Sarcasm Detection and Classification

Alice Drozd
UC Berkeley MIDS
W266 Natural Language Processing
adrozd62701@berkeley.edu

Abstract

Sarcasm detection is a critical yet difficult task in the Natural Language Processing community. In my project, I use the iSarcasm dataset to attempt two SemEval 2022 SubTasks: To determine whether or not a tweet is sarcastic, and to classify the ironic speech category a text belongs to: sarcasm, irony, satire, understatement, overstatement, and rhetorical question. Through this project I explored a variety of BERT-based models that leverage average pooling and CNN layers, leveraged techniques such as early stopping and learning rate scheduling, and used hyperparameter tuning to complete both tasks. The success my models achieved suggest that BERT-based models may be a good basis for modeling sarcasm detection and classification.

1 Introduction

Sarcasm detection is already a challenging task for humans, but for machines the difficulty is further heightened. During live human communication, sarcasm is often accompanied by facial expressions, gesturing, and context. Over online communication, though, sarcasm is much more difficult to detect, as the text often comes without context, gestures, or facial expressions. Machine Learning-driven classification of sarcasm could prove to be a useful tool for people who are not skilled at detecting sarcasm.

These two tasks were introduced as Task 6 of the 2022 SemEvals [5]. This task was comprised of 3 SubTasks, meant to be executed for both English and Arabic texts as follows.

- SubTask A: Given a text, determine whether it is sarcastic or non-sarcastic;
- SubTask B (English only): A binary multi-label classification task. Given a text, determine which ironic speech category it belongs to, if any;
- SubTask C: Given a sarcastic text and its non-sarcastic rephrase, i.e. two texts that convey the same meaning, determine which is the sarcastic one

This paper discusses models for the English versions of SubTasks A and B.

2 Background

Other papers have had success attempting these two subtasks through Hybrid Neural Networks, Recurrent Neural Networks like Long Short Term Memory Networks, and pre-trained Transformer-based models like BERT, RoBERTa, and CoLBERT [1].

I made to base my model on BERT, because I needed to balance having the ability to fine tune a great base model while working within the scope of my limited computational power [1,2]. Compared to LSTMs, which operate in a unidirectional manner, BERT offers greater complexity with its bidirectional sequence processing [7]. Although success was had with hybrid neural networks, their architecture can be complex, and may not necessarily outperform pre-trained models like BERT, RoBERTa, and CoLBERT. I also considered using RoBERTa or CoLBERT, which are more complex and robust but require greater computational power than a pre-trained BERT model [4]. However, I had to work within the scope of free Google Colab which has limited GPU, so I decided to work with a pre-trained BERT model through HuggingFace's **transformers** library.

3 Approach

3.1 Overview

This paper discusses my success and further potential experimentation in sarcasm detection and classification through various iterations of BERT models with CNN layers. Major experiments for SubTask A (binary classification) include a BERT-based model combined with CNN layers, which leverage the contextual understanding of BERT, along with the feature extraction capabilities of CNNs to detect sarcasm. Another successful experiment uses BERT with averaging layers. The final model for SubTask B (multi-class binary classification) is similar to SubTask A, and it leverages the sigmoid output layer to result in a multi-class classification model.

Both models take in input text sequences, process them with BERT to obtain contextual embeddings, and then apply additional layers to output class probabilities.

3.2 Data

The iSarcasm dataset is a publicly available dataset built for SemEval 2022 Task 6. The train and test datasets are split into different files. The train dataset consists of a column with the text of the tweet, a binary-valued column for whether or not the tweet was sarcastic, and 6 additional binary-valued columns for classifying ironic speech categories if the tweet was sarcastic: sarcasm, irony, satire, understatement, overstatement, and rhetorical question. The target variables were annotated by the authors of the tweets themselves, which makes the labels trustworthy and removed the need for third party annotators. The English training dataset included 3468 tweets, 862 of which were sarcastic and had associated ironic speech categories, so there was some class imbalance.

3.3 Data Processing

For both SubTasks, first the dataset was loaded from csv format into the Colab Python notebook. The training data was split into training and validation sets with a 80-20 split, and then all three sets (train, val, and test) were converted to TensorFlow constants to be compatible with TensorFlow. The data was tokenized through whitespace tokenization, then fed back into the initialized BERT tokenizer as strings, which returns TensorFlow tensors to be used in a BERT model. The tokenized data is comprised of token ids, token type ids to indicate segmentation of text, and attention masks to recognize which tokens should be paid attention to.

3.4 Baseline

The baseline model for SubTask A was random selection, meaning 50% of the time a sarcastic label would be assigned to the sarcastic class. This model resulted in a baseline Accuracy Score of 0.50, F1 Score of 0.19, Precision of 0.12, and Recall of 0.44.

The baseline model for SubTask B was random selection for each class, meaning 50% of the time the first category was true, 50% of the time the second category was true, etc. This model resulted in a macro F1 Score of 0.23, a macro Precision of 0.50, and macro Recall of 0.23.

3.5 Experiments

3.5.1 SubTask A

The first of many experiments I did was to test out a simple BERT Classifier model, with one hidden layer with ReLu activation, a dropout layer for regularization, and a classification layer with sigmoid activation. It was compiled with an Adam optimizer and binary cross-entropy loss. Although it gave results that were slightly better than baseline, the low complexity of the model made it an insufficient final choice. After seeing that the model gave reasonable first-attempt results with a BERT base, I added on top of the BERT an averaging layer, which leverages the contextual embeddings from BERT, along with averaging across the sequence, to perform binary classification. It provides an alternative approach, which I thought could capture more comprehensive information about the input text. This model was indeed a slight improvement on the original classifier. However, I then decided to pivot to adding CNN layers on top of BERT. The main inspiration for this was a CNN's ability to learn information about features that a simple averaging layer would not be able to learn. This last model also uses global max pooling layers to reduce the dimensionality of the CNN output.

For SubTask A, I performed hyperparameter tuning with the maximum sequence length, number of filters, kernel sizes, dropout, hidden layer size, and learning rate to achieve success in my models.

I also experimented throughout this project with early stopping, class weights, and reduced learning rate schedule. After implementing early stopping, I saw a more steady improvement in validation accuracy, and prevention of overfitting, as the model would stop training early when validation performance stopped improving, restoring the best model weights. Because I had a significant class imbalance in my dataset, I chose to implement class weights to give more importance to underrepresented class.

3.5.2 SubTask B

I had less experimentation with SubTask B than with SubTask A, primarily due to limitations in computational power, which I discuss later in this paper. I built my model off the BERT averaging model from SubTask A, but amended this model to result in a 6-class binary classifier, instead of a single-class binary classifier like in SubTask A.

One interesting experiment I conducted with this SubTask is to see how much of an improvement I could get with the removal of the class most often missed, which was the "understatement" class. After looking at the data, I saw that not only were the semantic differences minute, but also there was not enough data for the model to learn any meaningful embeddings, at a measly 5% positive class rate. I used only a slightly modified model to make predictions on only 5 classes, sarcasm, irony, satire, overstatement, and rhetorical question, instead of the SemEval Task's 6 categories.

I also experimented with the same early stopping, class weights, and reduced learning rate schedule as in SubTask A. Notably, the class weights addition to my model had the greatest effect for this SubTask, because the 6 layers had lots of variance in scarcity.

3.6 Final Model

3.6.1 SubTask A

The final model for SubTask A was a model that averages BERT embeddings across tokens. I then applied a dense hidden layer with ReLU activation, followed by batch normalization and dropout for regularization and prevention of overfitting. Finally, it is passed through a sigmoid activation and compiled with the same Adam optimizer and binary cross entropy loss. The hyperparameters I tuned were the number of training layers, which determines how many BERT layers to retrain. This model worked best at 5 re-training layers. I used 0.4 dropout, 0.005 l2 regularization, hidden layer size of 50, a reduced learning rate schedule, and early stopping to restore best weights.

3.6.2 SubTask B

The final model for SubTask B uses pre-trained BERT embeddings to encode the input text, then averages these embeddings, and applies a dense layer with dropout for classification. The model is compiled with Adam optimization and binary cross entropy loss for training.

Model	F1	Accuracy	Precision	Recall
Baseline	0.20	0.48	0.13	0.46
CLS	0.23	0.50	0.50	0.51
Averaging	**0.39**	**0.87**	**0.72**	**0.63**
CNN	0.20	0.71	0.51	0.51

Table 1: Results of Major Experiments for SubTask A

Model	F1 Macro	Precision	Recall
Baseline	0.21	0.22	0.48
All categories	0.34	0.28	0.60
5 Categories	**0.48**	**0.48**	**0.49**

Table 2: Results of Major Experiments for SubTask B

4 Evaluation

The SemEval task specified the metric used to evaluate each model's performance. For SubTask A, the metric was "F1-Sarcastic", which calculates the F1 score specifically for the positive binary class. For SubTask B, the metric was the F1 macro score across categories. I also chose to evaluate my models on their accuracy, precision, and recall for SubTask A, and macro precision and macro recall for SubTask B.

5 Results

I received varied quality of results from my experiments. The best-performing model for SubTask A was the BERT averaging model, with an F1-Sarcastic score of 0.39, accuracy 0.87, precision 0.72, and recall 0.63 (Table 1).

The best performing model for SubTask B was the model with 5 categories, where the Understatement ironic speech category was removed. This model achieved an F1-Macro score of 0.48, macro precision of 0.48, and macro recall of 0.49 (Table 2).

6 Discussion and Conclusion

For many of my experiments prior to the best models, I was receiving a 0 F1-Sarcastic Score, but high accuracy. I realized that the most likely explanation was the high class imbalance resulted in my model learning only when to classify negative cases well, but then it wouldn't classify any positive cases correctly. After learning this, I implemented class weights in most of my models so my model could learn to pay more attention to the smaller positive class in SubTask A, and the 5 classes that were not dominant for SubTask B: irony, satire, understatement, overstatement, and rhetorical question [7].

I did implement in some of my models a hyperparameter that allowed me to determine the number of BERT layers to re-train, and I found the optimal number of layers to be around 2-3. I believe this number to be this low because my dataset was not that large, with only around 1000 positive class datapoints. I think I would have had greater success with re-training more layers if my dataset was augmented.

After careful consideration, even though other papers had success with data augmentation and the iSarcasm dataset was not that large, I decided to not pursue data augmentation techniques in my model for several reasons [3,8]. Firstly, available datasets like the 2018 SemEval Task 3 Irony Dataset and the Multi-Modal Sarcasm Dataset used data labeling techniques that differed in concept from the iSarcasm dataset [6]. The Irony dataset used the hashtags like #*irony* and #*sarcasm* to assign the positive sarcasm class to tweets. Although this method is, in essense, the author labeling their tweet as sarcastic themselves, it notably lacked a reliable negative class to compare to. This method is also prone to online Twitter cultural nuances like authors adding a sarcasm hashtag to non-sarcastic tweets. As such, I decided that the labeling of this dataset could have altered the learning of my model too drastically. The Multi-Model Sarcasm Dataset is a collection of scenes from TV shows, where sarcastic phrases are surrounded by context, which the iSarcasm test dataset obviously lacks. For these reasons, I decided against using these two popular datasets to augment my data. In the future, I would like to experiment with augmenting my positive class data through mutation [4].

I believe that many of my models gave solid results, but some were noticeably over-fitted. I tried to reserve the test class for only final testing, but I ended up having to run through my Python notebook including evaluation on the test set several times for debugging. I tried to combat overfitting with techniques like dropout and early stopping, so I did end up getting precision and recall scores that were relatively promising for a non-overfit model. If I had been competing as part of the SemEval task, I likely would not have been able to get my models to perform as well, because of the blind nature of the competition-environment test set.

One curious result I found for SubTask A is that the averaging BERT model outperformed the CNN BERT model. I would have expected the CNN to outperform the BERT-averaging model. However, I believe the CNN model was more prone to overfitting than the averaging model, and also CNNs are not as well equipped to handle sequential data. On the other hand, the averaging BERT model was probably great for combatting overfitting this relatively small dataset, due to its simpler nature.

Overall, my best model for SubTask A would have placed 13th at the SemEval task [9]. It seems that my all-category SubTask B model would have placed 1st at the SemEval task, so I am led to believe that there must be gross overfitting going on with this model somewhere.

If I had more time and computational resources, I would have liked to try a number of different models and techniques, such as RoBERTa or CoLBERT embeddings. I also would have liked to try incorporating other popular sarcasm and irony datasets into my modeling.

Limitations

One notable limitation of this project was that the dataset was limited. With approximately 3500 datapoints, fine-tuning a BERT-based model was doable and proved successful, even. However, had the dataset had a number closer to 10,000, I would have been able to get much better results. In fact, this issue was most notable in SubTask B, mostly because of the huge class imbalance.

Among the difficulties of this project was notably a huge limitation in computational resources. I built, tested, and evaluated all my models on Free Google Colab, which has a GPU limit that resets several times a week after an undisclosed amount of computational resources are used. This resulted in a lot of frustration for myself, as I was able to run an extremely limited amount of true experiments a week, spread across several of my personal google accounts. Despite this limitation, I tried to optimize my computational resources. After implementing Early Stopping for some of my models, I was able to run more models as the computational resources were not being wasted.

Acknowledgements

I would like to acknowledge the MIDS W266: NLP curriculum for inspiration for my modeling decisions. I used the code from some of the various

assigments as a starting point, but I have altered the original code significantly to better represent the problem being solved in this SemEval Task, and bring in my own unique ideas.

Bibliography

[1] Mahdaouy, A. El et al. (2022). *CS-UM6P at SemEval-2022 Task 6: Transformer-based Models for Intended Sarcasm Detection in English and Arabic* (arXiv:2206.08415). http://arxiv.org/abs/2206.08415

[2] Hindy, A. et al. (n.d.) *Classifying and Automatically Neutralizing Hate Speech with Deep Learning Ensembles and Dataset Ensembles.*

[3] Nigam, S. K. and Shaheen, M. (2022). *Plumeria at SemEval-2022 Task 6: Robust Approaches for Sarcasm Detection for English and Arabic Using Transformers and Data Augmentation* (arXiv:2203.04111). http://arxiv.org/abs/2203.04111

[4] Abaskohi, A., Rasouli, A., Zeraati, T., Bahrak, B. (2022). *UTNLP at SemEval-2022 Task 6: A Comparative Analysis of Sarcasm Detection Using Generative-based and Mutation-based Data Augmentation. Proceedings of the 16th International Workshop on Semantic Evaluation* (SemEval-2022), 962–969. https://doi.org/10.18653/v1/2022.semeval-1.135

[5] Abu Farha, I., Oprea, S. V., Wilson, S., Magdy, W. (2022). *SemEval-2022 Task 6: iSarcasmEval, Intended Sarcasm Detection in English and Arabic.* Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022) (pp. 802–814). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.semeval-1.111

[6] Misra, R., Arora, P. (2018). *Sarcasm Detection using Hybrid Neural Network.* https://doi.org/10.13140/RG.2.2.32427.39204 SemEval 2022—Results. (n.d.). Retrieved August 4, 2024, from https://sites.google.com/view/semeval2022-isarcasmeval/results

[7] V, R., Sivanaiah, R., S, A., Rajendram, S. M., T T, M. (2022). *TechSSN at SemEval-2022 Task 6: Intended Sarcasm Detection using Transformer Models.*

[8] Yuan, M., Mengyuan, Z., Jiang, L., Mo, Y., Shi, X. (2022). *stce at SemEval-2022 Task 6: Sarcasm Detection in English Tweets.* https://doi.org/10.18653/v1/2022.semeval-1.113

[9] SemEval Results. 2022. https://sites.google.com/view/semeval2022-isarcasmeval/results