Irony Detection
In English Tweets

Elona Koromani, Ashish Ashutosh, Bhanu Priya Vijayraj

University of Passau

A Text Mining Project Report

April 2018

1 Abstract

Irony is a linguistic device used to express the contrary to what is literally being said. It appears in various forms or contexts and it often has ambiguous interpretations.
The purpose of our project is to detect the implicit presence of irony in English tweets. The project is divided into two different sub tasks:

1. Distinguishing between Ironic and non-ironic tweets.
2. Scoring of English tweets (-100 non-ironic to 100 ironic).

The proposed solution is based on supervised learning through a limited set of features extracted from the text.

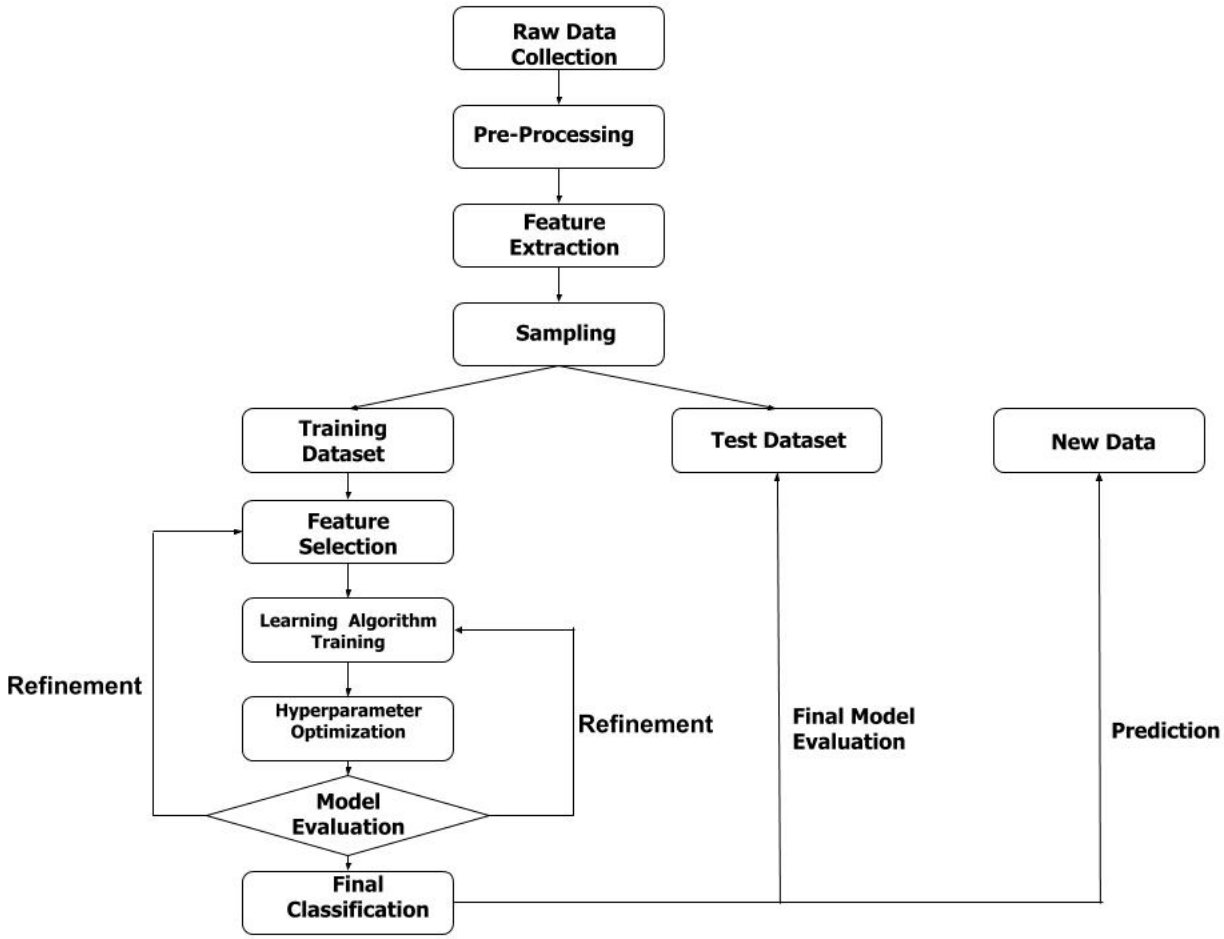
2 Introduction

Motivation and Challenges

The frequent use of irony, especially in social media, makes irony detection an interesting field for human-computer interaction. It has a large potential for applications in various research areas such as text mining, detecting online harassment and sentiment analysis. Detecting sentiment in social media like Twitter has become an essential task as they influence every business organization. In the same time it has a huge impact in the medical field where irony detection can help in detecting brain injuries at an early stage. Hence working on this project will provide a great insight on the NLP techniques and challenges that lies ahead.

3 Proposed method for Irony detection

In this work, we use a binary classification approach for irony detection where the labels of data instances are ironic and non-ironic. Before the supervised learning step, we go through data collection and annotation, preprocessing and feature extraction steps.



3.1 Data Collection and Annotation

The corpus for irony detection in English tweets is provided by SemEval 2018. The datasets have tweets with a corresponding binary score (0 or 1) indicating whether the tweet is ironic or not. The data was constructed from English tweets, by searching Twitter for the hashtags irony, sarcasm and not. To minimize the noise introduced by groundless irony-related hashtags, all tweets were manually labeled using a fine-grained annotation scheme for irony. Prior to data annotation, the entire corpus was cleaned by removing retweets, duplicates and non-English tweets, and replacement of XML-escaped characters.

Training and Test Datasets This corpus consists of 4,792 tweets: 2,936 ironic and 2,396 non-ironic. The tweets were collected by searching Twitter for hashtags: irony, sarcasm and not. To minimize the noise introduced by groundless irony-related hashtags, all tweets were manually labeled using a fine-grained annotation scheme for irony. The preprocessed entire corpus is cleaned by removing removing re-tweets, duplicates and non-English tweets. The data set will be randomly split into a training (80% or 3,833 instances) and test (20%, or 958 instances) set.

3.2 Data Preprocessing

Data Preprocessing is used to convert the raw data into a clean data set. Since the dataset may contain outliers, which are not in a suitable format, before starting with the process of detecting irony, we need to do some preprocessing. Also, for achieving better results from the applied model in Machine Learning we perform the following steps:

- Remove url (http://example.com)
- Remove mentions (@example)
- Remove unicode
- Standardising words, we replace wooow with wow
- Contractions are applied, words like theyre are replace with they are
- Word segmentation, words like Iamtheboss is split into I am the boss
- Unwanted special symbols are removed (@$—)
- Spelling correction
- Extra whitespace, open line or empty line are removed
- The length of a tweet as the number of characters must be more than 3
- Stop words are removed, they are high frequencies words such as is, at etc..
- Lemmatization is done, it converts a word into its base form.

3.3 Feature Extraction

Features are the basic clues and the most important part for detecting irony. Firstly, we focused on seven different approaches for feature extraction, however, we aimed to keep a limited set of features reported to be more effective. These approaches were used to represent the tweet as a feature vector, in order to train the classifier to detect irony. The definition of these features are as given below.

Punctuation and special symbols feature: It is considered that punctuation has a lot of influence in the text classification especially in the area of the sentiment analysis. Upper Case: In formal texts, the first letter of proper nouns and the first word of the sentence are written in capitals. In general, people using Twitter do not try to follow the grammar rules. However, when someone want to emphasize a word or a phrase, he/she writes the word/phrase in upper case.

Exclamation marks: Having an exclamation mark in a sentence does not indicate being ironic. Normally, exclamation marks are used to indicate strong feelings. However, having repeated exclamation marks is also one of the important clues.

Quotation marks: Quotation marks are used when a direct speech, name or title will be taken part in a sentence. Also, they are used for indicating a different meaning of a word/phrase. Often they express irony when the word/phrase between quotation marks has a positive/negative polarity.

Ellipsis and Punctuation: Having regular full stops does not give a clue for irony, however, an ellipsis(three periods of full stops) indicates an unfinished thought and a nervous or awkward silence. Also, combined with punctuation like(question/exclamation marks) makes this feature a stronger clue for irony.

N-grams features: N-grams refers to sequence of tokens within a statement where N refers to number of words in the sequence. We have taken n consideration unigrams and bigrams. To use the N-grams in the classifier the tweet is tokenized, lemmatized un-capitalized and stored as binary feature.

Sentiment and Subjective feature:
One of the most common structure in ironic statements is the presence of contrast in sentiments. Polarity of a sentence gives a strong clue for detecting irony. The gap between positive and negative sentiment scores indicate an unbalance in the sentence. Before performing the sentiment approach we replaced every emoticon and some interjections with a corresponding noun/word.

We used two different approaches to get features set based on the sentiments:

1. using SentiWordNet lexical resource for opinion mining.
2. using TextBlob python library(built-in sentiment score function).

For each of these two approaches we used three variations.

1. The sentiment score of the whole tweet is taken as a feature.
2. The tweet is split into two parts and the sentiment score of each half plus the difference(contrast) between both parts is used as a feature.
3. The tweet is divided into three parts and sentiment score of each part and their contrast are taken into consideration.

As long as TextBlob library provides the subjectivity score of a tweet, it is also taken as a feature.

POS-Tags feature: The parts of speech of the tweet are counted and inserted into the features. Here I have used four tags Noun, Verb, Adjective and Adverb. So, four different features are created using the POS-Tags.

3.4 Supervised Learning for Irony Detection

Sampling: After we extracted certain features from our data, we randomly split our dataset into training and test dataset. The training dataset is used to train the model, and the purpose of the test dataset is to evaluate the performance of the final model at the very end. The objective of training the model is to build a generalized model and make better predictions when unseen data samples are given to it. We model the task of irony detection as a supervised classification problem. In order to construct a model for irony detection, we have applied several classification algorithms on the constructed feature vectors. The used classification algorithms are chosen from different supervised learning approaches, Support Vector Machine(SVM), Logistic Regression, Gradient Boosting classifier and Decision Tree classifier, which are reported to have successful performance for similar problems in the literature.

4 Experimental Analysis

In this section, we present two sets of experiments in order to investigate the effect of the used classification algorithm and the effect of the considered attributes/features to detect irony. Irony detection performance is measured in terms of precision, recall, F1-score and overall accuracy for both ironic and non-ironic labels.

Support Vector Machine (SVM) Linear kernel: SVM is a supervised learning algorithm that can be used for both classification and regression. Our purpose is to use it for classification. SVM is known for its simplicity and effectiveness in binary classification. Sklearn is the used library in training. Hyper-parameter optimization: Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes. For Support Vector classifier the used kernel is linear and it doesn't consume as much time and resources on large data compared to polynomial kernel. For the parameter C (cost of classification or penalty parameter of error) the best performance among c=1, c=10, c=0.1, was for the value 0.1.

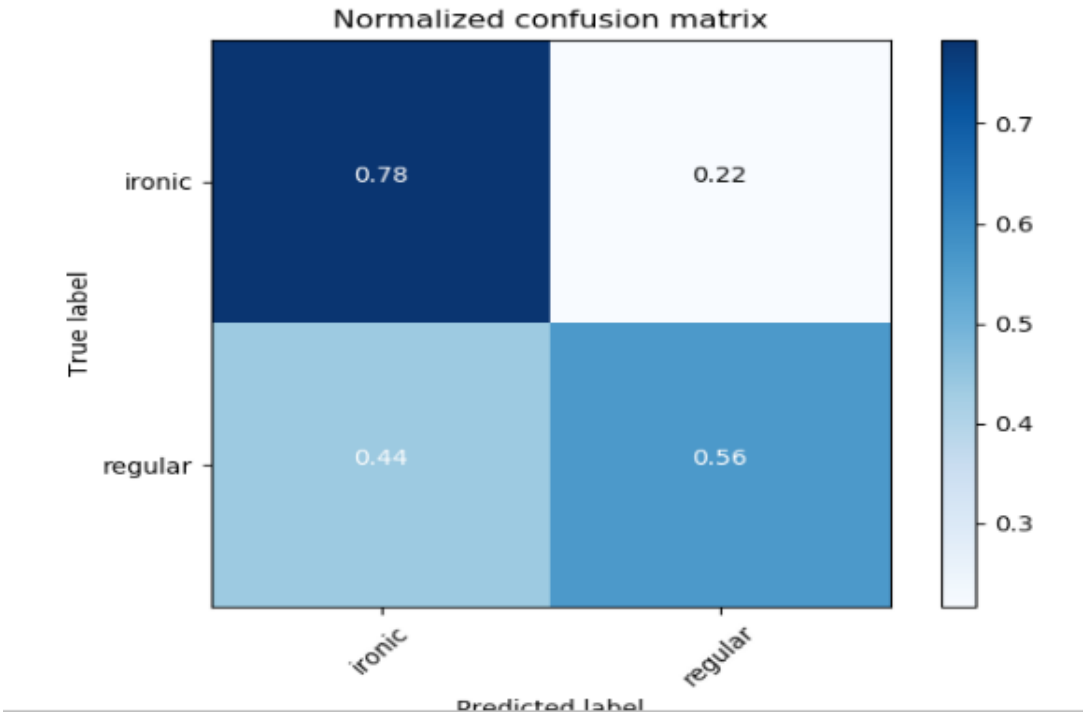
Cross-Validation: As long as our data set is small a major advantage is using cross-validation, because it does not waste too much data, although it can be computationally expensive. In our case we used 5-fold validation, which splits the training set into 5 smaller sets. The model is trained using 4 of the folds as training data and the remaining data is used to validate the resulting model(used as a test set to compute accuracy). For SVM classifier we had the best accuracy when using the cross-validation.

Evaluation: To evaluate the performance of the algorithm for the given features set the used metrics are Precision, Recall, F-score and Accuracy.

Scoring: For a new statement to be detected as ironic, a score is deduced in terms of percentage. This was done using the margin(distance from the boundary) of the new statement, which is calculated from the classifier and it is used in a sigmoid function to obtain a value between 0 and 1. This value is scaled to a value between -1 and +1 and multiplied by 100 to obtain the ironic score in terms of percentage. In overall, if the obtained score is positive the statement is considered ironic and the higher the positive score is the more ironic is the statement.

Prediction results under SVM classifier

SVM classifier	Precision	Recall	F1-score
Ironic	66%	78%	72%
Non-ironic	71%	56%	63%
Avg/total	68%	68%	67.8%



Logistic Regression Classifier

Logistic Regression is an algorithm that is relatively simple and powerful for deciding between two classes(binary classifier). It basically gives a function that is a boundary between two different classes. Logistic regression can be binomial, ordinal or multinomial. in our case we are using Binomial or binary logistic regression, which deals with situations in which the observed outcome for a dependent variable can have only two possible types, i.e. "ironic", "non-ironic". For detecting irony, logistic regression was the second best classifier, depending on the features that we select(later section).

Boosting

Boosting is a popular machine learning algorithm that increases the accuracy of the model. The idea behind boosting is converting many weak learners to form a single strong learner.
In our case we used the Gradient Boosting Classifier, which supports both binary and multi-class classification. We fit the gradient boosting classifier with 100 decision stumps as weak
learners.

Decision trees

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. We used decision trees for binary classification for detecting irony and parameter criterion is used as gini for quality split. We do not use entropy in our case since it does not yield good accuracy. The accuracy fluctuates due to random seed.

4.1 Analysis on the Effect of Classification Algorithms(Discussion)

As described before we used Support Vector Machine(SVM), Logistic Regression, Gradient Boosting and Decision Trees algorithms for irony detection. The working principles of all of these algorithms are different from each other. Therefore, we intended to investigate how the accuracy change according to the used algorithm. In this Experiment we used all 7 features. Irony detection performances under the given algorithm are presented below.

Prediction results for all features under the given classifiers

All Features		SVM	Logistic Regression	Gradient Boosting	Decision Trees
Ironic	Precision	67%	65%	64%	59%
	Recall	73%	75%	71%	70%
	F1-Score	**70%**	**70%**	**67%**	64%
Non-Ironic	Precision	65%	68%	64%	64%
	Recall	59%	57%	53%	54%
	F1-Score	**61%**	**62%**	**57%**	59%
Overall Accuracy		65.9%	66.5%	62.6%	61.2%

As it can be seen in the table, the most successful results are obtained by the Logistic Regression Classifier. SVM classifier is the second best classifier when considering all features. While the other two classifiers, did not have a good performance. Gradient Boosting was expected to improve accuracy, but in our case it didn't perform very well.

4.2 Analysis on the Effect of the Attributes/Features on Detecting Irony (Discussion)

In this set of experiments, we investigate the effect of feature filtering on irony detection. For this task we followed Wrapper Method which consists of three approaches: 1.Forward Selection- adding features one after the other and checking the accuracy. 2.Backward Selection- removing features one after the other and testing the accuracy. 3.Subset Selection.

We followed the first two approaches and we came into conclusion that Forward Approach was more efficient than the Backward Selection. Also, removing POS-tagging feature was more efficient to have a better performance for every classifier. The most accurate algorithm is SVM in among four classifiers and its overall accuracy 67.77 %.

The following table shows the performance for each classifier when not considering the POS feature.

All Features/POS		SVM	Logistic Regression	Boosting	Decision Trees
Ironic	Precision	66%	66%	63%	61%
	Recall	78%	73%	72%	70%
	F1-Score	**72%**	**70%**	**67%**	65%
Non-Ironic	Precision	71%	68%	63%	62%
	Recall	56%	61%	54%	52%
	F1-Score	**63%**	**64%**	**58%**	57%
Overall Accuracy		67.77%	67%	63.2%	61.6%

Table1. Prediction results under different classifiers

After performing the wrapper method the we also tried to investigate the classifiers performance when removing features such as POS + Exclamation Count and POS + Scare Quotes. The chart below shows the classifiers performance considering the above mentioned features.

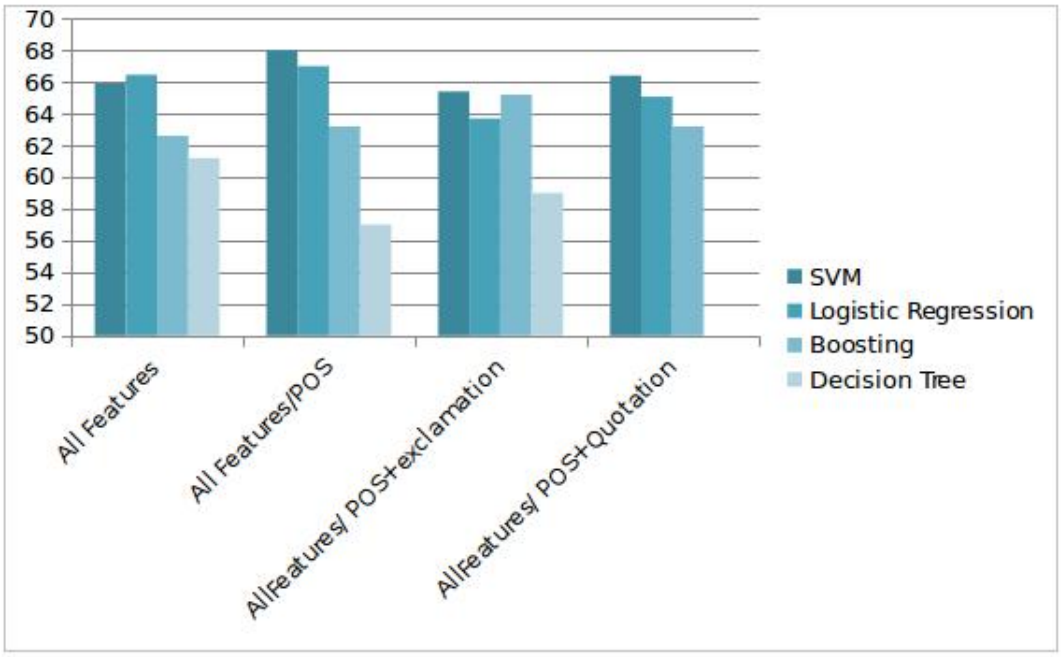


Figure1. comparison of the effect of different feature selections

The following tables shows the prediction success results under 5 best features: Upper Case, Bi-grams, Ellipsis and Punctuation, Sentiment Contrast, Exclamation Count/ Quotation Marks.

Table2&3. Prediction results under top 5 features for different classifiers

All Features/POS &Exclamation		SVM	Logistic Regression	Gradient Boosting	Decision Trees
Ironic	Precision	64%	62%	62%	60%
	Recall	76%	72%	78%	60%
	F1-Score	**69%**	**66%**	**69%**	60%
Non-Ironic	Precision	67%	66%	70%	57%
	Recall	54%	56%	52%	57%
	F1-Score	**60%**	**61%**	**60%**	57%
Overall Accuracy		65.2%	63.7%	65.2%	59%

All Features/POS &Quotation		SVM	Logistic Regression	Gradient Boosting	Decision Trees
Ironic	Precision	65%	63%	62%	61%
	Recall	76%	74%	73%	70%
	F1-Score	**70%**	**68%**	**67%**	65%
Non-Ironic	Precision	69%	68%	65%	61%
	Recall	56%	56%	53%	52%
	F1-Score	**62%**	**62%**	**58%**	56%
Overall Accuracy		66.44%	65%	63%	61%

5 Future Improvements

Firstly the accuracy can be improved by having a larger dataset. In this way the model can learn more and it will predict better. Also, when having more data we can include higher N value features (n-grams). Another Improvement would be to work with Neural Networks and Deep Learning. Machine Learning Algorithms parse data, learn from that data, and then apply what theyve learned to make informed decisions. Although deep learning is just a subset of machine learning and functions in a similar way, its capabilities are more impressive. A deep learning model is designed to continually analyze data with a logic structure similar to how a human would draw conclusions. To achieve this, deep learning uses a layered structure of algorithms called an **artificial neural network (ANN)**. The design of an ANN is inspired by the biological neural network of the human brain. This makes for machine intelligence thats far more capable than that of standard machine learning models.

6 Conclusion

Irony detection is a complex and challenging phenomenon, as it is hard to find full agreement even on manual annotation. It is useful to be able to detect irony especially for tasks that involve extraction user opinion, since sentiment analysis cannot correctly predict opinion for sarcastic expression in texts. Irony detection in Twitter has additional challenges due to the use of short text and informal language. In this project we used a supervised approach for irony detection in English tweets. We defined 7 different features, where some of them are language independent. for supervised learning we compared performance of fours different classification algorithms, SVM, Logistic Regression, Gradient Boosting, Decision Trees. Supervised learning based solution appears to be useful for irony detection under the extracted features approach.

In this project, we could notice how a social media like Twitter can act as huge asset in terms of data gathering and how few basic features like Sentiment Contrast, Punctuations, N-grams can be powerful in the detection of sophisticated language form of irony. Data pre-processing and feature engineering are with very importance for improving accuracy and more in-depth analysis in these domains will help in improving the accuracy considerably.

7 References

1. An Impact Analysis of Features in a Classification Approach to Irony Detection in Product Reviews. http://www.aclweb.org/anthology/W14-2608

2. Joshi, A., Bhattacharyya, P. and Carman, M. J.: 2016, Automatic Sarcasm Detection:A Survey, CoRR abs/1602.03426.

3. Reyes, A., Rosso, P. and Veale, T.: 2013, A Multidimensional Approach for Detecting Irony in Twitter, Language Resources and Evaluation 47(1), 239 268.

4. F.Barbieri, H.Saggion.: 2016 Modelling Irony in Twitter. http://www.aclweb.org/anthology/E14-3007

5. E.Forslid, N.Wikn, Automatic irony- and sarcasm detection in Social media. http://uu.diva-portal.org/smash/get/diva2:852975/FULLTEXT01.pdf

6. Hande Taslioglu, Pinar Caragoz Irony detection in microposts with limited features

7. Prashant Kikani Sarcastic sentence detector https://github.com/prashant-kikani/Sarcasm-detector/tree/master/app

8. Konstantin Buschmeier. Irony-detection https://github.com/kbuschme/irony-detection

9. Varuun Muthanna Sarcasm Detection https://github.com/varunmuthanna/Sarcasm-Detection

10. Supervised Learning http://scikit-learn.org/stable/supervised_learning.htmlsupervised-learning

11. Predictive modeling, supervised machine learning, and pattern classification. http://sebastianraschka.com/Articles/ $2014_{i}ntro_{s}upervised_{l}earning.htmlcross-validation$