NLPRL-IITBHU at SemEval-2018 Task 3: Combining Linguistic Features and Emoji pre-trained CNN for Irony Detection in Tweets

Harsh Rangwani, Devang Kulshreshtha and Anil Kumar Singh

Indian Institute of Technology (Banaras Hindu University) Varanasi, India
{harsh.rangwani.cse15, devang.kulshreshtha.cse14, aksingh.cse}@iitbhu.ac.in

Abstract

This paper describes our participation in SemEval 2018 Task 3 on Irony Detection in Tweets. We combine linguistic features with pre-trained activations of a neural network. The CNN is trained on the emoji prediction task. We combine the two feature sets and feed them into an XGBoost Classifier for classification. Subtask-A involves classification of tweets into ironic and non-ironic instances, whereas Subtask-B involves classification of tweets into *non-ironic, verbal irony, situational irony* or *other verbal irony*. It is observed that combining features from these two different feature spaces improves our system results. We leverage the SMOTE algorithm to handle the problem of class imbalance in Subtask-B. Our final model achieves an F1-score of 0.65 and 0.47 on Subtask-A and Subtask-B respectively. Our system ranks 4^{th} on both tasks, respectively, outperforming the baseline by 6% on Subtask-A and 14% on Subtask-B.

1 Introduction

According to the Merriam-Webster dictionary[1], one of the meanings of irony is defined as 'the use of words to express something other than and especially the opposite of the literal meaning' (*e.g. I love getting spam emails.*). Irony can have different forms, such as verbal, situational, dramatic etc. Sarcasm is also categorized as a form of verbal irony. Various attempts have been made in the past for detection of sarcasm (Joshi et al., 2017). Sarcastic texts are characterized by the presence of humor and ridicule, which are not always present in the case of ironic texts (Kreuz and Glucksberg, 1989). The absence of these characteristics makes automatic irony detection a much more difficult problem than sarcasm detection.

Irony detection is a problem that is central to the working of many Natural Language Understanding Systems. For example, users often use irony to express their opinions on social media like Twitter (Buschmeier et al., 2014). Detecting irony in social texts can aid in improving opinion analysis.

The SemEval 2018 task 3 (Van Hee et al., 2018) consists of two subtasks. Subtask-A involves predicting whether a tweet is ironic or not and Subtask-B involves categorizing a tweet into Non-Ironic, Verbal Irony (by means of a polarity contrast), Situational Irony and Other Forms of Verbal Irony. The task organizers use macro averaged F1, rather than accuracy to force systems to optimize to work well on all the four classes of tweets, as described in Section 3.1.

Systems built in the past primarily used handcrafted linguistic features for classification of ironic texts (Buschmeier et al. 2014; Farías et al. 2016). In our system, we try to combine them with the pre-trained activations of a neural network. Our results show that both types of features complement each other, as the results produced by the combination of them surpass the results of using either the linguistic or the pre-trained activation features individually by a large margin. We use XGBoost Classifier (Chen and Guestrin, 2016), as it performs at par with neural networks when the provided training data is of small size.

Our results indicate that oversampling techniques like SMOTE (Chawla et al., 2002) can also be used to oversample the representations generated using neural networks to improve performance on imbalanced datasets.

The rest of the paper is organized as follows: Section 2 gives a detailed description of how our system was built, Section 3 then describes the experimental setup and the results obtained and Section 4 concludes the paper.

[1] https://www.merriam-webster.com/dictionary/irony

2 Proposed Approach

For modeling irony in tweets, our system makes use of a combination of features. These features can be classified into two broad groups:

- Linguistic (Structure and User Behavior)
- Pre-trained Activations of a Neural Network.

These features were concatenated and the XGBoost classifier (Chen and Guestrin, 2016) was used to perform the classification.

For subtask B, to counter the imbalance in the dataset, which might lead classifiers to favor the majority class in classification, we used SMOTE for oversampling the data (Chawla et al., 2002). Then we used XGBoost Classifier again for classification into various classes.

The details of the classifier parameters are provided in Section 2.2. Basic preprocessing of tweets was performed before feature extraction, which involved removing hash symbols ('#'), converting contractions ('doesn't' to 'does not'), removing links and quotations and normalizing the text into lower case. We will explicitly mention those features whose extraction require the original tweets.

2.1 Feature Extraction

Our system generates a 72-dimensional handcrafted feature vector, based on the linguistic structure and user behavior. We then combine this with a 2304 dimensional feature vector generated using activations of a pre-trained CNN. The combined features are categorized into 11 broad classes:

2.1.1 Contrast Based Features

Contrast of sentiments is a feature that has been observed in sarcastic and ironic texts (Rajadesingan et al., 2015). *e.g. I love being ignored #not.* For capturing contrast, we use the affect score of lemmas (Warriner et al., 2013) and the sentiment score of words based on **SentiStrength** (Thelwall et al., 2010). The final feature vector consists of:

- The difference between the highest and lowest sentiment values of the words present in the tweet. (1 feature)
- The difference between the highest and lowest affect scores of the words present in the tweet. (1 feature)
- Longest unimodal sequence size and the number of transitions of sentiment polarity. (2 features)
- Sum of sentiment score and counts of positive and negative n-grams. (4 features)

2.1.2 Readability Based Features

Ironical texts are usually complex, and hence we use the total number of syllables in the tweet, along with number of words that contain polysyllables as features. According to Automated Readability Index (Senter and Smith, 1967) the standard deviation, the average and the median of the word length serve as indicators of the complexity of the text (Rajadesingan et al., 2015).

2.1.3 Incongruity of Context

Ironic similes are common in literature (*e.g. as clear as mud* in which both *clear* and *mud* are sentiment neutral words.). Due to this neutrality, the lexicon based methods are unable to capture the incongruity present. Therefore, maximum and minimum Glove (Pennington et al., 2014) cosine similarity between any two words in a tweet are used as features in our system (Joshi et al., 2016)).

2.1.4 Repetition-based Features

Users often change their writing style to depict sarcasm and irony, which is analogous to the change of tone in speech while expressing sarcasm. *e.g. Loooovvveeeeeee when my phone gets wiped.* We use the count of of words with repetitive characters and the count of 'senti words' (sentiment score $\geq$ 2 and sentiment score $\leq$ -2) with repetitive characters as our features (Rajadesingan et al., 2015).

2.1.5 Punctuation-based Features

Punctuation counts can serve as an indicator of ironic texts (Kreuz and Caucci, 2007). We use the counts of characters like hashtag (#), ellipsis (...), exclamation mark (!), question mark (?), colon (:), quote (") and apostrophe (') in a tweet as features.

2.1.6 Presence of Markers

Discourse markers are certain words that help in expressing ideas and performing specific functions (Farías et al., 2016). Our system uses a curated list of discourse markers. Similar to the list of discourse markers, we also use a list of intensifiers (*e.g. heck*), laughter words (*e.g. lmao, lol etc.*), interjections (*e.g. oops*) and swear words (*e.g. shit*) as their appearance in a tweet indicates the presence of unexpectedness, which can, in turn, serve as an indicator of irony. We use counts of these different types of words separately as features.

2.1.7 Word Count Features

According to (2016), ironic tweets depict their content in fewer words compared to normal tweets. Hence we use the word count of tweets as a feature. Apart from the word count, (Kreuz and Caucci, 2007) suggest that the counts of adjectives and adverbs can also be used as markers of ironic content. We also use the preposition count as a separate feature.

2.1.8 Semantic Similarity

Ironic tweets that span multiple lines are often found to have lines that are very much semantically dissimilar to each other (Farías et al., 2016). We use the **WordNet** based similarity function (Mihalcea et al., 2006) available online[2] to obtain a similarity score, which is used as a feature.

2.1.9 Polarity and Subjectivity

Ironic texts are usually subjective and often convey something negative (or positive) about the target (Wallace et al., 2015). We use the Polarity and Subjectivity Scores (Sentiment Score) generated using **TextBlob** as features in our model (Loria et al., 2014).

2.1.10 URL Count

We observed in the training set that users often used irony to express their opinion about online content, e.g. blogs, images, tweets, etc. For specifying the context of a comment (tweet), they often add a URL to the original content. So we used the counts of URLs in a tweet as a feature. Our system requires raw tweets for extracting this feature.

Apart from the above features, we also experimented with Named Entity Count and occurrence of popular hashtags like (*#hypocrisy*), using a curated list, as our features (Van Hee, 2017).

2.1.11 Pretrained CNN Features

Apart from extracting linguistic features from tweets, we leverage the activations of a Convolutional Neural Network (CNN) pre-trained on emoji prediction task. We use DeepMoji[3] (Felbo et al., 2017), a model trained on 1.2 billion tweets with emojis, and tested on eight benchmark datasets within sentiment, emotion and sarcasm detection. Since sarcasm is a form of verbal irony that expresses ridicule or contempt (Long and Graesser, 1988), we believe transferring the knowledge of CNN trained on sarcasm can improve the results of Irony Detection task.

Each tweet is converted into a 2304-dimensional feature vector by feeding into DeepMoji-CNN and extracting activations of the last hidden layer.

2.2 Classifiers

We construct XGBoost (Chen and Guestrin, 2016) feature-based classifiers for irony detection using the above features. Based on the 10-fold cross validation performance, the best performing parameters prove to be the default parameters used by the XGBoost Classifier Package [4].

2.3 Handling Class Imbalance

The data provided for subtask-B is highly skewed. To perform well on every class of irony, we used an oversampling technique (SMOTE (Chawla et al., 2002)). In SMOTE, for generating a new synthetic sample, from the k nearest neighbors of an instance, one is chosen at random, and a new sample is generated on the line joining the instance and the chosen neighbor. We use the SMOTE implementation available in *imblearn* (Lemaître et al., 2017) package for our system, with k_neighbors equal to 5.

3 Experiments and Evaluation

3.1 Dataset and Metrics

The annotated tweet corpus provided for training consists of 1390 instances of Verbal Irony due to polarity contrast, 205 instances of Other Types of Verbal Irony, 316 Situational Ironic instances, and 1923 Non Ironic instances. Our system only uses the training data provided by the organizers and no other annotated data is used (Constrained System).

The test dataset for Subtask-A contains 473 non-ironical tweets and 311 ironical tweets. For Subtask-B, the 311 ironical tweets are further classified into Verbal Irony by means of Polarity Contrast (164), Situational Irony (85) and Other Forms Of Verbal Irony (62).

The evaluation metric used for ranking teams in Sub-task A is the F1 score of the positive (Ironic)

[2] http://nlpforhackers.io/wordnet-sentence-similarity/

[3] https://github.com/bfelbo/DeepMoji

[4] http://xgboost.readthedocs.io/en/latest/parameter.html

Approach	Task-A			Task-B		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Linguistic	0.48	0.78	0.59	0.32	0.36	0.30
Pretrained CNN	0.60	0.63	0.62	0.51	0.44	0.42
Linguistic + Pretrained CNN	0.55	0.79	**0.65**	0.53	0.44	0.42
Linguistic + Pretrained CNN + SMOTE	-	-	-	0.46	0.51	**0.47**
Baseline (Linear SVC over BoW)	0.56	0.63	0.59	0.48	0.36	0.33

Table 1: F1 scores in Task A and Macro F1 in Task B on test set.

class whereas in Subtask-B, the organizers use macro averaged F1 (average of F1 for each class) as an evaluation metric for ranking teams.

3.2 Results and Discussion

We present the results achieved by our approaches, as well as the combination of our methods in Table 1. Our final submitted systems are: (Linguistic + Pretrained CNN) for Task-A and (Linguistic + Pretrained CNN + SMOTE) for Task-B. We discuss the major takeaways from the results below.

- Our submitted models achieve 4th position in public leaderboard [5] on both Task-A and Task-B and beat the task baselines by about 6% and 14%, respectively, on both tasks on the test set.

- Leveraging DeepMoji model for Irony detection domain yields a considerable improvement over purely linguistic features (0.03 and 0.12). This is because the model is trained on over a *billion* tweets on sarcasm and four other domains. As stated earlier, sarcasm is a verbal form of irony (Long and Graesser, 1988), and transfer learning works as domains are quite similar.

- Our combination of linguistic features with pre-trained CNN achieves an F-score of 0.65 and 0.42, with an improvement of at least 0.03 on Task-A and significant improvement in Task-B, compared to linguistic features. The higher accuracy points to the power of ensemble learning by combining different feature spaces, as both feature sets specialize in different types of tweets.

- The use of SMOTE oversampling technique leads to an F-score of 0.47 in Task-B, which is an improvement of 0.05 over (Linguistic + Pretrained CNN) model.

- The improvement in scores due to linguistic features are not as pronounced in Subtask-B, as compared to Subtask-A. One of the possible reasons for this is that linguistic features are not able to capture the fine grained differences between different forms of irony.

4 Conclusion

We reported the use of handcrafted features and pre-trained CNN activations for predicting the irony in tweets. We implemented a variety of features based on user behavior as well as the linguistic structure in a tweet. We further exploit the SMOTE oversampling technique to handle the class imbalance problem in Subtask-B, which involves categorizing a tweet into Non Ironic, Verbal Irony, Situational Irony and Other Verbal Irony. We then feed the features into XGBoost classifier for both the tasks. The benefit of using CNN models pre-trained on sarcasm, sentiment, and emotion domains can be clearly seen, yielding an improvement of 3% and 9% over task baselines. Our final submitted system stood 4^{th} in both the subtasks in the SemEval 2018 shared task on "Irony Detection in English Tweets".

References

Konstantin Buschmeier, Philipp Cimiano, and Roman Klinger. 2014. An impact analysis of features in a classification approach to irony detection in product reviews. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 42–49.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

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

Delia Irazú Hernańdez Farías, Viviana Patti, and Paolo Rosso. 2016. Irony detection in twitter: The role of affective content. *ACM Transactions on Internet Technology (TOIT)*, 16(3):19.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5):73.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are word embedding-based features useful for sarcasm detection? *arXiv preprint arXiv:1610.00883*.

Roger J Kreuz and Gina M Caucci. 2007. Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on computational approaches to Figurative Language*, pages 1–4. Association for Computational Linguistics.

Roger J Kreuz and Sam Glucksberg. 1989. How to be sarcastic: The echoic reminder theory of verbal irony. *Journal of experimental psychology: General*, 118(4):374.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

Debra L Long and Arthur C Graesser. 1988. Wit and humor in discourse processing. *Discourse processes*, 11(1):35–60.

Steven Loria, P Keen, M Honnibal, R Yankovsky, D Karesh, E Dempsey, et al. 2014. Textblob: simplified text processing. *Secondary TextBlob: Simplified Text Processing*.

Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 97–106. ACM.

RJ Senter and Edgar A Smith. 1967. Automated readability index. Technical report, CINCINNATI UNIV OH.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the Association for Information Science and Technology*, 61(12):2544–2558.

Cynthia Van Hee. 2017. *Can machines sense irony? : exploring automatic irony detection on social media*. Ph.D. thesis, Ghent University.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. SemEval-2018 Task 3: Irony Detection in English Tweets. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, SemEval-2018, New Orleans, LA, USA. Association for Computational Linguistics.

Byron C Wallace, Eugene Charniak, et al. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1035–1044.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207.