Sarcasm Detection in Twitter

CS 521 Final Project Report
Spring 2022

Abstract

Understanding both literal and figurative meaning of text from social media is crucial to study the user's opinions on various topics. Sarcasm is a popular figurative language in social media as it is generally associated with creativity and high intelligence. Due to its obscurity it is difficult to detect sarcasm from text. In this paper, we investigated various methods for sarcasm detection in Twitter. In particular, we looked into the importance of word representations on the performance of sarcasm detection classification. We observed that TF-IDF outperformed both word embeddings from word2vec and BERT. We speculate this could be due to the difference in training domains.

1 Introduction

Social media have become an integral part of human lives. People use these platforms to communicate and share their opinions and feelings over a wide range of topics. Unlike the more formal outlets, there is not any framework or acceptable norms for people in the way to express their opinions. Therefore, a significant portion of generated content is in the form of figurative language such as sarcasm. Sarcasm is "the use of words that mean the opposite of what you really want to say especially in order to insult someone, to show irritation, or to be funny" (Merriam-Webster). Sarcasm detection from text is a challenging task due to its obscurity, lack of contextual background knowledge and also lack of facial expressions. In addition, sarcasm detection from social media content has additional challenges due to the presence of spelling errors, use of special characters and slang.

Despite its many challenges, sarcasm detection from social media is particularly important as it has many applications such as opinion mining, marketing research and harassment detection. To this extent, researchers have proposed various computational methods such as rule-based, lexicon-based and machine learning/deep learning methods to detect sarcasm from text. We will talk about these methods in more detail in the next section.

In this paper, we employed various classification techniques such as Support Vector Machine (SVM), Random Forest (RF) and BERT to detect sarcasm in Twitter. We particularly focused on the importance of word representations on the performance of these classification techniques. In particular, tweets were represented using a TF-IDF vectorizer, word2vec pre-trained embeddings, and contextual word embeddings from BERT. This allowed us, to represent the tweets based on their relevance only (TF-IDF), semantic and syntactic relationship between words (word2vec), and finally based on the contextual information for words (BERT).

The rest of the paper is structured as follows: in sec. 2 we looked into related work on sarcasm detection, in sec. 3 we provided detailed information about the dataset used for this study, preprocessing steps and the models used for classification. The results of this study along with conclusions and proposed future work are presented in sec. 4.

2 Related Work

Various techniques have been used to detect sarcasm from text. In this section, we will look into five main techniques.

2.1 Rule-Based Approaches

In this approach, first a set of rules as indicators of sarcasm are proposed. These rules could be based on semantics, pragmatics or any other statistical-based methods. Evidence captured by the proposed rules are used to label unlabeled data as sarcastic or non-sarcastic. For example, (Maynard and Greenwood, 2014) proposed a hashtag-based rule to identify sarcasm from tweets. One of their proposed rules is to label a positive/neutral tweet with #sarcasm as a sarcastic tweet. In another study, Reyes. et al. (Reyes et al., 2013) proposed a model for

sarcasm detection based on conceptual descriptors of sarcasm such as signatures (punctuation marks and emoticons), unexpectedness of contextual imbalances, style and emotional scenarios.

2.2 Lexicon-Based Approaches

This approach relies on the sentiment of the text to decide whether the text is sarcastic or not. First, sentiment analysis is performed on the text and then sentiments that are associated with sarcasm such as exaggeration are used to label the text. Lexicon-based approach itself is categorized into dictionary-based and corpus-based approaches (Dave and Desai, 2016). In the dictionary-based method, only the sentiment of the individual tokens are considered. However, in the corpus-based method contextual relation of words are also considered. Riloff et. al. (Riloff et al., 2013) suggested that the contrast between positive sentiment of a tweet with negative situation is an indicator of sarcasm. In another study, authors leverage behavioral characteristics of a user extracted from the user's past tweets to identify sarcasm (Rajadesingan et al., 2015). They used a lexicon-based tool called SentiStrentgh (Thelwall et al., 2010) to compute the sentiment score of a tweet from its individual words.

2.3 Machine Learning

Classical machine learning algorithm such as Decision Tree (DT), RF, SVM, etc. have extensively been used to identify sarcasm. This technique automatically learns the features that are associated with sarcasm to classify text into sarcastic and non-sarcastic. Features include but not limited to syntactic (part-of-speech tagging), pragmatic (emojis, capitalization, ...) , hyperbole features (punctuantion marks, quotes, ...), term-frequency (TF), term frequency-inverse document frequency (TF-IDF), doc2vec (Le and Mikolov, 2014), bag-of-words (BoW) or word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2016; Howard and Ruder, 2018; Peters et al., 2018).

2.4 Deep Learning

Within this approach text is usually separated into tokens and these token are fed as the input to the model. The model is able to capture the hidden dependencies between tokens. Commonly used deep learning (DL) methodologies for sarcasm detection are recurrent neural networks (RNN) (Hiai and Shimada, 2019; Haoxiang, 2020), convolutional neural networks (CNN) (Ren et al., 2018; Mishra et al., 2017), long short-term memory (LSTM) (Ghosh et al., 2018), BiLSTM (Kumar et al., 2020) and Transformers (Kumar et al., 2021; Eke et al., 2021; Savini and Caragea, 2022; Khatri et al., 2020; Parameswaran et al., 2021).

2.5 Hybrid Models

In order to enhance the performance of the classification and/or incorporate the unique properties and strengths of different classifiers, a combination of two or more machine learning or deep learning models is used. Ashok et. al. (Ashok et al., 2020) reported that combination of a CNN network with an LSTM layer, increased the accuracy of CNN only model by $\sim 5\%$. In a similar study, Ghosh and Veale (Ghosh and Veale, 2016) observed state-of-the-art performance on their CNN+LSTM+DNN model. Kumar and Anand (Kumar and Anand, 2020) showed that addition of an LSTM layer to the RoBERTa model improved their F-1 score on a sarcasm detection shared task.

3 Methodology

In this section we will discuss the dataset, preprocessing steps, models and evaluation metrics used in this study in detail.

3.1 Dataset Description

The dataset used in this study was provided at the International Workshop on Semantic Evaluation 2018 (Van Hee et al., 2018). During the data collection process, first 3000 English tweets were collected from Twitter using #sarcasm, #irony and #not. The dataset consists of 1728 instances of verbal irony by means of a polarity contrast and 401 instances of situational irony. The collected corpus was then annotated manually by three students. After this step, $\sim$ 2400 tweets were labeled as sarcastic and $\sim$ 600 were labeled as non-sarcastic. To balance the class representations, $\sim$ 1800 non-sarcastic were added to the corpus. The final training and test dataset used in this study have 3817 (1916 sarcastic and 1901 non-sarcastic tweets) and 784 instances, respectively.

3.2 Data Preprocessing

First, corpus was converted to lowercase. Then, user handles, URLs, hashtags, special characters and punctuations were removed. Using *contractions* library, contractions were converted to long format (e.g., "I'm" was converted to "I am"). Using

nltk library, stopwords were removed and tokens were lemmatized and stemmed. To preprocess the corpus for BERT classifier, corpus was converted to lowercase, user handles and URLs were removed and contractions were converted to long formats. This was done to keep as much information as possible for BERT classifier.

3.3 Approach

In this paper, we focused on the importance of word representation on the performance of sarcasm detection in Twitter. To this end, we chose three different representations. First, after preprocessing tokens were converted to a matrix of TF-IDF features using *TfidfVectorizer* from *sklearn* library. TF-IDF measures the relevance of a given token within a document amongst a collection of documents. TF measures the frequency of a token within a document and IDF provides information on the rarity of a token amongst the collection of documents. Therefore, giving less weight to the more frequent words. TF-IDF is not able to capture the semantic relations between words. It also does not capture the word orders. Finally, if the vocabulary size becomes large, it will suffer from memory inefficiency.

The second choice for representing the words is word2vec (Mikolov et al., 2013). Word2vec word embeddings were taken from tensorflow hub (see acknowledgment). These embeddings are derived by training the skipgram version of word2vec trained on English Wikipedia which converts text into a 250-dimensional embedding vector. Unlike TF-IDF, word2vec is able to capture the syntactic and semantic relationship of the words. Given enough training data, word2vec is able to capture the meaning of the words and assign the vectors of similar words in a close vicinity in the vector-space. The main disadvantage of word2vec is that since it is a static model it cannot assign vectors to polysemous words dynamically based on the context. It also does not handle the out-of-vocabulary words well.

The final choice to represent the words is contextual word embeddings from pre-trained BERT model (Devlin et al., 2018). The pre-trained model was downloaded from tensorflow hub (see acknowledgement). The architecture of the BERT consists of multiple encoders from transformers. Pre-trained BERT model used in this study was trained on English Wikipedia and BooksCorpus. During the training process, BERT reads all of the input token simultaneously and therefore makes BERT a true bidirectional model. Bidirectional models are able to capture the context from both sides and therefore they have a better understanding of the language. Unlike word2vec, BERT creates word embeddings dynamically. This enables BERT to create different word vectors based on the meaning of a word within a particular context. The details of the parameters used in this model are provided in the classification section.

3.3.1 Classification

To classify the test dataset using the word representations obtained from TF-IDF and word2vec, three classical machine learning algorithms namely SVM, RF and multi-layer perceptron (MLP) were chosen. The objective of SVM classifier is to distinctly classify sarcastic tweets from non-sarcastic tweet by finding a hyperplane in an N-dimensional vector space that separates the two class from each other. For SVM, *sklearn* library was used with radial basis function (RBF) kernel. Probability estimates was set to *True* to enable 5-fold cross-validation. The second classical machine learning classifier used in this study is RF. RF employs an ensemble of decision trees to make the predictions. RF is a powerful technique that can perform reasonably without hyperparameter tuning. For RF *sklearn* library was used with default options. The last classifier used in this study is MLP. MLP is a fully connected feedforward neural network with at least three layers; input, at least one hidden layer and an output layer. For MLP classifier we used *sklearn* library with one hidden layer and 100 hidden units. Relu activation function with Adam optimizer was used. For classification with BERT, pre-trained BERT model with 12 hidden layers and 768 hidden units in each layer with 12 attention heads was downloaded from tensorflow hub. To convert text into suitable input for BERT, the companion preprocessor tool to BERT was downloaded from tensorflow hub. On top of BERT a *keras* dropout layer and a dense layer was added. Dropout layer prevents the model from overfitting. This particular model has $\sim$ 110 million parameters in which only 769 parameters are trainable via fine-tuning.

3.4 Evaluation

For assessing the performance of the classifiers in detecting sarcasm, we use Accuracy, Precision, Recall and and F-1 score. Accuracy determines

the number of correct predictions by the model. In a balanced dataset such as the one used in this study, Accuracy can be used as a helpful evaluation metric. Precision determines how many of the predicted positives are actually positive. Precision is particularly a good metric when the importance of false positive is higher than the importance of false negative (e.g., spam detection). Recall determines out of all of the actual positives, how many were successfully predicted as positive. This is particularly important when when the importance of false negative is higher than false positive (e.g., cancer detection). F-1 score is the harmonic mean of Recall and Precision. In cases in which false positives and false negatives are both important, F-1 score is usually used.

4 Results and Conclusion

The result of sarcasm detection using various classifiers are shown in Table 1. Among all of the techniques, TF-IDF + SVM performed the best with F-1 score of 0.67. SVM performed better than RF and MLP using the same word representations. Interestingly performance of all of the classifiers (SVM, RF and MLP) dropped when word2vec vector embeddings used instead of the TF-IDF vector. Both word2vec and BERT were trained on English Wikipedia. Wikipedia does not contain sarcastic text. In addition, it only consists of formal English which is drastically different than text used in Twitter. It seems like the difference between the training dataset for these models and the dataset used in this study rather hurts the performance of the classification. BERT (uncased) performed better than word2vec but worse than TF-IDF + SVM. This shows that contextual word embeddings perform better than static word embeddings.

Model	Acc	Pr	Re	F-1
TF-IDF+SVM	**0.66**	**0.67**	**0.66**	**0.67**
TF-IDF+RF	0.65	0.66	0.65	0.65
TF-IDF+MLP	0.64	0.67	0.64	0.65
word2vec+SVM	0.62	0.63	0.62	0.62
word2vec+RF	0.58	0.59	0.58	0.59
word2vec+MLP	0.60	0.62	0.60	0.60
BERT(uncased)	0.64	0.66	0.64	0.64

Table 1: Results on test dataset

It is worth to mention that our BERT model performed slightly worse than the one reported by Potamias et. al (Potamias et al., 2020). We are speculating this could be due to the difference in the test dataset. The number of test dataset instances are reported as 958 in the SemEval shared task website. However, the test dataset used in this study has 784 instances.

In conclusion, in this paper we studied sarcasm detection in Twitter. We particularly looked into the importance of word representations in sarcasm detection. Words were represented using TF-IDF, word2vec and BERT models. Multiple classifiers such as SVM, RF and MLP were utilized. Surprisingly, we observed that word embeddings from word2vec or BERT performed worse than a simple method namely TF-IDF. We believe this could be due to the inherent differences between Wikipedia which word2vec and BERT were trained on and the text used in Twitter. To further, assess this hypothesis, it would be interesting to use a pre-trained transformer model trained on Twitter dataset and compare the results to the ones reported in this study.

Acknowledgements

Word2vec embeddings were taken from https://tfhub.dev/google/Wiki-words-250/2. Pre-trained BERT model was taken from https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4. Pre-processing tool for BERT model is available at https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3.

References

Darkunde Mayur Ashok, Agrawal Nidhi Ghanshyam, Sayed Saniya Salim, Dungarpur Burhanuddin Mazahir, and Bhushan S. Thakare. 2020. Sarcasm detection using genetic optimization on lstm with cnn. In *2020 International Conference for Emerging Technology (INCET)*, pages 1–4.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information.

Anandkumar D. Dave and Nikita P. Desai. 2016. A comprehensive study of classification techniques for sarcasm detection on textual data. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 1985–1991.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep

bidirectional transformers for language understanding.

Christopher Ifeanyi Eke, Azah Anir Norman, and Liyana Shuib. 2021. Context-based feature technique for sarcasm identification in benchmark datasets using deep learning and bert model. *IEEE Access*, 9:48501–48518.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 161–169, San Diego, California. Association for Computational Linguistics.

Debanjan Ghosh, Alexander R. Fabbri, and Smaranda Muresan. 2018. Sarcasm Analysis Using Conversation Context. *Computational Linguistics*, 44(4):755–792.

Dr Wang Haoxiang. 2020. Emotional analysis of bogus statistics in social media. *September 2020*, 2(3):178–186.

Satoshi Hiai and Kazutaka Shimada. 2019. Sarcasm detection using rnn with relation vector. *Int. J. Data Warehous. Min.*, 15(4):66–78.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification.

Akshay Khatri, Pranav P, and Dr. Anand Kumar M. 2020. Sarcasm detection in tweets with bert and glove embeddings.

Amardeep Kumar and Vivek Anand. 2020. Transformers on sarcasm detection with context. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 88–92, Online. Association for Computational Linguistics.

Avinash Kumar, Vishnu Teja Narapareddy, Veerubhotla Aditya Srikanth, Aruna Malapati, and Lalita Bhanu Murthy Neti. 2020. Sarcasm detection using multi-head attention based bidirectional lstm. *IEEE Access*, 8:6388–6397.

Avinash Kumar, Vishnu Teja Narapareddy, Pranjal Gupta, Veerubhotla Aditya Srikanth, Lalita Bhanu Murthy Neti, and Aruna Malapati. 2021. Adversarial and auxiliary features-aware bert for sarcasm detection. In *8th ACM IKDD CODS and 26th COMAD*, CODS COMAD 2021, page 163–170, New York, NY, USA. Association for Computing Machinery.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

Diana Maynard and Mark Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4238–4243, Reykjavik, Iceland. European Language Resources Association (ELRA).

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Abhijit Mishra, Kuntal Dey, and Pushpak Bhattacharyya. 2017. Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 377–387, Vancouver, Canada. Association for Computational Linguistics.

Pradeesh Parameswaran, Andrew Trotman, Veronica Liesaputra, and David Eyers. 2021. BERT's the word : Sarcasm target detection using BERT. In *Proceedings of the The 19th Annual Workshop of the Australasian Language Technology Association*, pages 185–191, Online. Australasian Language Technology Association.

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, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Rolandos Alexandros Potamias, Georgios Siolas, and Andreas Georgios Stafylopatis. 2020. A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, 32(23):17309–17320.

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*, WSDM '15, page 97–106, New York, NY, USA. Association for Computing Machinery.

Yafeng Ren, Donghong Ji, and Han Ren. 2018. Context-augmented convolutional neural networks for twitter sarcasm detection. *Neurocomputing*, 308:1–7.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Lang. Resour. Eval.*, 47(1):239–268.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714, Seattle, Washington, USA. Association for Computational Linguistics.

Edoardo Savini and Cornelia Caragea. 2022. Intermediate-task transfer learning with bert for sarcasm detection. *Mathematics*, 10(5).

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

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*, pages 39–50, New Orleans, Louisiana. Association for Computational Linguistics.