Project Report: Sarcasm Detection

Lydia Xu
leediaxu@stanford.edu

Vera Xu
veraxrl@stanford.edu

Abstract

Sentiment analysis is often used in NLP to understand people's subjective opinions. However, the analysis results may be biased if people use sarcasm in their statements. In order to correctly understand people's true intention, being able to detect sarcasm is critical. Many previous models have been developed to detect sarcasm based on the utterances in isolation, meaning only the reply text itself. Ghosh et al. [2017]'s research used both context and reply texts to detect sarcasm in reply and showed great improvement in performance. The main models used were SVMs and LSTMs with attention. In this project, we aim to improve accuracy of sarcasm detection by further exploring the role of contextual information in detecting sarcasm. We will investigate the performances of different models (various LSTM models and BERT) and compare the models using reply-only, context-only and context-and-reply data setups. BERT, short for Bidirectional Encoder Representations from Transformers [Devlin et al., 2018], has never been used for this specific task of sarcasm detection before. We show that a simple BERT classifier provides surprisingly good results for this task and analyze how the sarcasm detection task can benefit from the pretraining on other, related tasks that distinguishes BERT.

1 Introduction

Sarcasm, a sharp and ironic utterance designed to cut or to cause pain, is often used to express strong emotions, such as contempt, mockery or bitterness. Sarcasm detection is of great importance in understanding people's true sentiments and opinions. Application of sarcasm detection can benefit many areas of interest of NLP applications, including marketing research, opinion mining and information categorization. However, sarcasm detection is also a very difficult task, as it's largely dependent on context, prior knowledge and the tone in which the sentence was spoken or written.

While most sarcasm detection models so far consider the utterance in isolation, we investigated in the role of conversational context information in sarcasm detection. Specifically, we explored how state-of-the-art NLP models can make use of such context information in improving classification accuracy and precision. In this project, we followed the various LSTM models explained in Ghosh et al. [2017] and re-implemented the approaches in PyTorch. The models we implemented include vanila LSTM, attention-based single LSTM, attention-based stacked LSTM and conditional LSTM. On top of the implementations, we paid particular attention to hyper-parameters tuning and understanding the behavior of these various LSTM models. We further conducted a qualitative analysis on the effect of learning rate, number of hidden states and total number of epochs on sarcasm detection performance.

Last but not least, we adapted the off-the-shelve BERT-based classifier model by Huggingface [2019] and ran the model on both Discussion Forum data [Oraby et al., 2016] and Reddit data [Khodak et al., 2018]. The BERT-based model takes a long time to train but performed very well beyond both the baseline and all the other LSTM models we tried. With very little hyper-parameters tuning, BERT clearly exhibits the benefits of utilizing pretraining models to obtain state-of-the-art results.

2 Related Work

This project is heavily inspired by Ghosh et al. [2017], who thoroughly explored the role of conversational context and various LSTM models in sarcasm detection. Prior to this paper, Ghosh et al. has also explored on the topic of sarcasm detection quite a bit, including his previous publication at EMNLP [Debanjan Ghosh and Muresan., 2015], which serves as the baseline for his paper in 2017 [Ghosh et al., 2017]. For this class's final project, we intentionally choose his 2017 paper to establish baseline, as it contains more mature SVM results and more deep learning model applications. Besides these papers, we referenced sentence-level attention by Yang et al. in our implementation of attention-based model. [Yang, 2016] Their paper discusses the concept of hierarchical attention network and applies attention on both word-level and sentence-level. Even though we didn't end up having time to implement word-level attention, this paper inspires future work. Lastly, the BERT paper offers insights into the power of pre-trained transformer models, its adaptation and training processes and its state-of-the-art high-performing results [Devlin et al., 2018].

3 Approach

3.1 Baseline

For our baselines, we referred to the scores obtained by `SVM` model conducted by Ghosh et al. [2017] using Discussion Forum data (Table 1). Ghosh et al. [2017] utilized n-grams, lexicon-based features and sarcasm indicators to perform two binary classification tasks: one for sarcastic instances S and one for non-sarcastic instances NS. The model is trained for both reply-only and context-and-reply data. The results provided in the paper only include Precision, Recall and F1, but we also add Accuracy as one of our performance measurements.

Table 1: SVM model with Discussion Forum data results

	Sarcasm (S)			Non-Sarcasm (NS)		
	Precision	Recall	F1	Precision	Recall	F1
SVM^{r}	65.55	66.67	66.10	66.10	64.96	65.52
SVM^{c+r}	63.22	61.97	62.63	62.77	64.10	63.5

3.2 Models

We mainly applied two models for this task: `LSTM` and `BERT`. We implemented all `LSTM` models ourselves (with references to several online LSTM implementations), and used the existing `pytorch-pretrained-BERT` package and its example classifier trainer and evaluation [Huggingface, 2019]. The word embedding reference we used is GloVe [Pennington et al., 2014].

3.2.1 LSTM-based models

We implemented three types of `LSTM` models and all of them are unidirectional (from left to right). We also implemented our own `ModelEmbedding` class to represent the sentence-embedding as the average of all of its word embeddings. Similar to default project, we use the pretrained Glove as our word embeddings.

1. **Vanilla LSTM**:
This is a simple Vanilla LSTM model with no attention. It only considers the response word embeddings.

2. **Conditional LSTM**:
This model contains two `LSTM` layers, one for the context inputs ($LSTM_{context}$), the other for the reply inputs ($LSTM_{reply}$). To condition $LSTM_{reply}$ on the representation of $LSTM_{context}$, the memory state of $LSTM_{reply}$ is initialized specifically with the last hidden state of $LSTM_{context}$. Figure 1 shows the high-level architecture of this model.

3. **LSTM with sentence-level attention**:
This model gives different attention weights for each sentence when training forward. When we process the contexts and responses, we can do two levels of attentions: word level and sentence

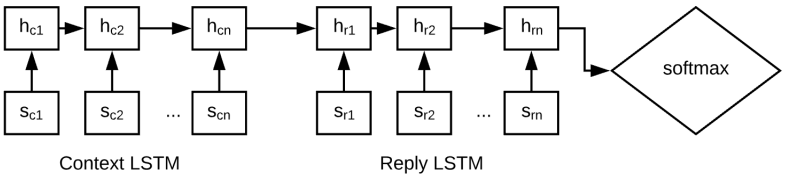


Figure 1: Conditional LSTM with context and reply

level. There are three kinds of combinations given these attention levels: word-level only, sentence-level only, and word-and-sentence-level together. We only explored sentence-level attention for the following reasons: (1) Based on the results generated by Ghosh et al. [2017], sentence-level attention gives better performance. (2) Sentence-level information helps understand which sentence triggers the sarcasm reply. (3) Given the strict timeline of this project, we weren't able to implement other alternatives.

Figure 2 shows the high-level architecture of this model. We have two separate LSTMs with sentence-level attention, for context and reply respectively. The model first apply LSTM to sentences to generate hidden states. To reward sentences that are more important, u_s is developed to represent all information in the sentences of a document. It's randomly initialized and jointly learned during the training process. Each hidden state is then feeded into a Linear layer and non-linearity `tanh` to generate u_i.

The model uses both u_i and u_s to generate attention scores, and the attention scores are used to calculate the weighted sum v of different hidden states, which summarizes all the information of sentences in a document. This process is done for context and reply separately. The two vectors for context (v_c) and response (v_r) are then concatenated and put through a single feed-forward linear layer and softmax to obtain two probability values for each document, one for sarcastic label and another for non-sarcastic label.

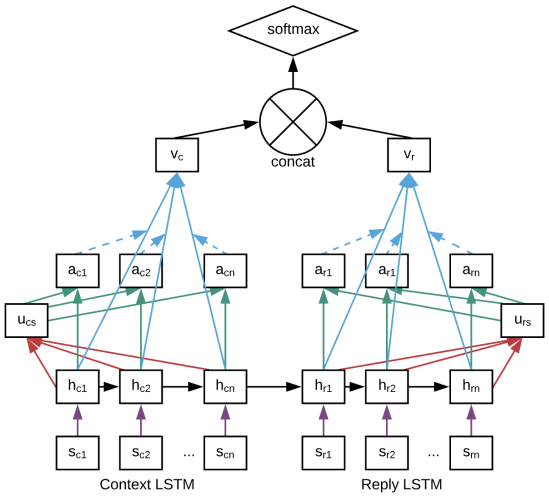


Figure 2: Sentence-level attention LSTM with context and reply (Modified based on Ghosh et al. [2017] and Yang [2016])

The main mathematical equation used in attention-based `LSTM` is [Yang, 2016]:

$$v = \sum_{i \in [1,d]} \alpha_i h_i$$

$$\alpha_i = \frac{exp(u_i^T u_s)}{\sum_{i \in [1,d]} exp(u_i^T u_s)}$$

$$u_i = tanh(W_s h_i + b_s)$$

3.2.2 pytorch-pretrained-BERT Classifier

The classifier model by Huggingface [2019] utilizes BERT in implementing and fine-tuning an instance of BertForSequenceClassification. For our binary classification task of sarcasm detection, we used the off-the-shelf example run_classifier.py with our own hyperparameters. Both the discussion forum data Oraby et al. [2016] and Reddit data Khodak et al. [2018] were pre-processed by our processing script to meet BERT's standard data format requirements and split data into train set, dev set and test set. Two separate bash scripts were written with tuned parameters for training and evaluation. To evaluate our BERT classifier performance, we further added precision, recall and F1 score calculations on top of the accuracy evaluation already provided in the classifier.

4 Experiments

We divided all data into three sets: train set, dev set and test set. We decided to use `NLLLoss` with `LogSoftmax()` on top-level instead of `CrossEntropyLoss()` for two reasons: (1) The log process in more explicit. It helps us debugging our code; (2) the Pytorch document for Softmax says `LogSoftmax()` with `NLLLoss` is faster and has better numerical properties.

4.1 Data

There are two datasets we used for this paper: Discussion Forum data [Oraby et al., 2016], and Reddit Sarcasm data [Khodak et al., 2018].

The Discussion Forum data has 4692 sarcastic and non-sarcastic posts in total. The annotation was done by crowd sourcing. Reddit data includes 1.3 million remarks of self-annotated Reddit corpus sarcasm data. Each entry in both datasets have the original context sentence, response sentence and a label indicating whether the response is sarcastic or not. The difference between these two datasets is, the label in Discussion Form data is tagged by crowd sourcing, while Reddit data is self-annotated by the author of the response. In this sense, sarcasm in discussion forum data is labeled from a reader's perspective, while the reddit data is more from the speaker's intention.

4.2 Evaluation Method

For quantitative evaluation, we consider Accuracy, Precision, Recall and F1 scores.

$$Accuracy = \frac{truePostive+trueNegative}{all}, Precision = \frac{truePostive}{truePositive+falsePositive}$$
$$Recall = \frac{truePostive}{truePositive+falseNegative}, F1 = 2 * \frac{Precision*Recall}{Precision+Recall}$$

Our primary metric will be Accuracy. We will also use Precision, Recall and F1 scores to understand whether our model is biased towards one label, and whether the selected train and test datasets are balanced. Moreover, since the SVM baseline explored by Ghosh et al. [2017] doesn't include Accuracy, we will use other metrics to perform comparisons.

4.3 Experiment Details

Since custom project doesn't have any scaffolding or data pre-processing, many of our early efforts were spent on setting up data and figuring out the high-level architecture of our models. We referenced the default project setup and used the GloVe pre-trained word embedding. Due to the fact that both datasets are forum data, the context and response usually contain multiple long sentences. In pre-processing, we used spaCy library to extract word tokens and sentences, and cut and padded each context and response to the size of [10 sentences * 50 words]. We implemented `torch.utils.data.DataSet`, as well as the convenient `data.DataLoader` for batch iterations. For our word embedding, we created our own `ModelEmbeddings` class, which achieves two unique goals: (1) To make it easier to load our pre-trained word vectors using `nn.Embedding.from_pretrained`; (2) To make it possible to average word embeddings for all words within the same sentence and to only return one averaged embedding per sentence.

We experimented different LSTM models with different combinations of metrics. The metrics below show the values we have experimented with:

- epochs = 10, 20, 30, 40, 50
- learning_rate = 0.01, 0.005, 0.001
- vocab_size = 24261
- embed_size = 300
- hidden_size = 128, 256
- output_size = 2 (binary classification)
- batch_size = 5, 10
- lost_fuction = `nn.CrossEntropyLoss`, `nn.NLLLoss` with `LogSoftmax`
- optimizer = `optim.Adam`, `optim.SGD`

For BERT, we utilized the existing BERT pytorch library and did some pre-processing on the inputs so that it can fit into the model. These are the BERT hyper-parameters we used:

- max_seq_length=128
- train_batch_size=16
- num_train_epochs=3.0
- eval_batch_size=8
- learning_rate=5e-5
- num_train_epochs=3

Training time for discussion forum data is fast because there are less than 5000 entries. For Reddit data, it takes about four hours to generate the indices for 100k entries. Since we notice that as dataset sizes increases over a threshold, the performance of LSTM models drops (refer to Parameters Tuning section), we set our training dataset size for LSTM models to be 100k. For training, LSTM models usually take around 30 minutes for a batch_size of 10 with 30 epochs for 100k entries. However, BERT takes more than 10 hours to train one epoch for 1.3 million entries.

4.4 Results

4.4.1 LSTM

As shown in the tables below, we report a series of evaluation metrics, including Accuracy(Acc) and Precision(P), Recall(R), F1 score (F1) on both Sarcastic(S) and Non-Sarcastic(NS) class. In total, we ran four different models: (1) `Vanila LSTM_r`: single LSTM based on only reply; (2) `LSTM_c+LSTM_r`: the combined model of context LSTM and reply LSTM stacked together with a feed-forward layer; (3) `Conditional_LSTM` (4) BERT. (Note: each model has its own best set of parameters. The numbers reported below are selected based on the best Accuracy result for each model.)

Table 2: LSTM results for the discussion forum dataset

Experiment	S			NS			Acc
	P	R	F1	P	R	F1	
SVM_r (Baseline)	65.55	66.67	66.10	66.10	64.96	65.52	n/a
SVM_{c+r} (Baseline)	63.22	61.97	62.63	62.77	64.10	63.50	n/a
$LSTM_r$	67.76	60.80	64.01	62.75	**69.54**	65.97	65.05
$LSTM_c$ + $LSTM_r$ with attention	51.44	**84.71**	64.01	43.51	12.83	19.82	51.22
$LSTM_{conditional}$	**71.32**	80.13	**75.47**	**75.82**	65.93	70.53	**73.23**
$LSTM_{conditional}$ [Ghosh et al., 2017]	70.03	76.92	73.32	74.41	67.10	**70.56**	n/a

(Accuracy not reported by Ghosh et al. [2017])

Table 3: LSTM results for Reddit dataset

Experiment	S			NS			Acc
	P	R	F1	P	R	F1	
$LSTM_r$	57.64	**80.95**	**67.33**	**68.28**	40.80	51.08	60.83
$LSTM_c$ + $LSTM_r$ with attention	50.56	65.18	56.95	47.07	32.70	38.59	49.38
$LSTM_{conditional}$	**67.86**	66.58	67.22	66.83	**68.11**	**67.47**	**67.34**

Table 2 shows the results of Discussion Forum Data for our SVM Baseline, LSTM models, and the LSTM_conditional model conducted by Ghosh et al. [2017]. LSTM_conditional and LSTM with word-level attention are the two best models according to Ghosh et al. [2017]'s previous research. We can see that our LSTM_conditional model is able to outperform the previous LSTM_conditional model on most metrics. When looking across different models we implemented, LSTM_conditional

also has the highest and most balanced performance, especially when comparing to our simple LSTM with response only model. This is expected as we believe adding context information to sarcasm detection model can help improve model accuracy. This is true for both Discussion Forum (small size) and Reddit Data (large size) (Table 3). The improvement is significant when we compare the results of LSTM_conditional with LSTM_r, the Accuracy improves 8% for Discussion Forum data, and 6.5% for Reddit data. Although LSTM_r has the best metrics for some categories for Reddit Data, we can see that the metrics across "S" class and "NS" classes are not balanced, meaning the result has some level of bias. Moreover, the overall accuracy is still lower than that of LSTM_conditional.

Our LSTM with attention didn't perform well as expected. We suspect a lot more tuning is required, and possibly we should try out other approaches to get sentence-level embeddings other than averaging all word embeddings in the sentence. In Ghosh et al. [2017]'s paper, the authors are not very specific about what attention they are using and how it is trained, we highly suspect that there are more nuances in their attention model other than what we represent here. On the other hand, it also likely that our implementation contains some errors.

4.4.2 BERT

Table 4 shows the results on both datasets for our BERT classifier with context-only, reply-only and concatenated context and reply. With similar architecture of the model, such results offer great insights of the nature of the data sets and how sarcasm detection works for them.

Table 4: BERT Results on both Discussion Forum and Reddit Data

Dataset	Experiment	S			NS			Acc
		P	R	F1	P	R	F1	
Discussion Forum	BERT_c	84.70	80.05	82.31	10.11	13.43	11.54	70.51
	BERT_r	68.60	84.14	75.58	44.94	25.15	32.25	64.10
	BERT_{r+c}	70.71	87.01	78.02	55.06	30.62	39.35	67.74
Reddit	BERT_c	41.40	62.82	49.91	71.74	51.18	59.74	55.27
	BERT_r	82.23	78.25	76.18	80.02	92.18	85.67	76.08
	BERT_{r+c}	64.79	60.93	62.80	74.99	77.96	76.45	71.18

(Note: BERT_{r+c} for Reddit is trained on 100k entries only due to time constraint while other models are trained on 1.3 million entries. We expect better performance with more data for BERT_{r+c}.)

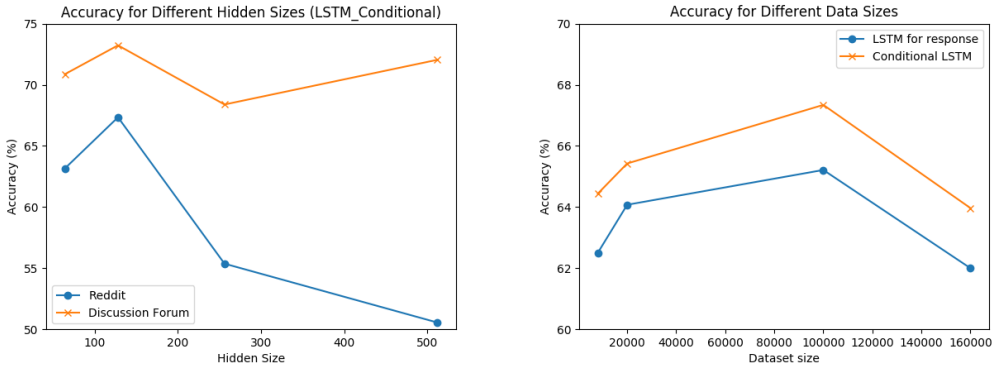
Discussion Forum: The discussion forum dataset performs the best with both context and response information. Although BERT_c has the highest Accuracy, the results are quite unbalanced between "S" and "NS" labels. This could be due to unbalanced training and test dataset. This result is consistent with the theory of Ghosh et al. [2017] that contextual information is essential to improve sarcasm detection. Consider the context format of discussion forum, the topic of discussion and background are of great important in understanding intentions of the response and detecting potential contrary emotions/opinions to flag sarcasm.

Reddit: The Reddit dataset performs the best with response-only information. This is a bit counterintuitive to us and different with the theory proposed by [Ghosh et al., 2017]. However, it's possible that the Reddit response-only dataset contains many flags for sarcasm, which are also present in the large dataset of other NLP tasks that BERT is pre-trained with. Compared to the LSTM response-only performance, which also did fairly well with just vanilla LSTM, it's also likely that the Reddit reply dataset contains various outstanding marker for sarcasm which make it easy to identify based on response alone. And of course, the most possible reason is because we have less training data for BERT_{r+c}.

4.4.3 Parameters Tuning

We also performed some hyper-parameter tuning for our LSTM models, and found useful insights from the tuning. In general, for all LSTM models, a combination of learning rate of 0.005, hidden size of 128, dropout rate of 0.2, Adam Optimizer, LogSoftmax and NLLLoss for loss function generates best Accuracy performances. Figure 3 shows how Accuracy changes for different hidden state sizes (Reddit and Discussion Forum Data) and data sizes (Reddit Data only). We found that extreme values for both cases will harm the model's performance. We are surprised to find that out even for dataset

sizes. We believe, in general, small hidden state size cannot provide enough information for the model, while large hidden state size may lead to overfitting problems. Similarly, Accuracy sharply increases at first when we increase the training data size, but when we reach a certain threshold (approximately 100k), the performance plateaus and starts to slightly drop. We believe this is also due to overfitting problem, as we were able to get back to high Accuracy at large dataset size with higher dropout rate (dropout rate = 0.4).



(a) Experiments with different hidden state sizes (b) Experiments with different data sizes (Reddit Data)

Figure 3: Accuracy for different experiments

5 Analysis

5.1 BERT

Overall, BERT has outperformed all the LSTM models by a large margin, for both the smaller discussion forum dataset and the Reddit dataset. For LSTM model, the performance is quite different for different datasets. For example, LSTM_conditional has an Accuracy of 73.23% for Discussion Forum, but a 67.32% for Reddit data. We suspect the main reason is due to the nature of these two data sources. Discussion Forum data has shorter sentences in each entry and all entries are labeled and selected by crowd-sourcing. The dataset contains generally proper English. However, for the self-annotated Reddit data, since it has never gone through professional human processing, the entries cover many different topics and internet slangs. For example, this response is self-annotated as sarcastic: `"NO WAIIIIIIIIIIII"`. It is hard for the Word Embedding model and LSTM model to pick up. On the other hand, this is not an issue for BERT.

We further compared different training data size and the performance of BERT and found that BERT is able to perform well even with a small set of data, and improving the size of the training dataset doesn't necessary lead to better performance.

Considering the design of BERT, these results actually make a lot of sense: as BERT is pertained on related NLP tasks and has much more data to learn from even before our training. Specifically, the BERT classifier we are using was pre-trained on BertForSequenceClassification, a fine-tuning model that include `BertModel` and sequence-level classifier on top of the `BertModel`. Our training data helps to fine-tune the model to a powerful single-sequence classifier, but too much data for tuning is unnecessary. It's far more beneficial when the NLP tasks the model is pre-trained on are related to our task. In our case, specifically, sarcasm detection can benefit from what BERT has learnt for the other tasks (e.g. general sentiment form sentiment analysis, general language knowledge from part-of-speech-tagging and parsing etc).

5.2 Areas of Improvements

We also identified several areas of future improvements. In this project, we averaged all the word embeddings in a sentence to calculate a sentence's embedding. This method has great advantage in its simplicity and also works fairly well in representing sentences' semantic meaning. One obvious drawback of this method is that: similar to the "bag of words" method, it ignores the ordering of words in a sentence. It also ignores the syntactic information embedded in sentence composition,

tones, etc. Alternatively, we could follow descriptions from Yang [2016] to implement a Hierarchical Attention Network, which entails adding a word-level attention to all words in the sentence in similar fashion as we add sentence-level attention.

5.3 Data and Error Analysis

Another thing we realized later in the process is that there are several other pre-processing techniques we can use to possibly improve the performance of our models. Due to the nature of these two datasets, the context and response texts often include word tokens that are not found in the GloVe pre-trained word embeddings. This is mainly because forums often introduce their own emojis or markdowns. Table 5 shows examples of context and response pair with incorrect predicted labels (Disclaimer: all examples don't represent our opinions). For instance, in the first example, the response contains word tokens such as "emoticonX-Good" and "emoticonX-Confused". We also found examples in Reddit data where certain words have markdowns like "**have**"" to make the word bold ("**have**"), or uses all uppercase letters. Usually, the usage of certain markdowns and emojis gives strong indication of sarcasm. However, in our pre-processing, these patterns are either being tokenized into different tokens or treated as words not found in the pre-trained embeddings. In the future, we could develop separate word embeddings for them, or maybe add prefix word embeddings to word tokens with certain pattern: $[0, 0, 0]$ for normal word, $[0, 0, 1]$ for all uppercase letters, and $[0, 1, 0]$ for emojis.

Table 5: Context and response examples with labels

Dataset	Context	Response	Golden Label	Predicted Label
Discussion Forum	Obviously you missed the point. So sorry the the irony was beyond you.	I guess we all missed your point Justine, whatever it might have been. emoticonXConfused Better luck next time. emoticonXGood	S	NS
Reddit	The Vanishing Of Ethan Carter is the reason why I play on PC	IT'S BETTER ON PS4	S	NS
Reddit	BLM is trying to shut down MSP International Airport	An hour ago I thought black lives *didn't* matter, but now that they've protested at the airport I've completely changed my mind and now think **black lives matter**	S	NS

6 Conclusion

In this paper, we experimented with different LSTMs and BERT model on sarcasm detection with both response-only and context-and-response Discussion Forum data Oraby et al. [2016] and analyzed their performances. The results show that for LSTM models, utilizing context information can actually improve model performance. Our LSTM_conditional model generates better results comparing to previous LSTM_conditional model developed in earlier papers. More importantly, we discovered that the pre-trained BERT classifier can achieve state-of-the-art results for sarcasm detection, even with small dataset. With larger dataset like Reddit data, it outperforms our best LSTM model by 7%.

For future work, we plan to improve our current model, either by adding bidirectional LSTMs or by introducing word-level attention. With better understanding of sarcasm detection with context and how sarcamsm triggered, we are also interested in exploring the possibility of sarcastic text response generation based on a certain context.

7 Additional Information

Mentor: Annie Hu

References

Weiwei Guo Debanjan Ghosh and Smaranda Muresan. 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*, 2015.

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

Debanjan Ghosh, Alexander Richard Fabbri, and Smaranda Muresan. The role of conversation context for sarcasm detection in online interactions. *SIGDial*, 2017. doi: 10.18653/v1/w17-5523.

Huggingface. The big--extending-repository-of-transformers: Pretrained pytorch models for google's bert, openai gpt gpt-2, google/cmu transformer-xl. 2019.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. A large self-annotated corpus for sarcasm. 2018.

Shereen Oraby, Vrindavan Harrison, Lena Reed, Ernesto Hernandez, Ellen Riloff, and Marilyn Walker. Creating and characterizing a diverse corpus of sarcasm in dialogue. *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2016. doi: 10.18653/v1/w16-3604.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

Diyi Yang Yang, Zichao. Hierarchical attention networks for document classification. *Human Language Technologies*, 2016.