Irony Detection

Shikha Agarwal
UMass, Amherst
Department of Computer Science
shikhaagarwa@cs.umass.edu

Abstract

Irony is a common phenomenon in social media, and is inherently difficult to analyse, not just automatically but often for humans too. The aim of this project was to experiment with the different combinations of features, blending it with different machine learning models. The dataset and major source of encouragement behind this project comes from the sentiment analysis task present in SemEval 2018. The best performance model was Logistic Regression that achieved an accuracy in the range of 65-67% and f1-score in the range of 63-65% with different combination of linguistic features.

1. Introduction

The traditional definition of irony is saying the opposite of what you mean *et al.* [7](Quintilien and Butler, 1953). The development of the social web has stimulated usage of such figurative languages - irony and sarcasm. In particular, irony is a very interesting phenomenon as it exposes the problems that current machines have in detecting the intended rather than the literal meaning of a sentence. As such, modeling irony has a large potential for applications in various research areas, including text mining, author profiling, detecting online harassment, and perhaps one of the most popular applications at present, sentiment analysis *et al.* [4](Maynard and Greenwood, 2014; Reyes, Rosso, and Veale, 2013) and opinion mining *et al.* [6](Pang and Lee 2008). Affective computing (AC) is an AI field dealing with Human-Computer Interaction. It studies intelligent systems that are able to recognise, process and generate human emotions *et al.* [8](Picard, 1997). One of the applications that AC finds in NLP is detecting irony and sarcasms in its conversation with human.

In this project, I aim to experiment with the different combinations of features in the automatic detection of irony, blending it with different machine learning models. Also, a major source of encouragement behind this project comes from the sentiment analysis task present in SemEval 2018 *et al.* [5]. The problem can be defined as a binary classification task where the system has to predict whether a tweet is ironic or not. The following sentences present examples of an ironic and non-ironic tweet, respectively:

I just love when you test my patience!! #not
Had no sleep and have got school now #not happy

2. Related Work

In this section, I briefly discuss the approach followed by 3 recent works on Irony/Sarcasm detection:

2.1. Modelling Irony in Twitter: Feature Analysis and Evaluation - Francesco Barbieri, Horacio Saggion

In this paper *et al.* [9], the author uses some dependant features on Decision tree model over independant features on Naive Bayes to classify ironic sentences, aiming to detect inner characteristic of Irony. The paper states that the model uses seven groups of features to represent each tweet. Some of which are designed to detect imbalance and unexpectedness, others to detect common patterns in the structure of the ironic tweets. These features are as following:

1. Frequency (gap between rare and common words)
2. Written-Spoken (written-spoken style uses)
3. Intensity (intensity of adverbs and adjectives)
4. Structure (length, punctuation, emoticons)
5. Sentiments (gap between positive and negative terms)
6. Synonyms (common vs. rare synonyms use)
7. Ambiguity (measure of possible ambiguities)

2.2. Clues for Detecting Irony in User-Generated Contents: Oh...!! It?s ?so easy" ;-) - Carvalho, P., Sarmento, S., Silva, M. J., and de Oliveira, E. 2009

The paper *et al.* [10] explores linguistic clues associated with the expression of irony in Portuguese in sentences containing positive predicates. In video/spoken discourse, especially in a conversational context, one detects a variety of external clues (e.g. facial expression, intonation, pause duration) that enable the perception of irony. The author then discusses that in written text, a set of more or less explicit linguistic strategies can be used to express irony. It describes the use of following patterns in detecting of irony particularly in positive Portuguese sentences:

1. Diminutive Forms
2. Demonstrative determiners
3. Interjections
4. Verb Morphology
5. Cross-constructions
6. Heavy Punctuation
7. Quotation Marks
8. Laughter Expressions

Although, many of the patterns are specific to Portuguese language, patterns like Laughter Expressions, Punctuation and Quotation marks are general to almost every language.

2.3. Identifying Sarcasm in Twitter: A Closer Look - Gonzalez-Ib a nez, Roberto, Muresan, Smaranda, and Wa- cholder, Nina.(2011)

Sarcasm transforms the polarity of an apparently positive or negative utterance into its opposite. The paper *et al.* [11] studies the use of lexical and pragmatic factors to distinguish sarcasm from positive and negative sentiments expressed in Twitter messages. Following are the two factors reported in the paper:

1. Lexical Factors: For this, they used two kinds of lexical features - unigrams and dictionary based.
2. Pragmatic Factors: They used three pragmatic features - positive emoticons, negative emoticons, and ToUser (which marks if a tweets is a reply to another tweet)

2.4. Glove: Global Vectors for Word Representation - Manning, C.D., Pennington, J., & Socher, R. (2014)

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus. The result of the paper is a new global log-bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods. The performance of model is 75% and it produces a vector space with meaningful substructure. It also outperforms related models on similarity tasks and named entity recognition. I have used the model's pre-trained word vectors for my experiments in this report.

2.5. Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model - Jin Wang, Liang-Chih Yu, K. Robert Lai, Xue-Jie Zhang (2016)

LSTMS have been used in the past to perform sentiment analysis of texts. In paper *et al.* [12] the authors have proposed a regional CNN-LSTM model consisting of two parts: regional CNN and LSTM to predict the VA(valencearousal) ratings of texts. Unlike a conventional CNN which considers a whole text as input, the proposed regional CNN uses an individual sentence as a region, dividing an input text into several regions such that the useful affective information in each region can be extracted and weighted according to their contribution to the VA prediction. Such regional information is sequentially integrated across regions using LSTM for VA prediction.

2.6. Automatic Sarcasm Detection: A Survey

This article is a compilation of past work in automatic sarcasm detection. In this work *et al.* [2] a compilation has been made of the previous work done in sarcasm detection. They have observed three milestones in the research so far: semi-supervised pattern extraction to identify implicit sentiment, use of hashtag-based supervision, and incorporation of context beyond target text

3. Dataset

Twitter provides functionality to users to summarize their intention via hashtags. To facilitate data collection and annotation, many supervised-learning approaches rely on hashtag-labeled (e.g. #sarcasm) Twitter data. The dataset used for the experiments reported in this paper has been prepared by SemEval2018 competition. They collected the tweets for automatic irony detection using the hashtags #irony, #sarcasm and #not between 01/12/2014 and 04/01/2015 and represent 2,676 unique users. To minimize the noise introduced by groundless irony-related hashtags, they manually annotated the corpus following a fine-grained annotation scheme (Van Hee et al., 2016c). Prior to data annotation, the entire corpus was cleaned by them by removing retweets, duplicates and non-English tweets, and replacement of XML-escaped characters (e.g., &). As a part of data annotation, they have provided a script that converts complex emoticons to their description enclosed within colon (e.g., ¡U+1F60A¿ to :smiling face with smiling eyes:) or vice versa. The script uses Python emoji module to make these conversions.

Approximately, 3834 instances of English tweets dataset is provided by the SemEval 2018 team. Out of 3834, 1911 instances are from class irony and 1923 from not irony class. The hashtags are removed from the tweets for the experiments.

4. Technical Approach

The data analysis, experiments with different features and models are described below:-

4.1. Data Analysis

Social media data contains many elements such as hashtags, profile references, urls. While analysing the dataset, I found that it contains twitter handle urls that provide no information to the tweet being irony. Therefore, I removed these urls as first task of data preprocessing.

I then analysed the length of tweets. I believe the effect of irony in a sentence is stronger when they are neither too short nor too long. For example, while analysing the dataset I found following sentences tagged as irony:

”Love this weather”
”Loving life”
”Wow I feel great”

Reading the above sentences alone (without any background), humans will never classify these sentences as irony. Similarly, longer sentences tagged ironic consists mostly the description of emoticons. For example,

”I love math, I’m so amped for my final like 7 hours from now. I hardly studied for it. #BeenDoingIntegralsSinceThe5thGrade :smiling_face_with_open_mouth_and_smiling_eyes::face_with_stuck_out_tongue_and_winking_eye::face_with_tears_of_joy::persevering_face::face_with_open_mouth_and_cold_sweat::tired_face::confounded_face:”

The average length of ironic tweet in the dataset was 15. Figure 1 shows the distribution of length of tweets across the dataset.

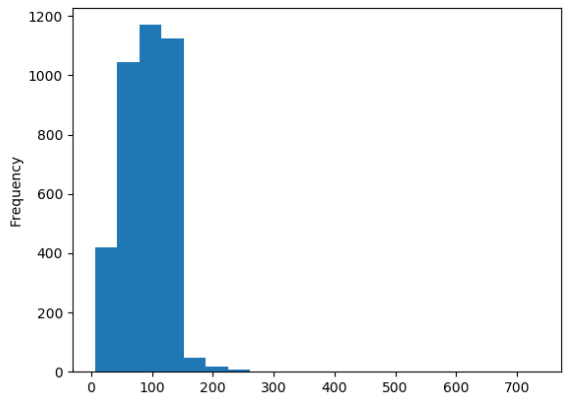

Figure 1. Count of sentences vs length of sentence

4.2. Features

Extracting important features from the given task is a very important step in machine learning. I identified and experimented with different features for different models. For baseline model Naive Bayes, I experimented with bag of words with unigram and bigram as features. Since, with Naive Bayes it is difficult to experiment with different kinds of features, therefore, I implemented Logistic Regression Classifier. With Logistic Regression, I experimented with various features like - length of tweets, parts-of-speech (POS) tagging, repeated characters (like multiple exclamation), usage of emoticons. Additionally, I experimented with Word2Vec feature in the Neural Network model LSTM.

4.3. Models

As mentioned in the above section 4.2, I experimented with three different models with different combination of feature sets to classify a tweet as irony. These models are:

1. Naive Bayes
2. Logistic Regression
3. LSTM

5. Experiments - Linguistic Features & Models

5.1. Naive Bayes

In this section, I will discuss the train/val/test splits, features, hyper-parameter tuning, and performance measure of Naive Bayes(NB) classifier. I have used Multinomial NB classifier.

5.1.1 Features

The NB classifier was trained with unigram and bigram features.

5.1.2 Tokenization and Normalization

I used BOW with unigram and bigram as feature for the baseline analysis. Sometimes, the dataset contains punctuations and delimiters that does not make sense. But I think for irony detection, punctuations could make sense. For example, ’,’ could have a sense of polarity in ironic sentences. So, I experimented with removing the punctuations and delimiters to analyse the model’s performance. I also experimented with term-frequency times inverse document-frequency (tf-idf) feature.

5.1.3 Train/Val/Test splits

I divided the datasets as 80% development data and 20% test data. The development data was further divided into

5-fold cross-validation. The classifier was re-trained with both the training data and validation data once the hyper-parameter tuning from validation set was done.

5.1.4 Performance Analysis

The Table 1 below represents the performance metrics(accuracy, precision, recall, and f1-score) of Naive Bayes for different experiments with different feature sets. First row represents experimentation with only bow on unigram and bigram as baseline. The second row is the performance metrics after removing punctuations/delimiters on baseline features. We see that there is not much difference in the performance of model on removing the punctuations from tweets. The third row shows analysis after including tf-idf on baseline. Adding this, decreases all the 4 metric values. This is because the tweets in the datasets are very small (avg of 90 characters long including descriptions of emoticons). This short texts are likely to have noisy tf-idf values and hence, decreasing the performance of the model.

Figure-2 below represents the confusion matrix for baseline - Naive Bayes with bow on unigram and bigram.

Case	Accuracy	Precision	Recall	F1-Score
NB	60.6	57.8	67.1	62.1
NB - punc	60.6	57.8	67.7	62.3
NB + tf-idf	58.8	56.6	61.4	58.9

Table 1. Performance Analysis for Naive Bayes

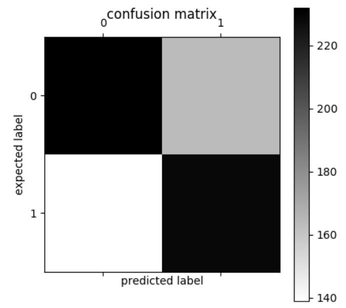

Figure 2. Confusion Matrix for Baseline

5.2. Logistic Regression

The accuracies achieved with Naive Bayes was approximately 60%. With Naive Bayes, it becomes difficult to experiment with different hand-crafted features. Hence, I chose Logistic Regression classifier and in this section I will discuss the different hand-crafted features that I used to train the model. With Logistic Regression, I achieved a better accuracy and f1-score compared to the baseline model - Naive Bayes.

5.2.1 Features

Along with bow on unigram and bigram, I experimented with four different features - word2vec, usage of emoticons, POS tags, count of exclamation marks and length of the tweet.

For word2vec feature, I used GloVe - Global Vectors for Word Representations. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. The results representation showcase interesting linear substructures of the word vector space. I experimented with pre-trained GloVe word vectors.

Since using irony in the language usually signifies the opposite of what it intends to mean, people use emoticons like wink ;) in their tweets/sentence. Also ironic sentences are typically used for humorous or emphatic effects, people tend to use laughing emoticons such as :D or other complex emoticons. There are two kinds of emoticons in the twitter dataset - simple emoticons like [:-), :D, ;-)] and complex emoticons with its descriptions enclosed within a colon like :face_with_tears_of_joy:. This is explained in section 3 of the report. I wrote a script to extract both simple and complex emoticons from the tweets. The validation of this script was done by manually investigating the correctness of the extracted emoticons after saving it in a file.

Additionally, I experimented with POS tags feature from nltk and some hand-crafted features like presence of repeated characters like count of multiple exclamation marks.

5.2.2 Tokenization and Normalization

First, I used nltk to perform normal word tokenization. I didn?t remove any stop words, or punctuations/delimiter since every token carries some meaning. Consequently, text normalization step just performed lower-casing of tokens, lemmatization and no word filtering. This method of tokenization and normalization resulted in splitting at every punctuation. For example, tokenization result on a sample sentence -
”Oh how I missed middle school drama :) :D” is:
[’Oh’, ’how’, ’I’, ’missed’, ’middle’, ’school’, ’drama’, ’:’, ’-’, ’)’, ’:’, ’D’]
On analysing the above ironic sentence with such simple tokenization, I realized that it splits on hashtags as well as emoticons. Thus, not being a good tokenizer for tweets. TweetTokenizer from nltk is a twitter-aware tokenizer that is specially designed to not tokenize on hashtags and emoticons. The output of TweetTokenizer on the above example looks like following:

[u'Oh', u'how', u'I', u'missed', u'middle', u'school', u'drama', u':-)', u':D']

5.2.3 Train/Val/Test splits

I divided the datasets as 80% development data and 20% test data. The development data was further divided into 5-fold cross-validation. The classifier was re-trained with both the training data and validation data once the hyper-parameter tuning from validation set was done.

5.2.4 Performance Analysis

The Table 2 below represents the performance metrics(accuracy, precision, recall, and f1-score) of Logistic Regression for different experiments with different feature sets. First row represents experimentation with only bow on unigram and bigram as baseline. The second row is the performance metrics after Lemmatization on baseline features. Performance significantly increases after adding lemmatization.

The third row tests with the feature length of tweets and we find no change in performance. Figure - 3 shows the distribution of length across the label 0 - Not irony and figure - 4 shows the distribution of length across the label 1 - Irony. On analyzing this graph, I find there is not much difference in the distribution across the different labels (except very few longer texts for label 0). Hence, length of text is not that of a good feature in detecting irony for this dataset.

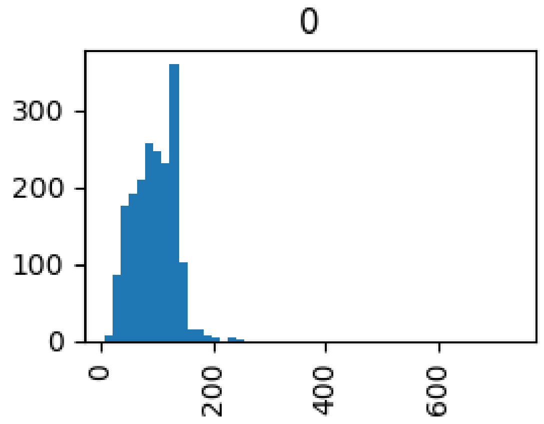

Figure 3. Distribution of length of sentence across label 0 - Not Irony

Fourth row of Table-2 presents performance with POS tagger. I have used nltk POS tagger for this. And the fifth row presents performance of different feature sets - lemmatization, length of text, POS tagger and one custom feature count of exclamation marks.

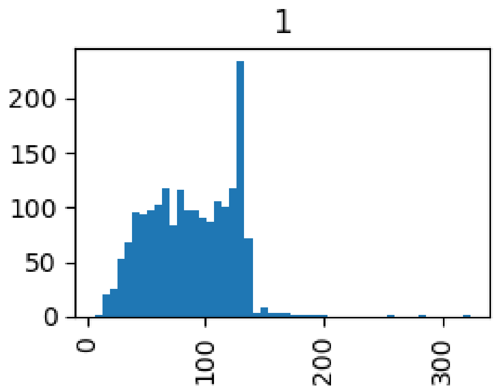

Figure 4. Distribution of length of sentence across label 1 - Irony

The sixth row, f5 - feature 5 includes usage of emoticons as feature along with length of text, POS tagger and one custom feature count of exclamation marks. The performance is similar t o that of other models. The seventh row, f6 - includes usage of emoticons combined with word2vec along with length of text, POS tagger and one custom feature count of exclamation marks. The performance metrics for all the different sets of feature is almost similar.

Figure-5 shows confusion matrix for Logistic Regression with bow (unigram and bigram), lemma, length of text, POS tags, and count of exclamation as feature set.

Table 2. Performance Analysis for Logistic Regression

Case	Accuracy	Precision	Recall	F1-Score
LR	62.0	60.8	59.8	60.3
LR + f1	66.3	65.6	63.3	64.5
LR + f2	65.3	64.3	62.8	63.5
LR + f3	65.6	65.0	61.7	63.3
LR + f4	65.7	65.0	61.6	63.3
LR + f5	65.1	64.0	63.3	63.6
LR + f6	65.7	64.4	64.1	64.3

5.3. LSTM

The Logistic Regression classifier performed much better than Naive Bayes. The LR model requires a lot of feature engineering and thus one needs to have a lot of domain knowledge in linguistics. For example, I have experimented with some of the structural features of a ironic sentence but one could add other sentimental features (like polarity) of ironic sentence to the model. This requires some knowledge on how ironic sentences are framed. Since, deep neural network extracts and learns these features, one does not require a strong domain knowledge in linguistics. So, I decided to experiment with one of the Neural Network model for this

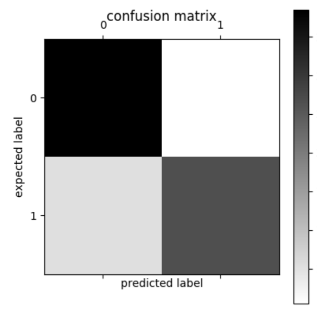

Figure 5. Confusion Matrix for LR

task - LSTM.

Long short-term Memory (LSTM) is a special type of Recurrent Neural Network (RNN) that is capable of learning long-term dependencies. RNN have loops which allow the network to use information from previous passes, acting as a memory. LSTM solves the problem of RNN not being able to learn to connect the information of long sentences. Thus, the ability of LSTMs to remember information for a long period of time finds application in NLP tasks as each word in a sentence depends greatly on what came before and comes after it.

I experimented with the LSTM architecture for detecting irony in a sentence using TensorFlow. To decide the number of LSTM unit cells, I looked at the average number of words in a sentence in the given twitter dataset - which was 15. Thus, I kept the maximum length of input as 25 and number of lstm unit cells to 12. If the sentence contains fewer words than 25, it would be assigned a vector of zeros.

5.3.1 Features

Just like Logistic Regression requires quantifiable features as intput to the model, LSTM inputs need to be scalar values or matrices of scalar values. If input to the LSTM is given in the form of raw sentence, it would be difficult to perform operations such as dot product, backpropagation on string. Thus, each word in the sentence is converted into a vector known as word embedding.

For this word2vec feature, I used GloVe - Global Vectors for Word Representations. As described in section 5.2.1, GloVe is an unsupervised learning algorithm for obtaining vector representations for words. The results representation showcase interesting linear substructures of the word vector space. I experimented with pre-trained GloVe word vectors.

5.3.2 Train/Val/Test splits

I divided the datasets as 80% development data and 20% test data. The development data was further divided into 5-fold cross-validation. The classifier was re-trained with both the training data and validation data once the hyper-parameter tuning from validation set was done.

5.3.3 Performance Analysis

The accuracy of LSTM model on the test data was 60.3%. The performance of this model was not up to the mark. One of the reason LSTM did not perform well as expected could be improper fine-tuning of the model due to time constraint.

6. Conclusion

The human performance for detecting irony in a sentence is 62%. With Naive Bayes experiment and simple features like unigrams and bigrams, I was able to achieve an accuracy similar to that of human performance. With more advanced linguistic features and Logistic Regression classifier, I have achieved accuracy in the range of 65-67% and f1-score in the range of 63-65% with different combination of features. I believe implementing more such features specific to the problem like the presence of polarity in the sentence, the model will have better performance results.

I would also like to mention here that when analysing some tweets labelled as irony I found that the dataset contained many ambiguous statements. Some of such ambiguous tweets are listed below:
I love you mom
We want turkey!!
Beautiful day
No human being would detect these tweets to be ironic.

7. Future Works

My initial observations is that there are many interesting phenomena to be observed to detect irony, and that structural features of a sentence, while useful, is not sufficient for accurate irony detection of such tweets. There are different types of irony. One such is polarity contrast, where sentences containing an evaluative expression whose polarity (positive, negative) is inverted between the literal and the intended evaluation. For example:
”I really love this year’s summer; weeks and weeks of awful weather.”
The first part of sentence contributes to the positive sentiment (”love”) whereas the second part is inverted to negative sentiment by usage of word ”awful”. We can leverage this polarity inversion as a feature to detect irony in a sentence as a future task.

Due to the time constraint, I could only experiment with presence of emoticon in a sentence as a feature. Some emoticons like a wink smile ;), or a laughter expression :D are more likely to be present in an ironic sentence. An extension of this experiment could be to include classification of emoticons as a feature than just a mere presence of emoticons.

Another area of future work would be to fine-tune and train LSTM model to give a better performance for this task. Other small enhancements could be made like using of GloVe pre-trained on twitter data than the normal corpus.

8. References

[1] Ghosh, A. and Veale, T.: 2016, Fracking Sarcasm using Neural Network, Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Association for Computational Linguistics, San Diego, California, pp. 161?169.

[2] Joshi, A., Bhattacharyya, P. and Carman, M. J.: 2016, Automatic Sarcasm Detection: A Survey

[3] Liu, B.: 2012, Sentiment Analysis and Opinion Mining, Synthesis Lectures on Human Language Technologies, Morgan and Claypool Publishers.

[4] Maynard, D. and Greenwood, M.: 2014, Who cares about Sarcastic Tweets? Investigating the Impact of Sarcasm on Sentiment Analysis, in N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk and S. Piperidis (eds), Proceedings of the Ninth International Conference on Language Resources and Evaluation (LRC'14), European Language Resources Association, Reykjavik, Iceland.

[5] https://competitions.codalab.org/competitions/17468

[6] Pang, B., and Lee, L. 2008. Opinion Mining and Sentiment Analysis. Found. Trends Inf. Retr. 2(1-2):1?135.

[7] Quintilien, and Butler, H. E. 1953. The Institutio Oratoria of Quintilian. With an English Translation by HE Butler. W. Heinemann.

[8] Picard, Elie et al. ?Rate and place of death from asthma among different ethnic groups in Israel: national trends 1980 to 1997.? Chest 122 4 (2002): 1222-7.

[9] Barbieri, Francesco and Horacio Saggion. ?Automatic Detection of Irony and Humour in Twitter.? ICCC (2014).

[10] Carvalho, Paula et al. ?Clues for detecting irony in user-generated contents: oh...!! it's "so easy" ;-).? (2009).

[11] Gonzlez-Ibez, Roberto I. et al. ?Identifying Sarcasm in Twitter: A Closer Look.? ACL (2011).

[12] Wang, Jin et al. ?Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model.? ACL (2016).

[13] Manning, C.D., Pennington, J., & Socher, R. (2014). Glove: Global Vectors for Word Representation. EMNLP.