ISSN:0975-9646

Sarcasm Detection in Amazon Product Reviews

Sahil Jain[#], Ashish Ranjan[#], Dipali Baviskar[*]

[#]*Student at Department of Computer Engineering, Maharashtra Institute of Technology, Pune*

[*]*Faculty at Department of Computer Engineering, Maharashtra Institute of Technology, Pune*

***Abstract*— Sarcasm detection: Sarcasm is defined as the use of irony to mock or convey contempt. It is a feature of natural language where the literal meaning of one's language is quite opposite of the implied meaning. On e-commerce websites such as amazon, many times customers make the use of sarcasm in their reviews in an attempt to criticize the product. With the help of sarcasm detection, products can be classified into the relevant categories with more accuracy. Methodology: A corpus of ironic and regular reviews is used for the purpose of this experiment. The data is extracted into python dictionaries using the module such as beautiful soup. Sentiment features, lexical features and parts of speech features are extracted from the training and testing data using NLTK and TextBlob. Various classifiers are trained with the features of the training set. The results are calculated based on the predictions of the testing set. Results: Reviews from the testing set are classified as sarcastic/ironic or regular. Based on the observations, the accuracy, precision, recall and f-score are calculated. Hence, the performance of our experimental setup is evaluated. Applications: Opinion mining, recommendations and advertisement systems.**

Keywords— ***Irony, Natural Language Processing, Opinion Mining, Sarcasm, Sarcasm Detection, Sentiment Analysis, SVM Introduction***

I. INTRODUCTION

Sarcasm is defined as *"the use of irony to mock or convey contempt"* by the Oxford dictionary. Irony itself is defined as "the expression of one's meaning by using language that normally signifies the opposite, typically for humorous or emphatic effect". When an upset or angry person suddenly exclaims during a conversation that *"I am so happy that I feel like dancing!"* obviously is making use of sarcasm. The sentence is very positive, however, in the given context is used to express frustration and anger. Hence, a sentence can be a sarcastic or a regular sentence based on the context of usage.

Customers of e-commerce website like amazon may write a negative review of a product if they are not happy with the product's price, quality or service of the dealer. The review can be regular one if it is written in the straight-forward way. The customer can also make use of sarcasm in the review to mock or criticize the product. Suppose a customer buys a very costly headphone and it turns out the quality of sound is not very good. He then writes a review as *"the speaker is so good that I can hear sounds from other galaxies! Totally worth the money"* The user is obviously not satisfied with the product and is mocking the product's quality and price using sarcasm.

Detection of sarcasm in language in the form of text has been a challenging problem. This is because a sarcastic text might appear as a regular text. Using traditional opinion mining techniques can lead to incorrect classification of the reviews. A sarcastic sentence written with the view of expressing contempt might contain abundant positive words. Humans can figure out sarcasm easily when they know the tone of the sentence and the context in which it is used. We can thus improve the performance of a sarcasm detection system if we use contextual features along with traditional features to train our model. A sarcastic review written in the view to mock a product will have low ratings. This can indicate the context with which the customer has written the review. We make use of similar features in our experiment and show the improved performance of the system.

II. RELATED WORK

Recently, a lot of research work has been done to improve the performance of sarcasm detection in text and natural language. Though most of the experiments were performed on a dataset of tweets from the popular social media website twitter, many of the techniques can be applied on amazon product reviews as well. There are a few features which are frequently used with twitter database, but become irrelevant or less useful when we are working with amazon product reviews. We see that most of the work is done to select and extract appropriate features which will help to improve the accuracy. The most frequently used classification approaches are using SVM and Naïve Bayes classifier.

In [1], the experimental setup makes use of the twitter data-set. The model used for classification is a LinearSVC and a Naïve Bayes classifier. Features such as n-grams, sentiments, parts of speech, capitalization and the topics of the tweet are used. The classifier works with high accuracy on non-sarcastic tweets but with lesser accuracy on sarcastic tweets. The recall using of the Naïve Bayes classifier was 56%.

In [2], the used of context based features is highly emphasized. Along with the conventional features, features which are related to the author, the audience and the environment were used. A significantly high accuracy, 85.91% being the highest, was achieved. Though such features aren't available with the amazon product review corpus, we make the use of ratings of the review as a context

In [3], the experiment is conducted both on twitter and product review dataset. A pattern based approach which used high-frequency and content words is used. The classification model makes use of the k-nearest neighbour method. An f-score of 0.83 on twitter corpus and 0.78 on the product review corpus was achieved.

We use the open resource known as the "Sarcasm Corpus" of amazon product reviews in [4]

III. DATASET AND PREPROCESSING

We use the open resource called the "Sarcasm Corpus" for the purpose of this experiment. The details of the corpus are explained by Filatova in [4]. The corpus contains hundreds of product reviews from the popular e-commerce website amazon and the reviews are labelled as ironic and regular. We can consider the ironic reviews as sarcastic reviews in this context. The reviews are properly formatted using the mark-up language. Each review contains fields such as the stars (ratings), title, date, author, product and the review. An example of a review present in the corpus is as follows

<STARS>5.0</STARS>
<TITLE>Liberals write your own book! </TITLE>
<DATE>October 2, 2007</DATE>
<AUTHOR>VRWC "Dittohead"</AUTHOR>
<PRODUCT>If Democrats Had Any Brains, They'd Be Republicans (Hardcover) </PRODUCT>
<REVIEW>
I did buy this book. Liberals will not buy this book, so their opinion is worthless. Besides most of them can't even read clearly enough to get the joke. Anger and frustration that Conservatives are clever and witty with a dash of smarta**, and they resort to reviewing a book that we know they wouldnt spend a penny on. Besides it would take too much of their welfare check to buy it! Quit trying to sabotage this book. Just go away.
</REVIEW>

We include three hundred sarcastic reviews and three hundred regular reviews in our training set. We include hundred sarcastic reviews and hundred regular reviews in our testing set. We first open the files in our python program with the help of the os module. Then, we use the BeautifulSoup module for python the extract the text between the respective tags from the formatted reviews. We then store this review in a python dictionary. A list of dictionary is generated each for sarcastic and regular reviews. Then, the training and the testing set is appropriately created using these lists. After building the dataset, we pre-process the datasets.

The sentiment score of each review is calculated using the Vader module available with the NLTK library module for python. The range of the score is between -1 to 1. 1 indicates that the text is highly positive, -1 indicates that the text is highly negative and 0 indicates that the text has a neutral sentiment. We normalize this score so that our score is in the range of 0 to 1. This is done for simplification of calculations. The rating of the product is given by the number of stars. It is on the scale of 1 to 5, 1 indicates that a very poor rating and 5 indicates a very high rating by the customer. We scale the rating to in the range of 0 to 1.

The text of the review is tokenized using sentence and word tokenizer available in the NLTK library. Parts of speech tagging for each of the sentences in a review are done. Punctuation marks for each review are also counted.

IV. FEATURE ENGINEERING

Feature extraction is perhaps the most crucial part of this experiment. Selecting the appropriate features can highly influence the performance of our classifier. In this experiment, we will use the conventionally used features which are used in most of the research work related to this topic. Along with that, we add a new feature which can indicate the context of the review based on the ratings given by the customer

1. Sentiment feature

Sentiment of a review can highly indicate the mood of the customer. Sarcastic language has a certain tone and this can be indicated with the help of the sentiment of the text. Perhaps, it is the most commonly used feature in most of the research papers ([1], [2], and [5]). We shall only use the sentiment of the entire review text for the purpose of our experiment.

The sentiment score of the entire text of a review is calculated while preprocessing the dataset. This score is added to the feature vectors of our reviews.

2. Punctuation features

Customers use punctuation marks such as exclamation marks and question marks to strongly express their view. Sarcastic text and regular text might follow a certain pattern with respect to the types and numbers of punctuation marks used. The use of punctuation marks as a feature is also explained in [5]. We thus use the following punctuation features

- Number of question marks (?)
- Number of exclamation marks (!)
- Number of full-stops (.)

These features are counted for each individual review. Then the ratio of the number of a punctuation mark with respect to total number of punctuation marks is calculated. These numbers are then inserted to the feature vector of our product reviews.

3. Parts of speech features

This is a pattern based feature and is another frequently used feature in much of the recent research work. We shall simply count the tags and use them as a feature. Its further use is demonstrated in [5].

We shall use the parts of speech tags which were extracted for each review before. Now we calculate the number of nouns, verbs, adverbs and adjectives for each review. Then we find the ratio of these numbers with the total number of words in the review. These ratios are then inserted into the feature vector. These ratios indicate the density of each individual part of speech, as mentioned in [2].

4. Word unigram and bigram features

We make a list of the most commonly occurring unigrams and bigrams in sarcastic and regular reviews. Then we test the occurrence of these in our reviews and insert it into our feature vector. This is another most commonly used feature in previous papers.

5. Contextual features

The importance of knowing the context in order to recognize sarcasm has been explained previously in this paper and also in [1], [2]. In [2], the context was derived from the details of the author and the audience. Since similar kind of data is not available in the product review corpus, we use other information to determine the context of the review. We will make the use of the rating/stars given by the customer to the product as well as the sentiment feature of the review text.

According to the definition of sarcasm, irony is used to mock or express contempt. Suppose a customer is not satisfied with a product he purchased, he will give poor ratings to the review. While writing the review, he might choose to make the use of sarcasm. Even though the user is mocking or criticizing the product intentionally for its poor quality, the sentiment of the overall could be predicted as positive by the computer since the text is sarcastic. Consider the following example

> Stars: 1.0
> Review: If you enjoy repeated phrases, overused metaphors, and capitalized words in the middle of a sentence for effect, then this is the book for you. Otherwise, rest assured that the title is really the best part of the book.

In the review above, the customer is clearly making the use of sarcasm in writing the review and is unhappy with the product. However, given the phrases such as “if you enjoy” and “the best part of the book” and also other aspects of the text of the review, the computer would detect a positive sentiment in the review text. Similarly, in case of a regular or non-sarcastic review, the customer would usually write the review in a straight-forward language. Here, the language of the text can be positive, negative or neutral according the rating given by the customer. Consider an example of a regular review

> Stars: 4.0
> Review: It’s bulkier than the apple one, but works just fine.

As we can see, the customer is satisfied with the product and has given a 4 star rating to the product. Similarly, the review text also expresses the opinion of the customer unambiguously and clearly. The text of the review has a neutral sentiment as it would be detected by the computer.

Here, we observe that the contrast between the rating of the review and the sentiment of the text is higher in case of a sarcastic review and lesser in case of a regular review. Generally, sarcastic reviews with lower ratings tend to have a review written with a positive sentence. On the other hand, the sentiment score of a regular review is usually close to the rating of the review. That means that a regular review which has poor ratings would have negative sentiment, higher ratings would have positive sentiment and average ratings would have a neutral sentiment in general. We scaled and normalized the sentiment score and rating of the review during pre-processing the dataset. Now, we take the absolute difference between these two parameters and insert it as a feature. Here, the customer rating gives us the context of the review.

V. EXPERIMENTAL SETUP

We have implemented our experiment using the python programming language and we have also used libraries and tools such as Sklearn, NLTK and TextBlob. NLTK and TextBlob are libraries which were used during the pre-processing and feature extraction from our dataset. Once we have extracted the required features, we use them to train and test our classifier. Sklearn is a scientific computing library which comes with high performance implementations of various classifiers such as support vector machines, naïve bayes classifier, random forest classifier and many different others. We use the implementations given int this library to calculate the results of our experiment. We train and test our dataset on multiple classifiers given below.

1. Naïve Bayes Classifier

We use an implementation of the Gaussian Naïve Bayes classifier provided with the sklearn library. The classifier uses a supervised learning algorithm based on the Bayes’ theorem. Every pair of feature is assumes to be independent. This classifier didn’t seem to show the best results in case of our experiment. We use the default value of every hyperparameter for the purpose of our experiment

2. Neural Network Classifier

We use an implementation of a neural network classifier called as the MLP or a multi-layer perceptron classifier provided with the sklearn library. Adam optimization is performed on the loss function. The value of the L2 penalty parameter is set as 0.0001 and the initial learning rate is 0.001. Maximum number of iterations is, by default, set to 200.

3. Support Vector Machine Classifier

An SVM maps the features of the two classes in an n-dimensional space, where n is the number of features we have extracted. Then, the SVM draws a hyper plane which best segregates the two classes. We use the SVC implementation provided into the sklearn library and use the default values of the hyperparameters such as penalty parameter (C=1), degree of polynomial kernel function (degree=3) and default value of gamma. We observed the best results with the default values in case of our experiment.

VI. RESULTS

We distribute our dataset into training set and testing set. Out of the total 800 reviews used, 600 reviews are used for training and 200 reviews are used for testing. Reviews from each group, which is sarcastic and regular are present in equal proportion in the training and testing sets. We extract the features of each review in the dataset and prepare the feature vectors. A feature vector is a list of floating point numbers which represent our extracted features. We then train our dataset on various classifiers mentioned above and observe the results. The results of our experiment are tabulated as follows

	Naïve Bayes	Neural Network	SVM
Accuracy	77.50%	81.00%	81.50%
Precision	82.35%	82.29%	82.47%
Recall	70.00%	79.00%	80.00%
F-Score	75.68%	80.61%	81.22%

We observe that the best performance is obtained using SVM classifier, followed by the neural network classifier and naïve bayes classifier. However, the performance of SVM and the neural network classifier is very similar. We successfully achieve an f-score above 80% using SVM and neural network classifier. This shows that our classifier classifies both sarcastic reviews and regular reviews with high accuracy. We also observe that the naïve bayes classifier tends to classify most of the reviews as sarcastic as it has high precision but a low recall score.

VII. CONCLUSIONS

We observe how difficult it is to differentiate a sarcastic sentence from a regular sentence given the similarity of the sentence structure. We also understand the importance of knowing the context in which a sentence is used in order to categorize a sentence as a sarcastic sentence or a regular sentence. We also infer that while conventional features such as sentiment of a sentence, parts of speech tags, word unigrams and bigrams and punctuation marks are helpful, but not enough to recognize sarcasm in text.

We showed how the use of user ratings of a product review can be used as a contextual feature in sarcasm detection. We also achieved a significant performance while categorizing product reviews into sarcastic and regular ones. Finally, we compared the performance of three different classifiers used in our experiment.

We conclude that sarcasm detection is still a very challenging task in natural language processing, but we can significantly improve the performance of our systems if we choose the appropriate features which clearly and distinctly indicate the presence of sarcasm in the language.

Sarcasm detection is an important part of opinion mining and sentiment analysis. With the help of knowing the opinion of the customer, enterprises can have the knowledge of the areas where they need to improve their product or service. Sarcasm detection in online product reviews can also help e-commerce websites to build better and more efficient algorithms for their recommendation and advertisement systems. In future, we will work on exploring different techniques to know the opinion, sentiment or emotion of the user and thus building more intelligent and smart systems.

REFERENCES

[1] Chu-Che Peng, Mohammad Lakis and Jan Wei Pan, *Detecting Sarcasm in Text: An Obvious Solution to a Trivial Problem*, 2015.

[2] David Bamman and Noah A. Smith, *Contextualized Sarcasm Detection on Twitter*, Proceedings of the Ninth International AAAI Conference on Web and Social Media.

[3] Davidov, D., Tsur, O., & Rappoport, *Semi-Supervised Recognition of Sarcastic Sentences in Twitter and Amazon*, 2010

[4] E Filatova, *Irony and Sarcasm: Corpus Generation and Analysis Using Crowdsourcing*, 2012.

[5] Mondher Bouazizi and Tomoaki Otsuki, *A Pattern-Based Approach for Sarcasm Detection on Twitter*.

[6] scikit-learn. Support vector machines, 2014. URL http://scikit-learn.org/stable/modules/svm.html.

[7] Rajadesingan, A.; Zafarani, R.; and Liu, H, *Sarcasm detection on Twitter: A behavioral modeling approach. In WSDM*, 2015