Neural Sarcasm Detection using Conversation Context

Nikhil Jaiswal
TCS Research, New Delhi, India
nikhil.jais@tcs.com

Abstract

Social media platforms and discussion forums such as Reddit, Twitter, etc. are filled with figurative languages. Sarcasm is one such category of figurative language whose presence in a conversation makes language understanding a challenging task. In this paper, we present a deep neural architecture for sarcasm detection. We investigate various pre-trained language representation models (PLRMs) like BERT, RoBERTa, etc. and fine-tune it on the Twitter dataset[1]. We experiment with a variety of PLRMs either on the twitter utterance in isolation or utilizing the contextual information along with the utterance. Our findings indicate that by taking into consideration the previous three most recent utterances, the model is more accurately able to classify a conversation as being sarcastic or not. Our best performing ensemble model achieves an overall F1 score of 0.790, which ranks us second[2] on the leaderboard of the Sarcasm Shared Task 2020.

1 Introduction

Sarcasm can be defined as a communicative act of intentionally using words or phrases which tend to transform the polarity of a positive utterance into its negative counterpart and vice versa. The significant increase in the usage of social media channels has generated content that is sarcastic and ironic in nature. The apparent reason for this is that social media users tend to use various figurative language forms to convey their message. The detection of sarcasm is thus vital for several NLP applications such as opinion minings, sentiment analysis, etc (Maynard and Greenwood, 2014). This leads to a considerable amount of research in the sarcasm detection domain among the NLP community in recent years.

The Shared Task on Sarcasm Detection 2020 aims to explore various approaches for sarcasm detection in a given textual utterance. Specifically, the task is to understand how much conversation context is needed or helpful for sarcasm detection. Our approach for this task focuses on utilizing various state-of-the-art PLRMs and fine-tuning it to detect whether a given conversation is sarcastic. We apply an ensembling strategy consisting of models trained on different length conversational contexts to make more accurate predictions. Our best performing model **(Team name - nclabj)** achieves an F1 score of 0.790 on the test data in the CoadaLab evaluation platform.

2 Problem Description

The dataset assigned for this task is collected from the popular social media platform, Twitter. Each training data contains the following fields: "label" (i.e., "SARCASM" or "NOTSARCASM"), "response" (the Tweet utterance), "context" (i.e., the conversation context of the "response"). Our objective here is to take as input a response along with its optional conversational context and predict whether the response is sarcastic or not. This problem can be modeled as a binary classification task. The predicted response on the test set is evaluated against the true label. Three performance metrics, namely, Precision, Recall, and F1 Score are used for final evaluation.

3 Related Work

Various attempts have been made for sarcasm detection in recent years (Joshi et al., 2017). Researchers have approached this task through different methodologies, such as framing it as a sense disambigua-

[1]The dataset is provided by the organizers of Sarcasm Shared Task FigLang-2020

[2]We are ranked 8th with an F1 score of 0.702 on the Reddit dataset leaderboard using the same approach. But we do not describe those results here as we could not test all our experiments within the timing constraints of the Shared Task.

Proceedings of the Second Workshop on Figurative Language Processing, pages 77–82

https://doi.org/10.18653/v1/P17

tion problem (Ghosh et al., 2015) or considering sarcasm as a contrast between a positive sentiment and negative situation (Riloff et al., 2013; Maynard and Greenwood, 2014; Joshi et al., 2015, 2016b; Ghosh and Veale, 2016). Recently, few works have taken into account the additional context information along with the utterance. (Wallace et al., 2014) demonstrate how additional contextual information beyond the utterance is often necessary for humans as well as computers to identify sarcasm. (Schifanella et al., 2016) propose a multi-modal approach to combine textual and visual features for sarcasm detection. (Joshi et al., 2016a) model sarcasm detection as a sequence labeling task instead of a classification task. (Ghosh et al., 2017) investigated that the conditional LSTM network (Rocktäschel et al., 2015) and LSTM networks with sentence-level attention on context and response achieved significant improvement over the LSTM model that reads only the response. Therefore, the new trend in the field of sarcasm detection is to take into account the additional context information along with the utterance. The objective of this Shared Task is to investigate how much of the context information is necessary to classify an utterance as being sarcastic or not.

4 System Description

We describe our proposed system for sarcasm detection in this section. We frame this problem as a binary classification task and apply a transfer learning approach to classify the tweet as either sarcastic or not. We experiment with several state of the art PLRMs like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), as well as pre-trained embeddings representations models such as ELMo (Peters et al., 2018), USE (Cer et al., 2018), etc. and fine-tune it on the assigned Twitter dataset. We briefly review these models in subsections 4. For fine-tuning, we add additional dense layers and train the entire model in an end to end manner. Figure 1 illustrates one such approach for fine-tuning a RoBERTa model. We sequentially unfreeze the layers with each ongoing epoch. We apply a model ensembling strategy called "majority voting", as shown in Figure 2 to come out with our final predictions on the test data. In this ensemble technique, we take the prediction of several models and choose the label predicted by the maximum number of models.

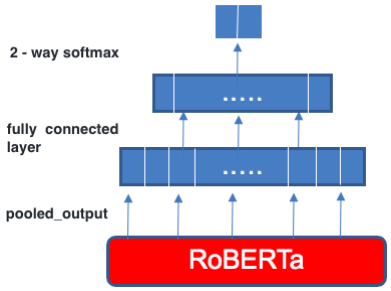

Figure 1: The proposed methodology to fine-tune a RoBERTa model

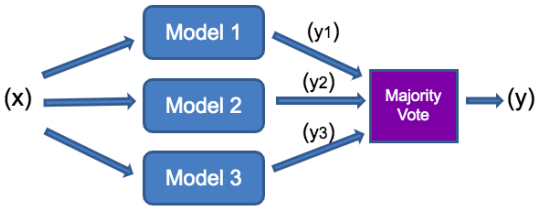

Figure 2: The majority voting ensemble methodology consisting of three sample models

4.1 Embeddings from Language Models (ELMo)

ELMo introduces a method to obtain deep contextualized word representation. Here, the researchers build a bidirectional Language model (biLM) with a two-layered bidirectional LSTM architecture and obtain the word vectors through a learned function of the internal states of biLM. This model is trained on 30 million sentence corpus, and thus the word embeddings obtained using this model can be used to increase the classification performance in several NLP tasks. For our task, we utilize the ELMo embeddings to come out with a feature representation of the words in the input utterance and pass it through three dense layers to perform the binary classification task.

4.2 Universal Sentence Encoder (USE)

USE presents an approach to create embedding vector representation of a complete sentence to specifically target transfer learning to other NLP tasks. There are two variants of USE based on trade-offs in compute resources and accuracy. The first variant uses an encoding sub-graph of the transformer architecture to construct sentence embeddings (Vaswani et al., 2017) and achieve higher performance figures. The second variant is a light model that uses a deep averaging network (DAN) (Iyyer et al., 2015) in which first the input embed-

ding for words and bi-grams are averaged and then passed through a feedforward neural network to obtain sentence embeddings. We utilize the USE embeddings from the Transformer architecture on our data and perform the classification task by passing them through three dense layers.

4.3 Bidirectional Encoder Representations from Transformers (BERT)

BERT, a Transformer language model, achieved state-of-the-art results on eleven NLP tasks. There are two pre-training tasks on which BERT is trained on. In the first task, also known as masked language modeling (MLMs), 15% of words are randomly masked in each sequence, and the model is used to predict the masked words. The second task, also known as the next sentence prediction (NSP), in which given two sentences, the model tries to predict whether one sentence is the next sentence of the other. Once the above pre-training phase is completed, this can be extended for classification related task with minimal changes. This is also known as BERT fine-tuning, which we apply for our sarcasm detection task. In the paper, two models ($\text{BERT}_{\text{BASE}}$ & $\text{BERT}_{\text{LARGE}}$) are released depending on the number of transformer blocks (12 vs. 24), attention heads (12 vs. 16), and hidden units size (768 vs. 1024). We experiment with $\text{BERT}_{\text{LARGE}}$ model for our task, since it generally performs better as compared to the $\text{BERT}_{\text{BASE}}$ model.

4.4 Robustly Optimized BERT Approach (RoBERTa)

RoBERTa presents improved modifications for training BERT models. The modifications are as follows: 1. training the model for more epochs (500K vs. 100K) 2. using bigger batch sizes (around 8 times) 3. training on more data (160GB vs. 16 GB). Apart from the above parameters changes, byte-level BPE vocabulary is used instead of character-level vocabulary. The dynamic masking technique is used here instead of the static masking used in BERT. Also, the NSP task is removed following some recent works that have questioned the necessity of the NSP loss (Sun et al., 2019; Lample and Conneau, 2019). Summarizing, RoBERTa is trained with dynamic masking, sentences without NSP loss, large batches, and a larger byte-level BPE.

Notation	Description	Seq. Length
RESP	only response	70
CON1	previous 1 turn followed by response	130
CON2	previous 2 turns followed by response	180
CON3	previous 3 turns followed by response	230
CON	entire context followed by response	450

Table 1: Different Variants of Data

5 Experiments and Results

5.1 Dataset Preparation

The dataset assigned for this task is collected from Twitter. There are 5,000 English Tweets for training, and 1,800 English Tweets for testing purpose. We use 10% of the training data for the validation set to tune the hyper-parameters of our model.We apply several preprocessing steps to clean the given raw data. Apart from the standard preprocessing steps such as lowercasing the letters, removal of punctuations and emojis, expansion of contractions, etc., we remove the usernames from the tweets. Also, since hashtags generally consist of phrases in CamelCase letters, we split them into individual words since they carry the essential information about the tweet.

To incorporate contextual information along with a given tweet, we prepare the data in the manner, as shown in Table 1. For data in which only the previous two turns are available, for them, only those two turns are considered in CON3 & CON case illustrated in Table 1. We fix the maximum sequence length based on the coverage of the data (greater than 90th percentile) in the training and test set. This sequence length is determined by considering each word as a single token.

5.2 Implementation Details

Here, we describe a detailed set up of our experiments and the different hyper-parameters of our models for better reproducibility. We experiment with various advanced state-of-the-art methodologies such as ELMo, USE, BERT, and RoBERTa. We use the validation set to tune the hyper-parameters. We use Adam (Kingma and Ba, 2014) optimizer in all our experiments. We use dropout regularization (Srivastava et al., 2014) and early stopping (Yao et al., 2007) to prevent overfitting. We use a batch size of $\{2, 4, 8, 16\}$ depending on the model size and the sequence length.

Firstly, the data is prepared as mentioned in subsection 5.1. For fine-tuning ELMo, USE, and $\text{BERT}_{\text{LARGE}}$ models, we use the module from Ten-

Model	ELMo			USE			$\text{BERT}_{\text{LARGE}}$		
Data	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**
RESP	0.688	0.690	0.688	0.728	0.728	0.728	0.720	0.735	0.716
CON1	0.677	0.679	0.676	0.718	0.718	0.718	0.714	0.719	0.712
CON2	0.652	0.670	0.643	0.715	0.715	0.715	0.729	0.731	0.729
CON3	0.697	0.697	0.697	0.719	0.725	0.717	0.741	0.758	0.737
CON	0.699	0.699	0.699	0.704	0.705	0.704	0.734	0.734	0.734

Table 2: We compare the fine-tuning of different individual models **ELMo** and **USE** and $\textbf{BERT}_{\textbf{LARGE}}$ on different variants of Twitter test data. The metric Precision (Pr), Recall (Re) and F1 Score (F1) denotes the test set results.

Model	$\text{RoBERTa}_{\text{LARGE}}$ A				$\text{RoBERTa}_{\text{LARGE}}$ B			
Data	**ID**	**Pr**	**Re**	**F1**	**ID**	**Pr**	**Re**	**F1**
RESP	**1**	0.742	0.745	0.742	**6**	0.744	0.744	0.744
CON1	**2**	0.751	0.756	0.750	**7**	0.752	0.753	0.751
CON2	**3**	0.751	0.751	0.750	**8**	0.763	0.764	0.763
CON3	**4**	**0.773**	**0.778**	**0.772**	**9**	0.766	0.766	0.766
CON	**5**	0.759	0.760	0.759	**10**	0.757	0.757	0.757

Table 3: We compare the fine-tuning of $\textbf{RoBERTa}_{\textbf{LARGE}}$ model on different variants of Twitter test data. We fine-tune this model twice on the same train and validation data with different weight initialization. We represent each of these results with a unique ID to utilize them in the ensemble network.

Description	Models IDs	Pr	Re	F1
Top 3 RoBERTa A	**3, 4, 5**	0.773	0.775	0.772
Top 3 RoBERTa B	**8, 9, 10**	0.778	0.779	0.778
Top 3 RoBERTa A & B	**4, 8, 9**	**0.790**	**0.792**	**0.790**
Top 5 RoBERTa A & B	**4, 5, 8, 9, 10**	0.788	0.789	0.787

Table 4: We compare the ensembling results based on several combinations of $\text{RoBERTa}_{\text{LARGE}}$ models. Bold font denotes the best results.

sorflow Hub [3][4][5] and wrap it in a Keras Lambda layer whose weights are also trained during the fine-tuning process. We add three dense layers {512, 256, 1} with a dropout of 0.5 between these layers. The relu activation function is being applied between the first two layers whereas sigmoid is used at the final layer. ELMo and USE models are trained for 20 epochs while $\text{BERT}_{\text{LARGE}}$ is trained for 5 epochs. During the training, only the best model based on the minimum validation loss was saved by using the Keras ModelCheckpoint callback. Instead of using a threshold value of 0.5 for binary classification, a whole range of threshold values from 0.1 to 0.9 with an interval of 0.01 is experimented on the validation set. The threshold value for which the highest validation accuracy is obtained is selected as the final threshold and is being applied on the test set to get the test class predictions.

For fine-tuning $\text{RoBERTa}_{\text{LARGE}}$ model, we use the fastai (Howard and Gugger, 2020) framework and utilize PLRMs from HuggingFace's Transformers library (Wolf et al., 2019). HuggingFace library contains a collection of state-of-the-art PLRMs which is being widely used by the researcher and practitioner communities. Incorporating HuggingFace library with fastai allows us to utilize powerful fastai capabilities such as Discriminate Learning Rate, Slanted Triangular Learning Rate and Gradual Unfreezing Learning Rate on the powerful pretrained Transformer models. For our experiment, first of all, we extract the pooled output (i.e. the last layer hidden-state of the first token of the sequence (CLS token) further processed by a linear layer and a Tanh activation function). It is then passed through a linear layer with two neurons followed by a softmax activation function. We use a learning rate of 1e-5 and utilize the "1cycle" learning rate policy for super-convergence, as suggested by (Smith, 2015). We gradually unfreeze the layers and train on a 1cycle manner. After unfreezing the last three layers, we unfreeze the entire layers

[3] https://tfhub.dev/google/elmo/2

[4] https://tfhub.dev/google/universal-sentence-encoder-large/3

[5] https://tfhub.dev/tensorflow/bert_en_uncased_L-24_H-1024_A-16/1

and train in the similar 1cycle manner. We stop the training when the validation accuracy does not improve consecutively for three epochs.

We use a simple ensembling technique called majority voting to ensemble the predictions of different models to further improvise the test accuracy.

5.3 Results and Error Analysis

Here, we compare and discuss the results of our experiments. First, we summarize the results of the individual model on the test set using different variants of data in Tables 2 & 3. From Table 2, we can observe that adding context information of specific lengths helps in improving the classification performance in almost all the models. USE results are better as compared to the ELMo model since Transformers utilized in the USE are able to handle sequential data comparatively better than that as LSTMs being used in ELMo. On the other hand, $\text{BERT}_{\text{LARGE}}$ outperforms USE with the increase in the length of context history. The highest test accuracy by $\text{BERT}_{\text{LARGE}}$ is obtained on the CON3 variant of data which depicts the fact that adding most recent three turns of context history helps the model to classify more accurately. This hypothesis is further supported from the experiments when a similar trend occurs with the $\text{RoBERTa}_{\text{LARGE}}$ model. Since the results obtained by RoBERTa are comparatively better than other models, we train this model once again on the same train and validation data with different weight initialization. By doing this, we can have a variety of models to build our final ensemble architecture. The evaluation metrics used are Precision (Pr), Recall (Re), F1-score (F1).

As observed in Table 3, RoBERTa fine-tuned on the CON3 variant of data outperforms all other approaches. In the case of fine-tuning PLRMs like $\text{BERT}_{\text{LARGE}}$ & $\text{RoBERTa}_{\text{LARGE}}$ on this data, we can observe the importance of most recent three turns of context history. From the experiments, we conclude that on increasing the context history along with the utterance, the model can learn a better representation of the utterance and can classify the correct class more accurately. Finally, RoBERTa model outperforms every other model because this model is already an optimized and improved version of the BERT model.

Table 4 summarizes the results of our various ensemble models. For ensembling, we choose different variants of best performing models on the test data and apply majority voting on it to get the final test predictions. We experiment with several combinations of the models and report here the results of some of the best performing ensembles. We can observe that the ensemble model consisting of top three individual models gave us the best results.

6 Conclusion & Future Work

In this work, we have presented an effective methodology to tackle the sarcasm detection task on the twitter dataset by framing it as a binary classification problem. We showed that by fine-tuning PLRMs on a given utterance along with its specific length context history, we could successfully classify the utterance as being sarcastic or not. We experimented with different length context history and concluded that by taking into account the most recent three conversation turns, the model was able to obtain the best results. The fine-tuned $\text{RoBERTa}_{\text{LARGE}}$ model outperformed every other experimented models in terms of precision, recall, and F1 score. We also demonstrated that we could obtain a significant gain in the performance by using a simple ensembling technique called majority voting.

In the future, we would like to explore with these PLRMs on other publicly available datasets. We also aim to dive deep into the context history information and derive insights about the contextual part, which helps the model in improvising the classification result. We also wish to investigate more complex ensemble techniques to observe the performance gain.

References

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Jacob Devlin et al. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network.

Debanjan Ghosh, Alexander Richard Fabbri, and Smaranda Muresan. 2017. The role of conversation

context for sarcasm detection in online interactions. *CoRR*, abs/1707.06226.

Debanjan Ghosh, Weiwei Guo, and Smaranda Muresan. 2015. Sarcastic or not: Word embeddings to predict the literal or sarcastic meaning of words. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1003–1012, Lisbon, Portugal. Association for Computational Linguistics.

Jeremy Howard and Sylvain Gugger. 2020. Fastai: A layered api for deep learning. *Information*, 11(2):108.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. 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)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Aditya Joshi, Pushpak Bhattacharyya, Mark Carman, and Vaibhav Tripathi. 2016a. Harnessing sequence labeling for sarcasm detection in dialogue from tv series 'friends'.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J. Carman. 2017. Automatic sarcasm detection: A survey. *ACM Comput. Surv.*, 50(5).

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 757–762, Beijing, China. Association for Computational Linguistics.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark James Carman. 2016b. Are word embedding-based features useful for sarcasm detection? *CoRR*, abs/1610.00883.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

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).

Matthew Peters, Mark Neumann, et al. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

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.

Tim Rocktäschel, Edward Grefenstette, Karl Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention.

Rossano Schifanella, Paloma Juan, Joel Tetreault, and Liangliang Cao. 2016. Detecting sarcasm in multimodal social platforms.

Leslie N. Smith. 2015. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186.

Nitish Srivastava, Geoffrey E Hinton, et al. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Byron C. Wallace, Do Kook Choe, Laura Kertz, and Eugene Charniak. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–516, Baltimore, Maryland. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. 2007. On early stopping in gradient descent learning. *Constructive Approximation*, 26:289–315.