CTSys at SemEval-2018 Task 3: Irony in Tweets

Myan Sherif
Faculty of Engineering
Alexandria University
sherif.myan@gmail.com

Sherine Mamdouh
Faculty of Engineering
Alexandria University
sherym51@gmail.com

Wegdan Ghazi
Faculty of Engineering
Alexandria University
wegdan.ghazi@gmail.com

Abstract

The objective of this paper is to provide a description for a system built as our participation in SemEval-2018 Task 3 on Irony detection in English tweets. This system classifies a tweet as either ironic or non-ironic through a supervised learning approach. Our approach is to implement three feature models, and to then improve the performance of the supervised learning classification of tweets by combining many data features and using a voting system on four different classifiers. We describe the process of pre-processing data, extracting features, and running different types of classifiers against our feature set. In the competition, our system achieved an F1-score of 0.4675, ranking 35th in subtask A, and an F1-score score of 0.3014 ranking 22th in subtask B.

1 Introduction

Irony detection in text has extended to different data forms (tweets, reviews, TV series dialogues), our domain of data in this task is a Twitter corpus provided by SemEval2018 organizers. Here, irony detection refers to computational approaches to predict if a given text is sarcastic. This problem is hard because of the nuanced ways in which irony may be expressed. The most difficult part of the problem mentioned is the process of feature engineering, because it defines the parameters and the relationships and dependencies between semantic meanings, and gives us the numerical model that the classifier would proceed to work on, thus being crucial to the soundness and efficiency of the system.

This led us to dive into deeper questions, such as the nature of tweets, and how we are dealing with a version of the English language that is not directly workable. We need to perform pre-processing to deal with annotations and hashtags. Another question is how to analyze irony in English language and derive a rule-based approach that can be implemented to better understand the semantics of ironic text.

The problem, as described by the SemEval-2018 task organizers, addresses both the binary distinction between irony and non-irony, as well as different types of irony.

1.1 Task Description

The SemEval-2018 Task 3 is divided into two subtasks:

Subtask A is a binary classification problem where we are asked to classify a tweet as ironic or not ironic, based on a given training set of labeled tweets (0 for non-ironic and 1 for ironic).

Subtask B is a multi-classification problem where we classify the tweets to which type they belong as either situational irony or verbal irony or other irony or not ironic. Each tweet in the training set is labeled as follows: (0 for non-ironic, 1 for situational irony, 2 for verbal irony, and 3 for other forms of irony).

1.2 The Dataset

The used dataset in this assignment is the one provided in SemEval-2018 task 3. It consists of 3,842 tweets in total. The tweets were collected by searching Twitter for the hashtags #irony, #sarcasm and #not.

The dataset was presented in two phases:

1- Training data: already labeled tweets used to train the classifiers. Each tweet was provided with a binary classification label and an index.

2- Testing data: unlabeled tweets to test the classifiers against. For each instance in the test

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 576–580
New Orleans, Louisiana, June 5–6, 2018.

data, participants submitted a predicted label. Based on these predictions, competition scores were calculated using four metrics (F1-score, precision, recall, and accuracy).

2 Literature overview

There has been much research involving the definition of irony and the distinction between irony and sarcasm. To date, however, experts do not formally agree on the distinction between irony and sarcasm as shown by Aditya Joshi et al., (2016). Moreover, when describing how irony works, Antonio Reyes et al., (2013), distinguish between situational irony and verbal irony. Situational irony is an unexpected or incongruous quality in a situation or event, as shown by Shelley (2001). Whereas verbal irony, in contrast, is a playful use of language in which a speaker implies the opposite of what is literally said.

In his work on the Sarcasm Detector website, Mathieu Cliche collected tweets from Twitter that were labeled with the hashtag #sarcasm. His hypothesis was that sarcastic tweets carry what he calls a contrast of sentiments (e.g. start with a positive sentiment and end with a negative sentiment). He also uses features such as n-grams and topics as accompanying features then trains an SVM algorithm as a classifier. Cliche's system harbored an F1-score of 0.60, an improvement from previous work on sarcasm detection as shown in Cliche (2014).

Chun-Che Peng et al., (2015) followed up on Cliche's work to acquire improved results and stated that irony detection models are prone to suffer from high variance, which be the effect of having a high dimensional feature space, therefore making it important to reduce the dimensions of the feature space and only use the most relevant features. Their paper also suggests that using a Gaussian kernel instead of a linear kernel might be a better approach, given that the data itself is not linearly separable.

In our work, we build upon Cliche's (2014) hypothesis and try to benefit from Peng et al.'s (2015) remarks on using the most relevant features.

3 Implementation

The system is based on natural language processing where we are targeting to improve performance for classifying tweets as ironic or non-ironic by combining many data features and a voting system on many classifiers, we design pattern-based features that indicate the presence of discriminative patterns as extracted from a large irony-labeled dataset.

3.1 Text Preprocessing

To generate good results and to control the number of unneeded computations, the tweets are filtered according to certain criteria. We will briefly go through the steps of pre-processing a tweet.

3.1.1 Tokenization

The first step to handle textual data is tokenization, which is the process of splitting sentences into single words.

3.1.2 Stop Words

The second step is to filter the data and remove any insignificant and redundant words. There are known words, called stop words, as shown by Alani (2014) are always removed to enhance the performance.

For the objective of the task, irony detection in tweets, we removed some words from the **Stop Words** sets because they are significant in detecting irony, especially in the sentiment analysis model. In sentiment analysis, we removed any negating words and conjunctions, such as: ("no", "not", "until", "but"). Whereas in BoW, keeping negation was unnecessary.

3.1.3 Lemmatization

Lemmatization is the process of getting the root of a word. It takes into consideration the morphological analysis of words. A lemma is the same for variations of a word, therefore; it reduces sparsity.

3.2 Extracting Features

We here convert the tweet into a vector of dimensional attributes. While feature mapping is the hardest step in the code, the pattern of feature engineering in task A and task B is all the same, we follow the same steps of mapping and classifying to get different outputs due to different training data on the models.

We have tried three different directions in regards to extracting features from the dataset. The first being the bag of words (BoW) model, the second is rule-based sentiment analysis, and the third being word embedding.

Four classifier models were used to train and test the three feature sets implemented. Each feature set of which is tested on each classifier model. In other words, we test (feature set '1 of 3', classifier '1 of 4') pairs. Then we used a voting system to compare between the results of (feature set, classifier) pairs, and then the classification with the higher number of votes is picked as the final classification.

3.2.1 Bag of Words Feature

First, we create three arrays. The first array for the words in ironic tweets, the second array for the words in non-ironic tweets and the final array for words in all tweets. **Second**, we calculate the number of repetitions of every word in the ironic tweets array across all ironic tweets. We repeat the same step for every word in the non-ironic tweets across all non-ironic tweets. **Third**, we extract the most common words (with highest frequency) across both tweets to eliminate them from our processing to the data - since they will not be effective in determining if a tweet is ironic or not. **Fourth**, we create hash-maps for the words as 'key' attribute and their frequency value as 'value' attribute - one hash-map for words in ironic tweets, another for words in non-ironic and the last one for the common ones. **Fifth**, we sort the hash-maps for easy acquiring of the words with highest frequencies. **Finally**, we add the hash-maps as another feature for the data processing procedure.

3.2.2 Sentiment Analysis

According to Van Hee et at., (2016), verbal irony arises from a clash between two evaluation polarities. We use sentiment analysis to help detect irony in a tweet via contrasting polarity. We used the **polarity** feature of a word to determine if the feelings in the tweet changed 180 degrees. We did not apply lemmatization prior to extracting this feature because it affects polarity. We also handle **emojis** and **negation words** in the tweets since they contribute to the polarity of the sentence. Below are the steps we perform.

a. Split the tweet into two parts on a conjunction from a list created by hand. We gather all the available conjunctions in English Grammar. We handle all the conjunctions except the ones that consist of more than one word like "not only... but also"… etc.
b. Perform pre-processing on each part of the tweet individually.
c. Evaluate polarity of each word of each part of the sentence, and then define the polarity of each part given the ratios of positive, negative, and neutral words to the total length of the sentence.
d. Each part is given a tag as positive (POS), negative (NEG), or neutral (NEU).
e. We tune the parameters that define the threshold of positivity or negativity of each part of the sentence, being 0.5 in this case.
f. Compare the polarities of the sentence parts.

To sum up: The **Sentiment Analysis** method uses contrasting polarity or extra positivity and extra negativity as an indication of irony. We split the tweet into two parts, taking each part as input into the **Sentiment Intensity Analyzer**, the polarity of each word is returned by the analyzer as either positive (POS), negative (NEG) or neutral (NEU). To calculate the **overall polarity** of one part of the tweet, we search for the polarity category that has **highest** number of words and return it as the **overall** polarity. The overall polarity of both parts of a tweet is then examined and classified as ironic if contrasting polarity (e.g. POS-NEG or NEG-POS) is found.

3.2.3 Word Embedding

First, we build a model using training data to act like a dictionary for upcoming processing. The model used in this step is a Word2Vec model. **Second**, we process each tweet in the training dataset, using every word in every tweet and passing it to the model - which as a result, returns an equivalent numerical vector to the word with a fixed length, in our case; we choose a length of one hundred (100) as a moderate length value. **Third**, we add all the vectors of the words in each tweet and divide this sum by their number. Thus, we acquire a numerical representation of a fixed length for every tweet. Fourth, we append all those vectors of all tweets. Finally, we pass the resulting appended vectors of all tweets to the classifier. If the word did not exist in the dictionary we made beforehand, a vector of length 0 is returned.

3.3 Choosing a classifier

We use four models for classification and we build a voting system for them all, tune the parameters, and record the findings to enhance the performance, the classification models are selected based on the literature review. The classification algorithms used are listed below:

- Naive Bayes Classifier.
- Support Vector Machine (SVM).
- Decision Trees**.**
- K-Nearest Neighbor Classifier: After some tuning, k=1 generated the best results for all the features.

4 Results

Our system is divided into three classes one for each feature. Then the result of each is classified using the four different classifiers stated above. Below we present a chart of the accuracies obtained with different classification algorithms and different feature types.

	BoW	Sen-A	Word-E
NB	65	44	49
SVM	62	45	53
Trees	57	59	57
1-NN	52	60	48

Table 1: accuracy of feature-classes when tested against classifiers using the training set for task A.

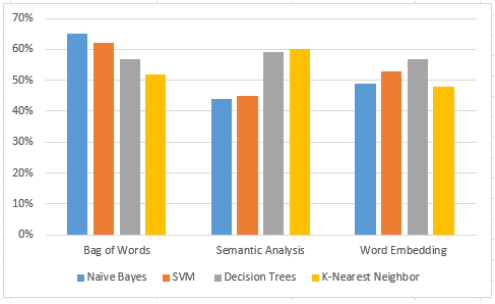

Figure 1: results obtained by two voting systems using three feature set types as shown.

4.1 Classifiers Voting System

We used a voting system to combine the predictions from the four classifiers exploiting different feature types.

4.2 Features Voting System

This voting system uses the four output results from the four classifiers voting system to get an overall result for the whole system.

The results of the system evaluation phase are as follows:

	Accuracy	Precision	Recall	F1-score
Task A	0.5089	0.4102	0.5434	0.4675
Task B	0.4923	0.2998	0.3108	0.3014

Table 2: The score obtained by the system in subtasks A and B as evaluated by SemEval.

4.3 Analysis

Looking at the results, we hypothesize that the system's performance can be improved by combining all features instead of testing them individually. It was also remarkable that the best accuracy was obtained by the bag-of-words model using the Naïve Bayes classifier.

We also believe better results can be achieved if there was a bigger dataset at hand to train upon, and if we had sufficient time to perform grammar checking on the tokens and other operations that can reduce noise.

5 Conclusion

This paper describes our irony detection system that was built in the framework of SemEval-2018 Task 3. We used the same architecture for subtask A and B and obtained F1-scores of 0.4675 and 0.3014, respectively. Our binary classification results are much better compared to multi-classification, which implies that we need to implement another feature model that could represent a whole sentence (e.g. Sentence2Vec rather than Word2Vec). In future work, we aim to enhance the performance of our classifier by combining all features. Moreover, we will add new features to solve the problem of word dependencies (by this we mean that all system features do not account for dependencies between words in the same sentence) so that the system gives more accurate results.

Acknowledgements

The authors would like to thank Prof. Ayman Khalafallah of Alexandria University for his constant guidance and support throughout the process of developing this system.

References

Harith Alani, Miriam Fernández, Yulan He and Hassan Saif. 2014. *On stopwords, filtering and data sparsity for sentiment analysis of Twitter*. In proceedings of LREC 2014, 9th International Conference on Language Resources and Evaluation:810–817.

Mathieu Cliche. 2014. *The sarcasm detector*. URL: http://www.thesarcasmdetector.com/about/.

Aditya Joshi, Mark James Carman and Pushpak Bhattacharyya. 2017. *Automatic Sarcasm Detection: A Survey*. ACM Computing Surveys 50(5) Article 73, 2017. https://doi.org/10.1145/3124420

Chun-Che Peng, Jan Wei Pan and Mohammad Lakis. 2015. *Detecting Sarcasm in Text: An Obvious Solution to a Trivial Problem.* Stanford CS 229 Machine Learning Final Project.

Antonio Reyes, Paolo Rosso and Tony Veale. 2012. *A multidimensional approach for detecting irony in Twitter*. Language Resources & Evaluation, March 2013, 47(1) :239–268. https://doi.org/10.1007/s10579-012-9196-x

Cameron Shelley. 2001. *The bicoherence theory of situational irony*. Cognitive Science 25:775-818. https://doi.org/10.1016/S0364-0213(01)00053-2

Cynthia Van Hee, Els Lefever and Veronique Hoste. 2016. *Guidelines for Annotating Irony in Social Media Text*. LT3, Department of Translation, Interpreting and Communication, Faculty of Arts, Humanities and Law - Ghent University, Belgium.