A Transformer Approach to Contextual Sarcasm Detection in Twitter

Hunter Gregory, Steven Li, Pouya Mohammadi, Natalie Tarn, Rachel Draelos, Cynthia Rudin
Department of Computer Science, Duke University
{hlg16, sl561, pm201, nt94, rlb61}@duke.edu, cynthia@cs.duke.edu

Abstract

Understanding tone in Twitter posts will be increasingly important as more and more communication moves online. One of the most difficult, yet important tones to detect is sarcasm. In the past, LSTM and transformer architecture models have been used to tackle this problem. We attempt to expand upon this research, implementing LSTM, GRU, and transformer models, and exploring new methods to classify sarcasm in Twitter posts. Among these, the most successful were transformer models, most notably BERT. While we attempted a few other models described in this paper, our most successful model was an ensemble of transformer models including BERT, RoBERTa, XLNet, RoBERTa-large, and ALBERT. This research was performed in conjunction with the sarcasm detection shared task section in the Second Workshop on Figurative Language Processing, co-located with ACL 2020.

1 Introduction

Sarcasm detection is an important step towards complete natural language comprehension since a sarcastic phrase typically expresses a sentiment contradictory to its literal meaning. Humans usually detect sarcasm with contextual clues, especially intonation, which is not available in text-based social media data.

One challenge of sarcasm is the frequent necessity of prior knowledge. Consider an example:

> *Context*: driver tailgating a cyclist gets instant justice
> *Response*: maybe he tried to save gas like you do when you tailgate a transport truck

As humans, we can discern this response is sarcastic since we know a large car behind a small bike would not improve aerodynamics and hence gas mileage. This is a Herculean inference for an algorithm. In other situations, sarcastic inference relies on knowledge about science, sports, politics, or movies.

A broader challenge in classifying sarcasm is that a model trained on one dataset does not necessarily generalize to another one.

Using a Twitter dataset from the Second Workshop on Figurative Language Preprocessing, we tackle this difficult challenge of sarcasm detection and its specific issues as discussed above.

2 Datasets

The workshop provided two balanced sarcasm datasets from both Twitter and Reddit with 5,000 and 4,400 observations respectively. The workshop collected and labeled the Twitter data using the hashtags #sarcastic and #sarcasm, and preprocessed the data by replacing URLs and user mentions with placeholders. The Reddit dataset is a subset of that from Khodak et al. (2017). Both datasets consist of a sarcasm label, response, and conversation context. Both the test datasets contained 1800 observations.

Unfortunately, there were notable limitations in the Twitter data. We removed 300 observations that were duplicates (further discussed in Preprocessing). Nuances in twitter data such as acronyms, hashtags, and emojis needed to be processed. Furthermore, the Twitter dataset had missing hashtags, affecting the meaning of the response. Some tweets also contained images in the response, but there were no images in the data. For example, one tweet contained "they gave me the most (#beautiful, removed in the dataset) eggs for breakfast" with an image of chickens, but without the image it would be hard to determine its label.

There was also a considerable domain shift between training and test data for the workshop. For all models, training and validation scores were significantly higher than test scores.

3 Related Works and How We Use Them

A wide variety of models have been created for sarcasm detection. Our baseline model stems from the work of Ghosh et al. (2018) about sarcasm analysis using conversation context. This model involves one LSTM reading the context and another reading the response. While Ghosh and Veale (2017) proposed a similar architecture based on bidirectional LSTMs to detect sarcasm in Twitter, we found that bidirectional LSTMs performed similarly to unidirectional LSTMs. We used two different word embedding architectures, one from a previous paper trained on a separate Twitter data set and one using CBOW with position-weights, with character n-grams of length 5, a window of size 5 and 10 negatives (Ghosh et al., 2018; Grave et al., 2018). Sentence embeddings were obtained from averaging the word embeddings. We experimented with both word-level and sentence-level attention models, but we found that attention-based models performed similarly to those without (Yang et al., 2016).

The GRU architecture seemed to be a promising alternative option to the LSTM. Exploring tasks on audio, handwriting, and musical classification, Greff et al. (2016) discovered that GRUs performed the same as LSTMs. Their finding across multiple domains suggested that GRUs would perform similarly to LSTMs in the Twitter domain. Given the small size of our dataset and the reduced parameter size and complexity in GRUs, we believed this architecture could generalize better than LSTMs.

We also experimented with transformer models, which have been very successful for other applications such as sentiment analysis, question answering, and recently even for sarcasm detection (Devlin et al., 2019; Peters et al., 2018; Kayalvizhi et al., 2019; Potamias et al., 2019). We experimented with using pre-trained representations from BERT as well as RoBERTa, obtained from a bidirectional approach of a masked language model and ELMo which uses a concatenation of representations obtained from a left-to-right and a right-to-left language model (Peters et al., 2018; Liu et al., 2019). We applied ensemble learning using various pretrained transformer models (BERT, RoBERTa, XLNet, RoBERTa-large, ALBERT with no fine-tuning on the transformer weights), where each model has a learned weighted sum for the hidden layers that is concatenated with hand-made features, that is then fed into a dense layer for classification similar to the work of Wang et al. (2015).

4 Methodology

4.1 Preprocessing

For the Twitter dataset, we had to perform preliminary preprocessing. In the data, we removed several exact duplicates of observations and several almost-exact duplicates, where the context or responses would only have minor differences in punctuation (e.g. including or excluding a period) or a few characters. Text in this dataset also had an issue where a space was always placed on both sides of an apostrophe (e.g. "ol' Pete isn't happy" would become "ol ' Pete isn ' t happy"). To fix this, we created a list of English contractions and slang terms paired with their expanded form (e.g. "ol'" with "old" and "isn't" with "is not") and then cleaned all the text using the list. We also removed all occurrences of "<URL>" and "@USER" in the text. Lastly, we expanded hashtags using the python package ekphrasis (Baziotis et al., 2017).

For both this cleaned Twitter data and the Reddit data, we proceeded with the following preprocessing and feature creation. We lower-cased text, removed punctuation (including hashtag symbols), and expanded contractions and slang terms using our list mentioned above. For additional features, We created most of the extra features that Ghosh et al. (2018) used when they were feasible (e.g. did not require paying for a dataset). These included binary features encoding whether any word in the response was entirely capitalized and whether the response contained quotation marks. We also created a ternary feature for whether zero, one, or multiple exclamation points were used in the response. Additionally, we developed features that were duplicated for both the context and response. These included a binary feature indicating if any word in the text had unnecessary repeating letters as well as a ternary feature for the sentiment of the context or response (positive, negative, or neutral) using TextBlob (Loria, 2018). In addition, we had a binary feature for whether a positive sentiment emoji was in the text, and similar features for negative and neutral sentiment emojis based on a table from Kralj Novak et al. (2015).

We also created a political feature after noticing that a large number of the sarcastic tweets were political in nature. More specifically, from a sample of 200 tweets that we took, we saw that 67.7% of the political tweets were sarcastic while 31.8% of the non-political tweets were sarcastic. We created this feature by conducting a boolean search on the

Word	Train set	Extra Set
joy	6.5%	76.9%
wonderful	6.7%	69.7%
voted	67.6%	6.0%
love	30.4%	72.7%

Table 1: Sarcasm frequencies in different corpora

response and context for key words and phrases that we deemed to be political. These included the names of famous politicians and words or phrases associated with relevant political issues such as "elections", "capitalists", "planned parenthood", or "sanctuary city". In order to reduce our false positive rate for political classification, we were careful to exclude words that are often political but could easily be used in a non-political context such as "president" (e.g. the president of a basketball team) or "immigrant" (e.g. a person referencing their life story).

After running an XGBoost model with sentence-level response embeddings and our features, all seemed to have relatively equal feature importances. Therefore, we proceeded with including all of the features above in our final models.

We also thought certain words or certain topics other than politics might be good predictors of sarcasm. We calculated the frequency of sarcasm for all k-grams and discovered certain words (unigrams) appeared almost strictly in sarcastic or non-sarcastic responses in the Twitter training data. As a sanity check, we calculated the same word frequencies for an extra Twitter dataset with over 65,000 responses gathered in the same way as the training set (Ptáček et al., 2014). Many comparisons between frequencies, some displayed in Table 1, proved that the patterns in the training set were not representative of Twitter posts as a whole.

We also found that the LDA topic models with the best coherence were not predictive of sarcasm. Predicting a tweet based on its top three topics with a SVM or Logistic Regression Classifier yielded 53% training accuracy. Such low accuracy scores suggested these features would not be likely to generalize. Thus, we did not use topic models or frequency-based features in our final models.

4.2 Final Models and Performance

For all of our models, we focused on classifying the Twitter dataset and left classifying the Reddit dataset to future works.

For our final baseline model, we used Twitter embeddings (Ghosh et al., 2018), and we employed a bidirectional LSTM with sentence-level embeddings for context and a bidirectional LSTM with word-level embeddings for response. These LSTMs outputted sequences, which were modified by the dimension-reducing attention scheme proposed by Yang et al. (2016). Finally, these outputs were concatenated with our extra features and passed into a dense layer. Switching to unidirectional LSTMs, changing the number of LSTM units, or adding an extra dense layer yielded no improvement. Despite our hopes, replacing LSTMs with GRUs also did not impact training/validation performance.

As our two datasets have different distributions, mainly due to differences in text structure, formatting, and content, training by combining the two datasets together reduced our testing performance on the Twitter data. This was due to some features that are characteristic of the Reddit data, but may not generalize to the Twitter data. In order to fully leverage both of the given datasets for predicting sarcasm on the Twitter datasets, we utilized a weighted binary cross entropy loss, where we weighed each batch of Reddit data less than our Twitter data. By doing so, the model picked up on more universal sarcasm characteristics inherent to both datasets, while still being tailored primarily to the Twitter dataset. While we only tested this for our LSTM based models, this approach could be generalized to our other models.

We also utilized pre-trained transformer models to create sentence-level embeddings for classification and word-level embeddings to feed into our LSTM model. For our testing, we chose to not fine-tune the weights of the transformer models themselves, however future works may consider doing so. Instead, we use a feature-based approach as suggested in Devlin et al. (2019), where we extract features from the hidden states of the pre-trained transformer model for prediction.

Two different approaches were used to incorporate context into the transformer inputs. The first approach was to concatenate the full context and response sentences into a single input. A potential downside to this method is that the concatenated sentences exceeded the transformer's sequence length limit for certain observations, and we were forced to remove words from the front of the context until the string was compatible with the transformer. The second approach was to feed the full context and responses into the transformer sep-

arately and concatenate the outputs. The second approach has the benefit of allowing for sentences that are longer than the transformer's sequence length limit, allowing for more context to be utilized. We also create a baseline model trained solely on the responses. From our validation results (Table 3), we decided to use the second approach for our final submissions.

We obtained token representations for each of the hidden states in the transformer model, which were then averaged across the whole sentence to obtain a sentence representation for each of the hidden states. To determine the best way to combine these hidden representations into a single sentence embedding, we experimented with using a weighted sum of all layers, a weighted average of the last four layers, using only the last layer, summing all of the layers, and concatenating the last four layers. We obtained the best validation results from using a weighted sum of all hidden states (Table 2), and used this for our final model. From here, we concatenated our additional features to this embedding and pass it through two dense layers for prediction.

For our final submission, we applied this method to five separate pretrained transformer models and ensembled their results, which boosted our F1-macro score by a significant margin on the test set (0.733 for a single model to 0.756). The models we chose to ensemble were BERT-base-uncased, RoBERTa-base, XLNet-base-cased, RoBERTa-large, and ALBERT-base-v2 from the Huggingface transformers library (Wolf et al., 2019).

We also developed an LSTM on a weighted sum of the BERT hidden layer outputs. This model was similar to the baseline LSTM. BERT outputs from context were passed into one bidirectional LSTM, and BERT outputs from response were passed into another; however, these LSTMs outputted final states instead of sequences. These results were concatenated with the extra features and passed into two dense layers. On the test set, this model had an approximate 2.2% increase in F1-macro score compared to the other LSTM baseline.

Additionally, we implemented an ensemble learning approach in the hopes of highlighting the strengths of the many models that we developed. We were hopeful after seeing the improvement from the multiple transformers model that this model could achieve similar success. The assumption was that if we allowed models that were extremely confident in certain observations or performed best on observations with certain characteristics to classify on those observations while allowing other models to classify observations in which this was not the case. In order to implement this, we first trained and validated a multitude of models on only half of the training data given, returning classification probabilities on the other fifty percent of our training data and all of the test data. These models included the LSTM model and BERT model embeddings passed through an SVM, both discussed earlier, in addition to an LDA topic model, an XGBoost model, and a Gaussian Process Classifier, all of which seemed to perform decently on their own (besides LDA) and implement varying logic to the same classification problem. We chose a 50/50 split for our first training since we wanted to supply these initial models with enough training data to develop accurate classifications while leaving enough data for our ensemble classifier to train on. Once obtaining these outputs, we trained a Logistic Regression model on the prediction probabilities generated by the many models for the remaining 50 percent of the training set as well as the sarcasm features that we had developed. We chose Logistic Regression because it seemed to overfit less than polynomial kernel SVM or decision forest, and there has been research on the benefits of Logistic Regression on accuracy for ensemble learning (Wang et al., 2015). We then used this Logistic Regression to predict whether the test data was sarcastic or not. This model achieved a validation score of 0.779 and a test score of 0.686, indicating that the model did not generalize well. This ensemble model also worsened scores of our best models, which would achieve a higher F1-macro validation score when trained on the entirety of our training

	RoBERTa			BERT		
Method	Precision	Recall	F1	Precision	Recall	F1
Weighted sum of last four hidden layers	0.770	0.769	0.768	0.782	**0.776**	0.776
Sum all layers	**0.786**	0.767	0.771	0.767	0.750	0.751
Last layer only	0.761	0.760	0.760	0.772	0.767	0.768
Concatenate last four layers	0.773	0.772	**0.773**	0.769	0.751	0.751
Weighted sum of all layers	0.764	**0.774**	**0.773**	**0.785**	**0.776**	**0.778**

Table 2: Macro-averaged validation scores using various methods of combining the hidden states. Best scores for each transformer are in bold

	RoBERTa			BERT		
Method	Precision	Recall	F1	Precision	Recall	F1
Response only	0.757	0.754	0.755	0.752	0.754	0.752
Concatenated context and response	**0.791**	0.762	0.765	0.776	0.761	0.762
Separate context and response	0.764	**0.774**	**0.773**	**0.785**	**0.776**	**0.778**

Table 3: Macro-averaged validation scores for our transformer architecture with varying levels of contexts

dataset. However, the best transformer models and RNN models would drop from a validation score of above 0.703 and 0.682 to 0.667 and 0.504 respectively, when trained on half of the data as done in the ensemble model. Therefore, we believe that the ensemble model would have achieved better scores if it were trained on a larger data set than the one we worked with. However, it appears that the weighted transformers model above performed best given our dataset.

As seen in Table 5, all our final models outperformed the LSTM with attention (our implementation of the baseline from Ghosh et al., 2018). Transformer representations seemed to capture the most relevant information for sarcasm detection, and having context tends to improve results.

5 Error Analysis

These results, however, are far below the F1-macro scores that were achieved in validation, which led us to believe that our training data may have come from a different distribution than our test data. In order to investigate this further, we calculated distributions of our features for both the train and test set. We found a few differences between the distributions indicating that there may be a covariate shift between the training and test sets. We provide a few results from this analysis in Table 4. Notably, we see that there was a difference of .0816 in the percentage of tweets that were political between the test set and the training set. We also see far less observations with quotes or fully capitalized words in the test set than the training set. While this may be due to a small dataset and a high variance in the distribution of tweets, it still provides us with enough information to believe that there is a covariate shift between the training and test sets.

As mentioned above, we also saw a large difference between our validation and test scores. For example, we achieved a validation F1-macro score of 0.767 for our LSTM model with attention. However, this model achieved an F1-macro score of 0.669 on the test set. Additionally, our ensemble model went from a validation F1-macro score of 0.779 to a test F1-Macro score of 0.686. In conjunction with the covariate shift, we believe that our training and test set come from different distributions which would greatly increase our error.

Features	Train	Test	Difference
Contains a Capitalized Word	0.034	0.002	-0.032
Contains a Quote	0.078	.056	-0.022
Positive Emoji in Context	0.192	0.223	0.031
Political	**0.397**	**0.315**	**-0.082**

Table 4: Feature Distribution in Train and Test Sets

Final Models	Precision	Recall	F1-macro
Transformer Ensemble	**0.758**	**0.767**	**0.756**
Solo RoBERTa Transformer	0.733	0.734	0.733
BERT embeddings + LSTM	0.695	0.704	0.692
Ensemble model	0.687	0.689	0.686
LSTM w/ attention	0.669	0.669	0.669

Table 5: Test Scores of Final Models

6 Discussion and Conclusions

This research can hopefully guide future work on the topic of detecting sarcasm in social media. On that note, we would like to provide a few suggestions that others may find helpful in tackling the problem and a few of the findings that resulted in the largest improvements for our results.

First, utilizing multiple transformers and weighting them by performance seems to perform far better than a single transformer approach. Presumably, this allows the model to contain the information provided in each of the embeddings as opposed to a single form of embeddings. Next, we believe that this task requires a large amount of training data. We believe the reason the transformers performed so well was that they were pre-trained on large datasets. The models that we trained from scratch did not have as much training data, and we believe that they would have performed better with more training data. We believe that the weighted binary cross-entropy loss function to incorporate both datasets is a potential approach to help with this, and a future step would be to incorporate this into our final transformer models. Additionally, as emojis are an incredibly prevalent form of communication on social media, as a future step, we would like to incorporate emojis into the embedding space of our models. Finally, sarcasm detection is a difficult task. Even for humans, it is difficult to determine whether individuals are being sarcastic online. There are many complex variables that are difficult to quantify when determining if a short post is sarcastic. Therefore, capturing this notion of sarcasm within a model is difficult as well. We hope our techniques may be improved and expanded upon to solve other challenging natural language tasks.

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.

Aniruddha Ghosh and Tony Veale. 2017. Magnets for sarcasm: Making sarcasm detection timely, contextual and very personal. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 482–491.

Debanjan Ghosh, Alexander R. Fabbri, and Smaranda Muresan. 2018. Sarcasm analysis using conversation context. *Computational Linguistics*, 44(4):755–792.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2016. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.

S. Kayalvizhi, D Thenmozhi, B Senthil Kumar, and Chandrabose Aravindan. 2019. SSN_NLP@IDAT-FIRE-2019: Irony detection in arabic tweets using deep learning and features-based approaches. In *FIRE*.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PLoS ONE*, 10(12):e0144296.

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. *arXiv preprint arXiv:1907.11692*.

Steven Loria. 2018. textblob documentation. *Release 0.15*, 2.

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

Rolandos Alexandros Potamias, Georgios Siolas, and Andreas Georgios Stafylopatis. 2019. A transformer-based approach to irony and sarcasm detection.

Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. pages 213–223.

Hong Wang, Qingsong Xu, and Lifeng Zhou. 2015. Large unbalanced credit scoring using lasso-logistic regression ensemble. *PloS one*, 10(2).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.