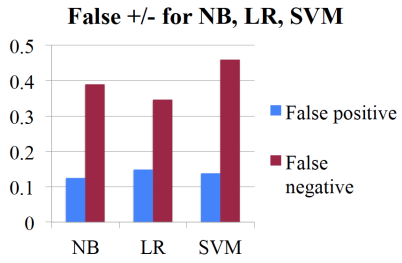

Figure 1: *False positive and negative rates for multinomial naive Bayes, logistic regression, and SVM*

For all three models, the false positive rate (non-sarcastic samples classified as sarcastic) was lower than the false negative rate (sarcastic samples labelled as non-sarcastic). We believe this may be due to the fact that the dataset was unbalanced, and took measures to address this issue later in the report (see Oversampling, Undersampling).

Furthermore, we found the precision recall-curves for each of the three models. Multinomial naive Bayes performed the best in terms of the precision/recall tradeoff, while SVM performed the worst. This result matches the accuracies presented in Table 1.

Model Name	Average Precision
Multinomial Naive Bayes	0.6236
Logistic Regression	0.5862
Linear SVM	0.5230

Table 2: *Average precision accuracy (AUC for PR curve) for each model*

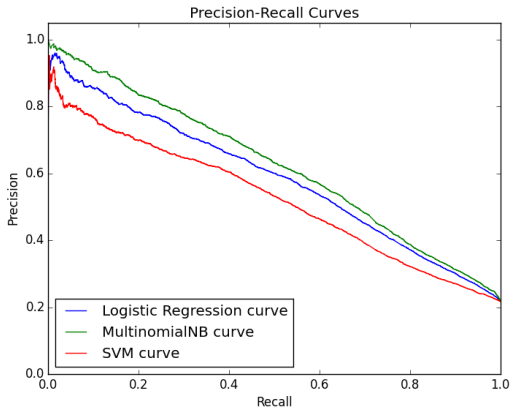

Figure 2: *Precision and recall curves*

We further performed 4-fold cross validation using *scikit-learn's* cross validation module on each of the three models. Here logistic regression performed better than multinomial naive Bayes, but the accuracies were relatively close.

Model Name	Accuracy
Multinomial Naive Bayes	0.8341
Logistic Regression	0.8475
Linear SVM	0.8284

Table 3: *Accuracy with 4-fold cross validation*

B. Oversampling

In order to achieve a more balanced data set, we implemented oversampling and undersampling. In oversampling, the samples belonging to the more scarce class are overrepresented in the final data set. Thus, we had each sarcastic tweet occur 4 times in the data set, so that both sarcastic and non-sarcastic sets had about 100,000 samples. Thus, the random classification rate was closer to 50%. We again split the final data set into 70% train and 30% test. Under the oversampled set, logistic regression had the highest accuracy.

Model Name	Accuracy	Random
Multinomial Naive Bayes	0.7807	0.5246
Logistic Regression	0.8027	0.5246
Linear SVM	0.7997	0.5246

Table 4: *Accuracy with an oversampled data set*

We again found the confusion matrix to analyze false positive and false negative rates.

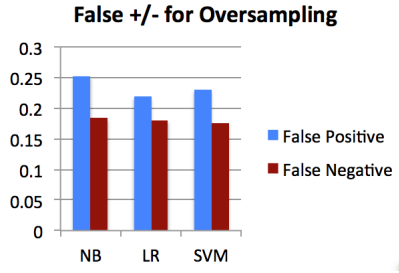

Figure 3: *False positive/negative rates for oversampled data*

The false negative rate was significantly lower, which indicates that the training phase may be more effective with a balanced data set because the classifier was no longer automatically classifying the samples as not sarcastic. The false positive rate was slightly higher, because more samples were now being classified as sarcastic.

Finally, we found the precision-recall curves under the new data set. Both the precision and the recall improved when training on the oversampled set. Among the three curves, multinomial naive Bayes continued to have the highest average precision while SVM performed relatively poorly.

Model Name	Average Precision
Multinomial Naive Bayes	0.8855
Logistic Regression	0.8797
Linear SVM	0.8549

Table 5: *Average precision accuracy (AUC for PR curve) for each model with an oversampled data set*

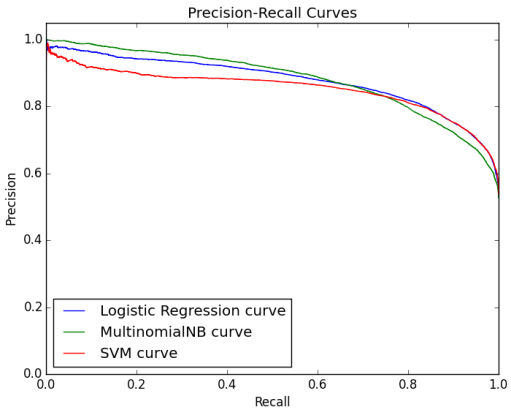

Figure 4: *Precision-recall curves for oversampled data set*

C. Undersampling

Undersampling involved taking only $\frac{1}{4}$th of the non-sarcastic tweets for the testing and training sets each time so that both sarcastic and non-sarcastic sets had around 25,000 tweets. We performed undersampling 4 times (each with a different quarter of the non-sarcastic set) and averaged the accuracies in Table 6.

Model Name	Accuracy	Random
Multinomial Naive Bayes	0.7308	0.5225
Logistic Regression	0.7235	0.5225
Linear SVM	0.6955	0.5225

Table 6: *Accuracy with an undersampled data set*

Similar to the unbalanced data set, multinomial naive Bayes and logistic regression were most effective while SVM performed more poorly. Overall, comparing the unbalanced, oversampled, and undersampled sets, the oversampled set had the greatest improvement in classification accuracy over the random classifier (Figure 5). The oversampled set may have performed better than the undersampled set because it

encompassed more of the non-sarcastic examples during the training phase.

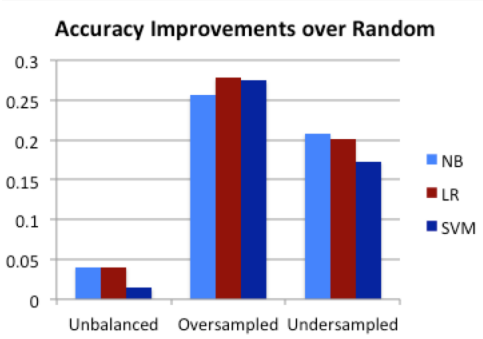

Figure 5: *Difference between random and model accuracies for unbalanced, oversampled, and undersampled data sets*

D. Feature Analysis

To analyze the impact of each feature on the classifier, we found the accuracy of the classifiers depending on each feature alone when trained and tested on the oversampled data set. Table 7 contains the average single feature accuracies over the three models

Feature Name	Accuracy
Unigram	0.7033
Bigram	0.7759
Capitalization	0.5247
Sentiment Split	0.5249
Subjectivity	0.5242
Random	0.5246

Table 7: *Accuracy of classifier with single feature averaged over multinomial naive Bayes, logistic regression, and SVM*

As shown, unigrams and bigrams were the most effective in classifying a tweet as sarcastic or not sarcastic, while capitalization and sentiment split achieved results that were only slightly better than those obtained by random feature.

VI. Conclusion and Future Work

Of the three models, logistic regression was the most effective at classifying sarcastic tweets, with an accuracy of 0.8027 on an oversampled set. SVM generally performed more poorly; this may be because there is not a large enough margin between the two classes for a sufficient linear hyperplane. Multinomial naive Bayes performed well overall, and even performed better than logistic regression in the unbalanced and undersampled sets. However, it makes the underlying assumption that each of the features are conditionally independent, but unigrams and bigrams are dependent by definition. Thus, multinomial naive Bayes may be overweighting the importance of certain unigrams and bigrams. Of our features, bigrams were the most significant in performing a correct estimate, with a single feature accuracy of 0.7759 on an oversampled set.

Currently, there are some limitations to our data set. Due to our methodology of selecting non-sarcastic tweets based on emotion-related hashtags, we have no non-sarcastic tweets that express neutral emotion. In the future, we can try the Liebrecht et. al's proposed method of obtaining non-sarcastic tweets, which involves simply taking tweets that do not have the tag "#sarcastic" and accepting the resulting noise.

In order to provide a reasonable classifier, we created a data set with comparable numbers of sarcastic and non-sarcastic tweets. However, sarcasm is relatively rare, so these sets do not represent the proportion of sarcastic tweets in real life. In the future, we can explore other classifiers that handle unbalanced sets more appropriately.

Finally, sarcasm is often based on current events. We polled sarcastic and non-sarcastic tweets over a period of four weeks. Ideally, however, tweets should be pulled over a longer period of time to provide a larger and more unbiased corpus of tweets.

References

[1] Bamman, David and Noah A. Smith. "Contextualized Sarcasm Detection on Twitter." *Association for the Advancement of Artificial Intelligence* (2015): 574-578.

[2] Cliche, Mathieu, Ph.D. *The Sarcasm Detector*. N.p., n.d. Web.

[3] De Smedt, Tom. and Walter Daelemans. "Pattern for Python." *Journal of Machine Learning Research* (2012): 20312035.

[4] Gonzalez-Ibanez, Roberto, Smaranda Muresan and Nina Wacholder. "Identifying Sarcasm in Twitter: A Closer Look." *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics* (2011): 581-586.

[5] Liebrecht, Christine, Florian Kunneman, and Antal Van Den Bosch. "The Perfect Solution for Detecting Sarcasm in Tweets #not." *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (2013): 29-37.

[6] Loria, Steven, Pete Keen, Matthew Honnibal, Roman Yankovsky, David Karesh, Evan Dempsey, Wesley Childs, Jeff Schnurr, Adel Qalieh, Lage Ragnarsson, and Jonathon Coe. TextBlob, version v0.11.0. Available at https://textblob.readthedocs.org

[7] Pedregosa, Fabian, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and douard Duchesnay. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* (2011): 2825-2830.

[8] Riloff, Ellen, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. "Sarcasm as Contrast between Positive Sentiment and Negative Situation." *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (2013): 704-714.