4.3.1 Data Extraction

To train an algorithm to detect sarcasm, we first need some data to train our algorithm on. Classification is a supervised learning exercise, which means we need to have some sentences labeled as sarcastic and sentences labeled as non-sarcastic so that our classifier can learn the difference between the two.

The idea here is to use the Twitter API to stream tweets with the label #sarcasm, these will be our sarcastic texts, and other tweets that don't have the label #sarcasm, these will be our non-sarcastic texts. To collect non-sarcastic data, we have extracted tweets containing hashtags '#amazing, #sad, #happy, news, #motivation, #art'.

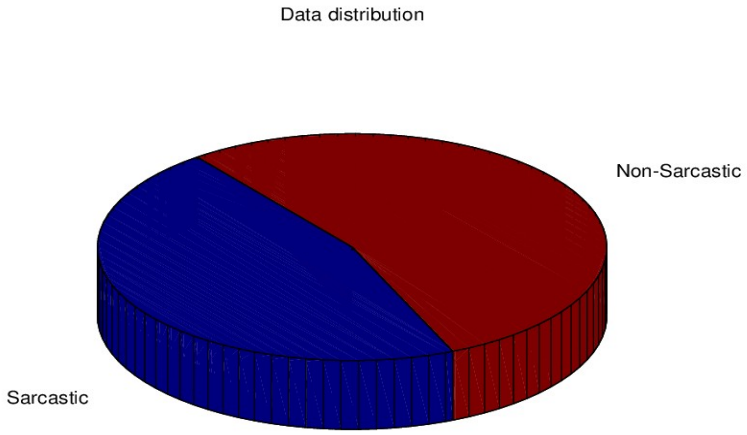

Fig. 5. Sarcastic and non-sarcastic data distribution

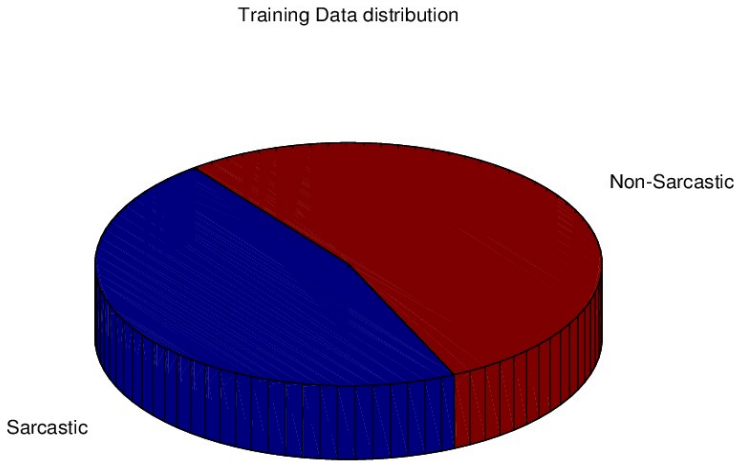

Fig. 6. Sarcastic and non-sarcastic training data distribution

We have extracted approximately 6000 tweets using Twitter API in python. Data is in the form of files and each file contain current tweet and past tweets of a user. The first tweet in the sarcastic files is the user's current tweet and rest 30 are user's past tweets. Fig. 5. shows the data distribution.Fig. 6. shows the training data distribution. Fig. 7. shows the test data distribution.

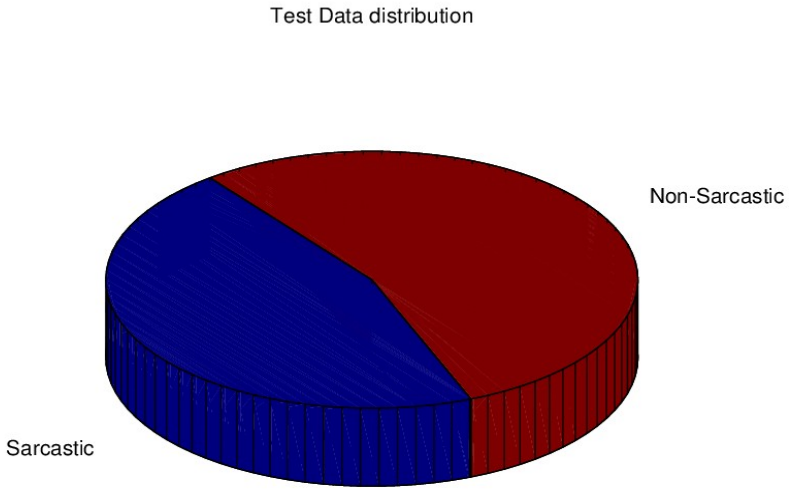

Fig. 7. Sarcastic and non-sarcastic test data distribution

4.3.2 Preprocessing the Data

Before extracting features from our text data it is important to clean it up. On Twitter or Facebook, the data is very noisy as it contains http links, hashtags, slang words, non-english tweets and non-imformative tweets. To process such words, we employed the preprocessing such as follow:

- Removed all non-english tweets manually.
- Removed all non-informative tweets manually.
- Removed http links.
- Remove #sarcasm and #sarcastic hashtags.
- Perform stemming.

If after this pruning stage the tweet is at least 3 words long, we add it to our dataset.

4.3.3 Feature Extraction

Features as described in previous section have been extracted by writing code in python and using

NLTK.

4.3.4 Classification:

- **Choosing a Classifier**

  There is a very wide range of machine learning algorithms to choose from, most of which are available in the python library Scikit-learn. However, most of the implementations of these algorithms do not accept sparse matrices as inputs, and since we have a large number of nominal features coming from our n-grams features it is imperative that we encode our features in a sparse matrix. We have employed SVM as the classifier. Past works also showed that SVM gives better results than other baseline algorithms.

- **Choosing Performance Indicator**

  The performance indicator metrics we are using are F1-measure, Accuracy, Precision and Recall as described:

  - **Precision:** It represents the fraction of retrieved sarcastic tweets that are relevant. In other words, it measures the number of tweets that have successfully been classified as sarcastic over the total number of tweets classified as sarcastic.
    Precision = True positive/ Total number of predicted positives
  - **Recall: I**t represents the fraction of relevant sarcastic tweets that are retrieved. In other words, it measures the number of tweets that have successfully been classified as sarcastic over the total number of sarcastic tweets.
    Recall = True positive/ Total number of actual positives
  - **Accuracy:** It represents the overall correctness of classification. In other words, it measures the fraction of all correctly classified instances over the total number of instances.
  - **F1-score** (also **F-score** or **F-measure**) is a measure of a test's accuracy. It considers both the precision *p* and the recall *r* of the test to compute the score: *p* is the number of correct positive results divided by the number of all positive results, and *r* is the number of correct positive results divided by the number of positive results that should have been returned. The F1-score can be interpreted as a weighted average of the precision

and recall, where an F1-score reaches its best value at 1 and worst at 0.The traditional F-measure is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

5. HARDWARE AND SOFTWARE REQUIREMENT

- **Hardware Used:** Desktop computer with Ubuntu 14.04 LTS

Here is a list of tools to be used in this project:

- **Data Collection Tools:**

  Twitter API: Twitter is an information network and communication mechanism that produces more than 200 million tweets a day. The Twitter platform offers access to that corpus of data through APIs. Each API represents a facet of Twitter, and allows developers to build upon and extend their applications in new and creative ways. The Twitter API is simply a set of URLs that take parameters. It let you access many features of Twitter, such as posting a tweet or finding tweets that contain a word, etc.

- **Language Used:**
  - **Python:** Python is a simple yet powerful programming language with excellent functionality for processing linguistic data. We chose Python because it has a shallow learning curve, its syntax and semantics are transparent, and it has good string-handling functionality. As an interpreted language, Python facilitates interactive exploration. As an object-oriented language, Python permits data and methods to be encapsulated and re-used easily. As a dynamic language, Python permits attributes to be added to objects on the fly, and permits variables to be typed dynamically, facilitating rapid development. Python comes with an extensive standard library, including components for graphical programming, numerical processing, and web connectivity. NLTK defines an infrastructure that can be used to build NLP programs in Python. It provides basic classes for representing data relevant to natural language processing; standard interfaces for performing tasks such as part-of-speech tagging, syntactic parsing, and text classification; and standard implementations for each

Precision and Recall (described in 4.3.4). By including all the features, we have been able to achieve:

Accuracy: 75.54%

F-measure: 0.756

Precision: 0.693

Recall: 0.833

7.1 TEST STATISTICS

Following test statistics are used for tests that detect the presence of sarcasm (as shown in Fig. 8):

- **True Positive:** A true positive test result is one that detects the condition when the condition is present. Predicted: 1, Actual: 1 --- True positive

- **True Negative:** A true negative test result is one that does not detect the condition when the condition is absent. Predicted: 0, Actual: 0 --- True negative

- **False Positive:** A false positive test result is one that detects the condition when the condition is absent. Predicted: 1, Actual: 0 --- False positive

- **False Negative:** A false negative test result is one that does not detect the condition when the condition is present. Predicted: 0, Actual, 1 --- False negative

		condition	
		present	absent
Test	positive	true positive	false positive
	negative	false negative	true negative

Fig. 8. Test Statistics

Fig. 9 illustrates the pictorial representation of test statistics achieved by our model.

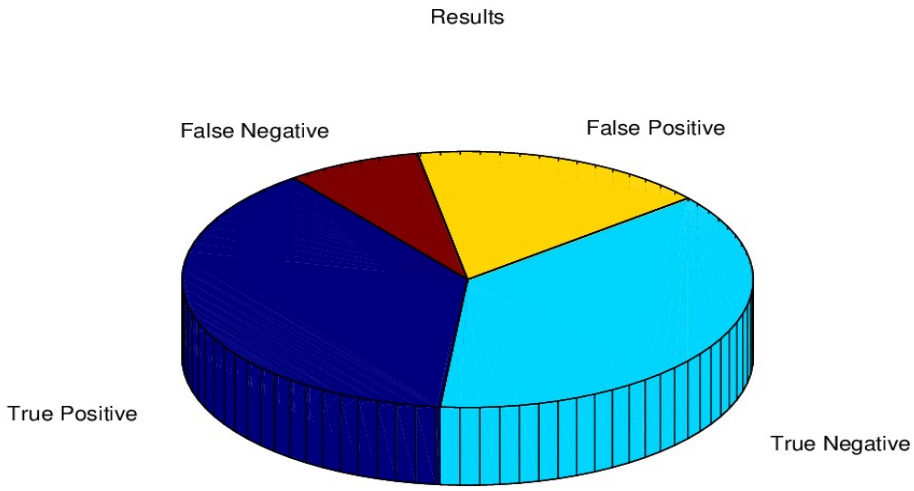

Fig. 9. Results(true positive, true negative, false positive and false negative)

7.2 COMPARISON WITH PREVIOUS RELATED WORK

Table 3 shows the comparison between previous proposed models and our proposed model. Davido et al. [3] use a semi-supervised approach on the binary sarcasm detection task using various feature types (punctuation: punctuation mark; patterns: patterns; enrich: after data enrichment; all features combined). They achieved an accuracy of 89.6%. The explanation of their excellent results is: SASI's robustness is achieved by the sparse match (α) and incomplete match (γ) that tolerate imperfect pattern matching and enable the use of variations of the patterns in the learned feature vector. SASI learns a model which spans a feature space with more than 300 dimensions. Their sparse and incomplete pattern matching with the vectors in the extended training set is responsible for excellent results. Lunando et al. [4] presented their sarcasm detection classifiers, including a Naive Bayes classifier and a Support Vector Machine, for analyzing Indonesian Social media, using features: unigram (negation, word context, affix), negativity, number of interjection words and question word. They achieved accuracy of 54.1% using SVM classifier. Their low accuracy result is due to following reasons: Sarcasm texts have no global topics and hence negativity feature is useless. They didn't include any context information and lot of text can be analysed if the reader know about the context or know about the writer. In our model we have included the writer's information by analysing his past tweets and we have included more number of features (59 features). Reyes et al. [5] used following featues: Signatures: concerning pointedness,