ISSN:0975-9646

An Improved Method for Detection of Satire from User-Generated Content

Syed Taha Owais[1] Prof . Tabrez Nafis[2] Seema Khanna[3]

[1] *Research Scholar, Jamia Hamdard, New Delhi*

[2] *Assistant Professor, Jamia Hamdard, New Delhi*

[3] *Technical Director – National Informatics Centre, New Delhi*

Abstract: Sarcasm is a form of speech act in which the speakers convey their message in an implicit way. It is a sophisticated form of speech act widely used in online communities. The inherently ambiguous nature of sarcasm sometimes makes it hard even for humans to decide whether an utterance is sarcastic in nature or not. Recognition of sarcasm may anticipate benefits in many sentiment analysis of NLP applications, such as safe search, review summary reports, engaging dialogue systems and review ranking applications and systems.

Classification of online news articles for satire has been very much done in the manual way. In our system, we have experimented with an automated approach to classify online news article using the SVM (Support Vector Machine) classification method. SVM has been shown to give good classification results when ample training documents are given. Obtaining the best results with SVMs requires an understanding of their workings and the various ways a user can influence their accuracy.

Keywords-Sarcasm, Satire, Irony, Sentiment Analysis, Supervised and Unsupervised Classification, Irony Detection, Algorithm, Machine Learning, Polarity, Opinion Mining, Data Corpus, Toenization,Support Vector Machine.

1. INTRODUCTION

Irony and satire can be useful weapons in any communicator‘s rhetorical arsenal. They provide a nuanced means for expressing critical sentiments and for openly exploring divisive subjects. However, the very subtlety that grants these devices their utility also lends to their greatest drawback: implied meanings are often lost on their intended audience. In textual communication this difficulty is magnified by the absence of any non-verbal cues that might imply a non-literal interpretation.

The goal of this project is to utilize machine learning strategies to develop a classifier for recognizing satirical or sarcastic web articles. Such content is, by definition, written to resemble more sincere communication and may be unrecognizable as ironic without sufficient contextual information. Therefore, in an attempt to capture broader ―context, the proposed classifier will rely not only on original, article texts, but also upon user-generated comments associated with each article.

2. RELATED WORKS & THEIR LIMITATIONS

While the use of irony and sarcasm is well studied from its linguistic and psycho logic aspects, automatic recognition of sarcasm is a challenging and emerging area of application in NLP, accomplished only by few researches. As far as the opinion mining is concerned, the sarcasm is considered as a hard nut that is yet to be cracked.

Tepperman et al. (2006) identify that sarcasm in verbal systems, their research is limited to sarcasm expressed in utterances that hold the expression 'Yeah right' and it depends heavily on cues in the spoken dialogue such as laughter, pauses within the speech stream, the gender (recognized by voice) of the speaker and prosodic features [1].

Tsur (2010) propose a semi supervised framework for recognition of sarcasm [2]. The proposed algorithm utilizes some features specific to (Amazon) product reviews. This paper continues this line, proposing SASI a robust algorithm that successfully captures sarcastic sentences in other, radically different, and domains such as twitter.

Utsumi (1996) introduces the implicit theory of display, a cognitive computational model that frameworks an ironic system [3]. The complex axiomatic system depends heavily on complex formalism representing world knowledge. While comprehensive, it is currently impractical to implement on a large scale or for an open domain.

Detailed work done by Paolo Rosso et al, in their work “Figurative Language Processing in Social Media for Human recognition and Irony detection” tries to bring out a linguistic-based framework for figurative language processing with special reference to emotion human like humor and irony in text generated in social media [4]. The author has suggested “Ambiguous-based pattern” using the concept of lexical, morphological, syntactic and semantic construction of the text in English language. The author emphasizes his models with examples and mathematical expressions of the model.

Lexical: Drugs may lead to nowhere, but at least it’s a scenic route.

Morphological: Customer: I’ll have two lamb chops, and make them lean, please.

Waiter: To which side, sir?

Syntactic: Parliament fighting inflation is like the Mafia fighting crime.

Semantic Jesus saves, and at present costs, which is a nothing but a miracle!

$$PP(W) = \sqrt[n]{1 \div P\{w1.w2.w3.....wn\}}$$

The author also considers the figurative language processing as a field of natural language processing. The

authors also focus on finding the formal element to computational process the figurative uses of natural language. The author makes detailed study of phonology, incongruity, semantics, similes, etc. However this study does not explain the subjective task and personal decisions.
Another study conducted by Po-Ya Angela using a corpus of 500 #irony and 500 #sarcasm tweets, have concluded that the sarcastic tweets use more positive words but ironic tweets use more neutral words [5].
E. Riloff and other have made a comparative study of positive and negative sentiments and a negative situation [6].

3. PROPOSED WORK

3.1 Data Corpus

Our satire corpus consists of a total of 2500 news wire documents and 110 satire news articles, split into fixed training and test sets as detailed in Table 1. The news wire documents were randomly sampled from the English Gigaword Corpus. The satire documents were selected to relate closely to at least one of the news wire documents by:

1. Randomly selecting a news wire document;
2. Hand-picking a key individual, institution or event from the selected document, and using it to formulate a phrasal query (e.g. Bill Clinton)
3. Using the query to issue a site-restricted query to the Google search engine; and
4. Manually filtering out ―non-newsy, irrelevant and overly offensive documents from the top-10 returned documents (i.e. documents not containing satire news articles, or containing satire articles which were not relevant to the original query).

Table- 1: Data Corpus Statistics

	Training	Test	Total
True	1800	700	2500
Satire	70	40	110

It is important to note that the number of satirical news articles in the corpus is significantly less than the number of true news-wire articles. This reflects an impressionistic view of the web: there is far more true news content than satirical news content. The corpus is novel to this research, and is publicly available for free downloading at: *http://www.csse.unimelb.edu.au/research/lt/resources/satire/.*

3.2 Method

3.2.1 Standard text classification approach

We take our starting point from topic-based text classification (Dumaise al., 1998; Joachims, 1998) and sentiment classification (Turney, 2002; Pang and Lee, 2008) [7], [8],[9] and [10].
State-of-the-art results in both fields have been achieved using support vector machines (SVMs) and bag-of-words features. We supplement the bag-of-words model with feature weighting, using the two methods described below.

Binary feature weights

Under this scheme all features are given the same weight, regardless of how many times they appear in each article. The topic and sentiment classification examples cited found binary features gave better performance than other alternatives.

Bi-normal separation feature scaling: BNS (Forman, 2008) has been shown to outperform other established feature representation schemes on a wide range of text classification tasks. This superiority is especially pronounced for collections with a low proportion of positive class instances. Under BNS, features are allocated a weight according to the formula:

$$|F^{-1}(tpr) - F^{-1}(fpr)|$$

where F^{-1} is the INCDF (inverse normal cumulative distribution function), tpr is the true positive rate (P(feature|positive class)) and fpr is the false positive rate (P(feature|negative class)).
BNS produces the highest weights for features that are strongly correlated with either the negative or positive class. Features that occur evenly across the training instances are given the lowest weight. This behavior is particularly helpful for features that correlate with the negative class in a negatively skewed classification task, so in our case BNS should assist the classifier in making use of features that identify true articles.
SVM classification is performed with SVMlight (Joachims, 1999) using a linear kernel and the default parameter settings.

3.2.2 Targeted lexical feature

This section describes three types of features intended to embody characteristics of satire news documents.

Headline features

Most of the articles in the corpus have a headline as their first line. To a human reader, the vast majority of the satire documents in our corpus are immediately recognizable as such from the headline alone, suggesting that our classifiers may get something out of having the headline contents explicitly identified in the feature vector. To this end, we add an additional feature for each unigram appearing on the first line of an article. In this way the heading tokens are represented twice: once in the overall set of unigrams in the article, and once in the set of heading unigrams.

Profanity

True news articles very occasionally include a verbal quote which contains offensive language, but in practically all other cases it is incumbent on journalists and editors to keep their language ―clean‖. A review of the corpus shows that this is not the case with satirical news, which occasionally uses profanity as a humorous device. Let P be a binary feature indicating whether or not an article contains profanity, as determined by the Regexp::Common::profanity Perl module given at: *http://search.cpan.org/perldoc?Regexp::Common::profanity*

Slang

As with profanity, it is intuitively true that true news articles tend to avoid slang. An impressionistic review of the corpus suggests that informal language is much more common to satirical articles. We measure the informality of an article as:

$$i \stackrel{\text{def}}{=} \frac{1}{|T|} \sum_{t \in T} s(t)$$

where T denotes the set of tokens (unigram) in the article

and s is a function taking the value 1 if the token has a dictionary definition marked as slang and 0 if it does not.
It is important to note that this measure of ―informality‖ is approximate at best. We do not attempt, e.g., to disambiguate the sense of individual word terms to tell whether the slang sense of a word is the one intended. Rather, we simply check to see if each word has a slang usage in Wiktionary.

3.3 Proposed Algorithm

Step 1—Text pre-processing:
Pre-processing of the text and representing each document as a feature vector.
Step 2—Feature Extraction:
Generating set of features by transforming input data.
Step 3—Training:
To train a classifier using a classification tool (e.g. LIBSVM).
Step 4—Classification:
Applying classifiers in new documents.

3.3.1 Text Pre-processing

3.3.1.1 Tokenization

Tokenization is the process of breaking a stream of text into symbols, phrases, words or other useful elements referred tokens. This token-list is inputed for further processing like parsing or mining of the text. Tokenization is very useful in both computer science and linguistics (where it is a form of text segmentation), where it forms part of lexical analysis.
Example:
Phrase: We are attending a tutorial now.
After Tokenization: ―we,―are, ―attending ,―a, ―tutorial, ―now .
For these operations, we used the Python nltklibrary**.**

3.3.1.2 Stop Word Removal

In computing, stop words are words which are filtered out prior to, or after, text processing. It is controlled by human input and is not automatic in nature. There is no list (definite) of stop words which uses all tools, if even used. Any group of words can be chosen as the stop words for a given purpose. But they consist of those words which are commonly used and not useful for text classification.
Example:
Remove words such as ―a, ―the, ―I, ―he, ―she, ―is, ―are, etc.
Removal of stop words is done using nltk, where English stop words are predefined.

3.3.1.3 Stemming

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form. In other words, it is the process to normalize words derived from the same root.
Example:
Attending― attend; Teacher― teach etc.
Stemming is done using Porter stemmer, which is a part of nltk.

3.3.2 Feature Extraction

Feature extraction involves simplifying the amount of resources required to describe a large set of data precisely. When the analysis of complex data is done, one of the major issues arises from the number of variables present. Analysis which includes a large number of parameters generally requires a large amount of computational memory and computation power or a classification algorithm which over-fits the training sample and generalizes poorly to new samples. Transforming the input data into the set of features is called *feature extraction.*
We have used unigram features, i.e. to use each word as a feature. We adopted and then implemented the following weighting approaches, like:

1. Binary feature weights (BIN)
2. Term Frequency - Inverse Document Frequency (TF-IDF)
3. Term Frequency - Bi-normal separation feature scaling (TF-BNS)
4. TF-IDF-BNS (Original work)

Binary feature weights (BIN)

Under this scheme all features are given the same weight, regardless of how many times they appear in each article. The topic and sentiment classification examples cited found binary features gave better performance than other alternatives.

Term Frequency - Inverse Document Frequency (TF-IDF)

Normalized is the most popular weighting schema for word frequency is '*tfidf*', given below:

$$tfidf(w) = tf \cdot \log\left(\frac{N}{df(w)}\right)$$

1. *tf(w)*–term frequency (number of word occurrences in a document)
2. *df(w)*–document frequency (number of documents containing the word)
3. *N*–number of all documents
4. *tfidf(w)*–relative importance of the word in the document

Term Frequency - Bi-normal separation feature scaling (TF-BNS)

Under TF-BNS, features are allocated a weight according to the formula:

$$\mathbf{TF*(F^{-1}(tpr) - F^{-1}(fpr))}$$

where F^{-1} is the INCDF (inverse normal cumulative distribution function), tpr is the true positive rate (P(feature|positive class)) and fpr is the false positive rate (P(feature|negative class)) and TF is term frequency.
BNS produces the highest weights for features that are strongly correlated with either the negative or positive class. Features that occur evenly across the training instances are given the lowest weight. This behavior is particularly helpful for features that correlate with the negative class in a negatively skewed classification task, so in our case BNS should assist the classifier in making use of features that identify true articles.

TF-IDF-BNS

Under TF-IDF-BNS, features are allocated a weight according to the formula:

(TF-BNS)* (IDF)

TF-BNS–Term Frequency Bi-Normal Separation feature scaling (Calculated earlier).

IDF–Inverse document frequency (Calculated earlier).

3.3.3 Training

We train a classifier using a classification tool, e.g. LIBSVM which is a SVM (Support Vector Machine) classifier.

3.3.3.1 Support Vector Machine

A support vector machine (SVM) is a concept in statistics and computer science and IT for a set of related SLM (supervised learning methods) that analyze data and recognize patterns, used for classification and regression analysis. For each given input the standard SVM takes a set of input data and predicts which of the two possible classes forms the input, making the SVM a *non-probabilistic binary linear classifier*.

Given a set of training examples, each marked as belonging to one of two categories, an SVM (support vector machine) training algorithm constructs a model that assigns new ways into one category or the other.

An SVM model is a representation of the examples as points in space, mapped so that the separate categories' examples are broken by a clear gap that is as wide as possible. New examples are then linked into that same space and predicted to join to a category based on which side of the introduced gap they fall on.

More formally, a SVM (support vector machine) forms a hyperplane or set of hyperplanes in a space (high or infinite-dimensional), which can be used for classification, regression, or other tasks.

Intuitively, the hyperplane achieves a good separation that has the longest distance to its nearest training point of data to any class.

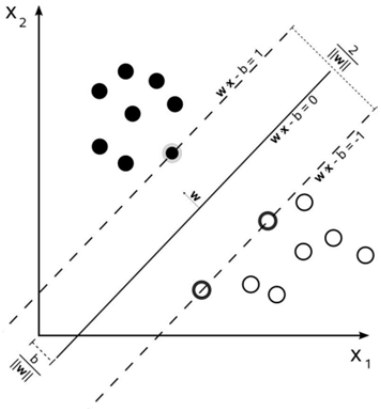

Figure-1: Hyperplane having maximum margin and SVM trained margins with samples.

3.3.3.2 Non-Linear Classification

The original optimal hyperplane algorithm proposed by Vapnik in 1963 was a linear classifier. However, according to, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik in 1992 suggested a way to create nonlinear classifiers by introducing the kernel trick (originally given and proposed by Aizerman et al. to hyperplanes having maximum margin.

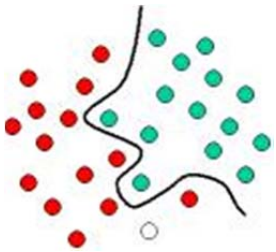

Figure-2: Classification (Non-Linear).

3.3.3.3 Kernel Trick

Kernel trick helps map your original data into a different space so that you can use linear classifiers. This mapping can often substantially increase the number of features to consider. This can be problematic as your number of dimensions grows. The *Kernel Trick* addresses this by putting a cap on the feature explosion so that the complexity of your classifier increases only linearly with the size of your original data.

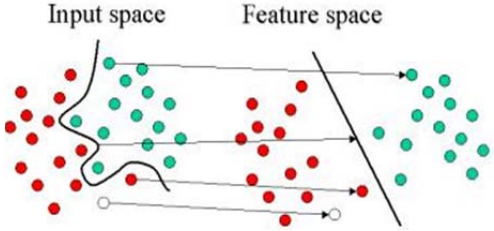

Figure-3: Kernel Trick

3.3.3.4 Classification

An algorithm that implements classification, especially in a concrete implementation, is known as a classifier. The term "classifier" sometimes also refers to the mathematical function, is implemented by a classification algorithm that maps input data to a category. In our case we used SVM Classifier (LIBSVM) for classification of Satire Articles.

The classification measure is given by some parameters. These parameters are as follows:

Accuracy: It is the proportion of true results (both true positives and true negatives) in the population. It is a parameter of the test.

$$\text{Accuracy} = \frac{\text{Number of true positives} + \text{Number of true negatives}}{\text{Number of true positives} + \text{False positives} + \text{False negatives} + \text{True negatives}}$$

Precision: It is the proportion of the true positives against all the positive results (both true positives and false positives).

$$\text{Precision} = \frac{tp}{tp + fp}$$

Recall: It is the proportion of the true positives against all the true results (both true positives and false negatives).

$$\text{Recall} = \frac{tp}{tp + fn}$$

F-Score: It 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.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Table-2: Classification

	Actual Class (Expectation)	
Predicted Class (Observation)	True Positive (tp)	False Positive (fp)
	False Negative (fn)	True Negative (tn)

4. RESULTS

4.1 Performance of the system

The baseline is a naïve classifier that assigns all instances to the positive class (i.e. SATIRE). An SVM classifier with simple binary unigram word features provides a standard text classification benchmark.

All of the classifiers easily outperform the baseline. This is to be expected given the low proportion of positive instances in the corpus. The benchmark classifier has very good precision, but a low recall. Adding the exaggeration, slang, and profanity features provides a small improvement in both precision and recall.

All of the classifiers achieve very high precision and considerably lower recall. Error analysis suggests that the reason for the lower recall is subtler satire articles, which require detailed knowledge of the individuals to be fully appreciated as satire. While they are not perfect, however, the classifiers achieve remarkably high performance given the superficiality of the features used.

Table- 3-:Results for Satire Detection

Weighing Parameters	Precision	Recall	F-Score
BIN	0.746	0.788	0.767
TF-IDF	0.722	0.732	0.727
BNS	0.802	0.859	0.829
TF-IDF-BNS	0.802	0.870	0.830

5. CONCLUSION

5.1 Choices made and Reasons

This project was implemented entirely in Python, because Python has inbuilt support for natural language processing using NLTK Library. It is efficient in performing complex operations easily with a few lines of code and has inbuilt functions for almost every trivial or non-trivial task.

5.2 Key Features of the project

We have done feature extraction of all the articles in the training set and assigned weights to all the words that remain after pre-processing task.

We assigned weights to the words according to the following weighing schemes:

1. Binary feature weights (BIN).
2. Term Frequency - Inverse Document Frequency (TF- IDF).
3. Term Frequency - Bi-normal separation feature scaling (TF-BNS).
4. TF-IDF-BNS (original work).

This research project has introduced a novel task to computational linguistics and machine learning: determining whether a news-wire article is ―true‖ or satirical. We found that the combination of SVMs with BNS feature scaling achieves high precision. Also, some notable mentions are as follows:

1. Our classification using various weighing technique provided varied result.
2. The best result of Satire Classification was given by TF-BNS weighing scheme.
3. Controlling the parameters of the SVM Classifier also helped in getting better result and we arrived at the best result by exhaustive experiments and trials with the SVM classifier.

FUTURE SCOPE

Lexical approaches are clearly inadequate if we assume that good satirical news articles tend to emulate real news in tone, style, and content, what is needed is an approach that captures the document semantics.

Semantic based classification may yield better result in the case of satire article detection, as dealing with the meaning of a word is better than dealing with the usage of the word.

As the successful use of satire relies heavily upon context and subtlety, methods that consider only whether a word was used and not how it is used may ultimately prove incapable of driving a highly effective classifier. Further research may need to explore more advanced language processing methods.

REFERENCES

[1] Joseph Tepperman, David Traum, and Shrikanth S. Narayanan. 2006. ”Yeah right”: Sarcasm recognition for spoken dialogue systems. In Proceedings of InterSpeech, pages 1838–1841, Pittsburgh, PA, USA, September.

[2] Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. ICWSM - A great catchy name: Semi-supervised recognition of sarcastic sentences in product reviews. In Proceeding of AAAI Conference on Weblogs and Social Media (ICWSM–10), Washington, DC, USA, May.

[3] Akira Utsumi. 1996. A unified theory of irony and its computational formalization. In Proceedings of the 16th conference on Computational linguistics (COLING 1996), pages 962–967, Stroudsburg, PA, USA. Association for Computational Linguistics.

[4] Paolo Rosso et al. 2012. March. Figurative Language Processing in Social Media for Human recognition and Irony detection, http://www.academia.edu/3032533/From_humor_recognition_to_irony_detection_The_figurative_language_of_social_media.

[5] Po-Ya-Angela Wang. 2013. #Irony or #Scarcas: A quantitative and qualitative study based on Twitter.

[6] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, Ruihong Huang. 2013. Sarcasm as Contrast between a Positive Sentiment and Negative Situation emnlp13-sarcasm.pdf.

[7] Dumaise al., 1998. Inductive Learning Algorithms and Representations for Text Categorization. http://robotics.stanford.edu/users/sahami/papers-dir/cikm98.pdf.

[8] Joachims, 1998. Text categorization with Support Vector Machines: Learning with Many Relevant Features, http://www.cs.iastate.edu/~jtian/cs573/Papers/Joachims-ECML-98.pdf.

[9] Peter D. Turney, 2002, Thumbs Up or Thumbs Down? Semantic Orientiation Applied to Unsupervised Classification of Reviews.

[10] Bo Pang, and Lillian Lee. 2008. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2): 1-135.