Team GFZ Final Report

Bei Zhao(Captain) beizhao3@illinois.edu
Ryan Fraser rfraser3@illinois.edu
Yiming Gu yimingg7@illinois.edu

Table of Contents:

Explain what problem we are trying to solve

This code attempts to detect sarcasm in tweets.

This is a particularly tricky problem in Natural Language Processing (NLP) due to the nature of sarcasm itself. Sarcasm is highly dependent on the author, the specific topic in question, cultural attitudes about the topic, current events, tone, and much more. The difficulty is further compounded by the fact that sarcasm is (usually) intentionally designed to not sound sarcastic. In a simple example, a person asking his friend if an idea is good may hear that his idea is “a really good idea,” when in fact the friend means the opposite.

In particular, this code was built for the Text Classification competition for CS 410 at the University of Illinois at Urbana-Champaign. In this competition, 2 sets of tweets are provided. The training set is pre-labeled and identifies if example tweets are sarcastic or not. The test set is not labeled. Both sets also contain the context of the tweets, which in this context means the parent tweets that the tweets in question are responding to. The goals of the competition are twofold. First, teams must outperform the baseline, which means realizing an F1 score of about 0.723 (based on current results on LiveDataLab). Second, teams must try to rank highest out of all other teams.

Explain our model

We achieved our best results for this task by using a pre-trained deep learning model called RoBERTa. This model builds off of another well-known deep learning model called BERT, which stands for “Bidirectional Encoder Representations from Transformers.”

A few key differences between BERT and RoBERTa are outlined below:

	BERT	RoBERTa
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340
Training Time	**Base**: 8 x V100 x 12 days* **Large:** 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**

As one can see, RoBERTa takes longer to train because it has much more data. It also does not use Next Sentence Prediction, which is explained further below.

At its core, BERT uses Transformers. These are a deep learning construct that “learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms - an encoder that reads the text input and a decoder that produces a prediction for the task.”

A key feature of BERT is that it is bidirectional. This means it reads all of the words at once and uses all context available. Prior models used left, right, left-right, right-left context or some combination of those. This means that BERT obtains a more holistic understanding of context versus prior methods, usually leading to better results.

In very general terms, BERT uses 2 strategies to train on the data. First is Masked LM and second is Next Sentence Prediction.

In Masked LM, BERT hides a number of words in each sequence before training and then attempts to predict the hidden words. The loss function only considers the prediction of those hidden values. Because of this, the model is a bit slower to converge than prior models that did not make use of the full context.

In Next Sentence Prediction, the BERT model also “receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence.” Note that RoBERTa does not use Next Sentence Prediction.

Both strategies are used when training the model and the goal is to balance and minimize the loss of the combined strategies. We believe that RoBERTa outperforms BERT for sarcasm detection in tweets because tweets are very short and tend to have less overall context. Therefore, it makes sense that a scoring mechanism that does not focus as much on Next Sentence Prediction will be more fit for purpose for this competition.

Explain how we perform the training

Our very general outline of steps taken is below. To keep things concise, only high level descriptions are given. Our code is somewhat short and more detailed documentation is available in the code comments.

Basic Steps:

- Convert JSON data to CSV format while cleaning the labels
- Create k-fold cross-validation training sets
- Create a list of dictionaries for our training and test data
- Create CSVs with the context data, the response data, and the labels
- Convert CSVs into pandas dataframes
- Split data into respective training/test sets
- Select model and import into the environment
- Encode the text data using the Tokenizer library
- Convert data into Tensors using PyTorch
- Set up model parameters
    - Batch Size
    - Wrap Tensors
    - Create sampler for sampling data during training
    - Create dataLoader for training set
    - Freeze parameters
- Create model class and initialization logic
- Initialize model
- Import model optimizer from HuggingFace Transformers library
- Compute starting class weights
- Convert class weights to Tensors
- Define number of training epochs
- Define model training function
- Define model evaluation function
- Run and train the model

Explain any other methods tried and hyperparameter tuning

The team spent time on a number of different tasks outside of just building the model:

1. We spent a few weeks researching the competition and educating ourselves on deep learning models. When we first started, we believed we'd use word2vec and/or sentence2vec, two approaches for NLP. However, as we did more research, we found that there were better models we could use.

2. We spent time trying various deep learning models to see how each performed. We used a library from HuggingFace that allowed us to easily swap out pretrained NLP deep learning models. Below we've outlined the performance of various pretrained model/hyperparameter combinations, which is how we ultimately decided to use RoBERTa.

3. We also spent time trying to identify which data to train the model on. The data includes both target tweets and the context surrounding them, therefore, there were 3 intuitive combinations we felt were worth trying: response-only, context-only, and a combination of both. We found that the combination of both works best. This made sense to us given the very nature of sarcasm.

   In fact, sarcasm is almost always an extremely context dependent response to a topic that is oftentimes serious, or not inherently sarcastic. By its nature, sarcasm taken out of context sounds like any other normal statement. (Just think of the phrase, "good work," which can take on many different meanings depending on a huge number of factors. Most of the time, this should be taken as a friendly, encouraging, and positive statement. However, depending on the context, it could also be a sarcastic statement.)

Results from testing various pre-trained model and hyperparameter combinations

Following are the different testing results we've tried so far (We only include the best scores of different setup models). Passing the baseline F1 scores are highlighted in green color.

Model	precision	recall	f1	Model Description
ALBERT BS=25	0.65294	0.74000	0.69375	ALBERT model with Batch Size 25
ALBERT BS=25 MSL= 300	0.64178	0.70667	0.67266	ALBERT model with Batch Size 25 max_seq_len=300
ALBERT BS=26	0.64400	0.75778	0.69627	ALBERT model with Batch Size 26
ALBERT BS=27	0.63944	0.75667	0.69313	ALBERT model with Batch Size 27
ALBERT BS=28	0.63520	0.77000	0.69613	ALBERT model with Batch Size 28
ALBERT BS=29	0.62868	0.76000	0.68813	ALBERT model with Batch Size 29
ALBERT BS=30	0.62960	0.79889	0.70421	ALBERT model with Batch Size 30
ALBERT BS=31	0.65814	0.77222	0.71063	ALBERT model with Batch Size 31
ALBERT BS=32	0.65145	0.69778	0.67382	ALBERT model with Batch Size 32
ALBERT BS=33	0.64371	0.77889	0.70488	ALBERT model with Batch Size 33
ALBERT BS=34	0.61558	0.82556	0.70527	ALBERT model with Batch Size 34
ALBERT BS=35	0.65490	0.74222	0.69583	ALBERT model with Batch Size 35
ALBERT V2 BS=31	0.64377	0.56222	0.60024	ALBERT model v2 with Batch Size 35
BERT BS=25	0.68477	0.71444	0.69929	BERT model with Batch Size 25
BERT BS=26 RS&UP	0.63957	0.72556	0.67985	BERT model with Batch Size 26 remove stopword and unnecessary punctuations
BERT BS=27	0.65652	0.76667	0.70733	BERT model with Batch Size 27
BERT BS=27 RS&UP	0.64245	0.75667	0.6949	BERT model with Batch Size 27 remove stopword and unnecessary punctuations
BERT BS=28	0.62681	0.86778	0.72787	BERT model with Batch Size 28
BERT BS=28 R_context	0.66042	0.70444	0.68172	BERT model with Batch Size 28 and Reverse Context
BERT BS=28 RS&UP	0.64717	0.76222	0.7	BERT model with Batch Size 28 remove stopword and unnecessary punctuations
BERT BS=29	0.6519	0.80111	0.71884	BERT model with Batch Size 29
BERT BS=29 R_context	0.67768	0.57000	0.61919	BERT model with Batch Size 29 and Reverse Context
BERT BS=29 RS&UP	0.64358	0.70222	0.67163	BERT model with Batch Size 29 remove stopword and unnecessary punctuations

BERT BS=30	0.68614	0.66556	0.67569	BERT model with Batch Size 30
RoBERTa BS=28	0.66057	0.80222	0.72453	RoBERTa model with Batch Size 28
RoBERTa BS=29	0.64521	0.86888	0.74053	RoBERTa model with Batch Size 29
RoBERTa BS=29 R_context	0.67836	0.77333	0.72274	RoBERTa model with Batch Size 29 and Reverse Context
RoBERTa BS=30	0.66324	0.78778	0.72016	RoBERTa model with Batch Size 30
RoBERTa BS=31	0.63974	0.89777	0.74711	RoBERTa-large model with Batch Size 31
RoBERTa-large BS=27	0.64571	0.62778	0.63662	RoBERTa-large model with Batch Size 27
RoBERTa-large BS=28	0.54708	0.98778	0.70416	RoBERTa-large model with Batch Size 28
RoBERTa-large BS=29	0.71308	0.37556	0.49199	RoBERTa-large model with Batch Size 29
SqueezeBERT BS=28	0.6506	0.72	0.68354	SqueezeBERT model with Batch Size 28
SqueezeBERT BS=29	0.6537	0.74667	0.6971	SqueezeBERT model with Batch Size 29
SqueezeBERT BS=30	0.65345	0.76889	0.70648	SqueezeBERT model with Batch Size 30
SqueezeBERT BS=32	0.63973	0.84444	0.72797	SqueezeBERT model with Batch Size 32
SqueezeBERT BS=33	0.61348	0.91	0.73289	SqueezeBERT model with Batch Size 33
SqueezeBERT BS=34	0.63851	0.84	0.72553	SqueezeBERT model with Batch Size 33
XML-RoBERTa BS=27	0.70183	0.34	0.45808	XML-RoBERTa model with Batch Size 27
XML-RoBERTa BS=28	0.6059	0.91222	0.72816	XML-RoBERTa model with Batch Size 28
XML-RoBERTa BS=29	0.61864	0.89222	0.73066	XML-RoBERTa model with Batch Size 29
XML-RoBERTa BS=29 MSL=150	0.64249	0.82667	0.72303	XML-RoBERTa model with Batch Size 29, max_seq_len = 150
XML-RoBERTa BS=30	0.60588	0.91556	0.72920	XML-RoBERTa model with Batch Size 30

Team Member Contributions

- **Bei Zhao (Team Captain):** Bei served as our team captain for this project. Bei helped on all facets of the project. She wrote our initial proposal and coordinated all meeting times. Bei also spent a lot of time working to try various model/hyperparameter combinations to get the best results.
- **Yiming Gu:** Yiming was instrumental in getting our model up and running. He was quickly and efficiently able to get us a very solid foundation to work off of, such that we could spend time focusing on the best model and hyperparameters. Yiming also spent a lot of time trying various model and hyperparameter combinations to find the best results and wrote the README for the project.
- **Ryan Fraser:** Ryan helped the team strategize about how to approach the competition. He spent time trying to get the model to run on a GPU for faster training times, but ultimately the team decided to use CPUs. Ryan also wrote the project progress report, the documentation, and the final report.

References

Amardeep Kumar and Vivek Anand 2020. Transformers on Sarcasm Detection with Context. *Proceedings of the Second Workshop on Figurative Language Processing*

Debanjan Ghosh, Avijit Vajpayee, Smaranda Muresan, and Educational Testing Service 2Data Science Institute, Columbia University {dghosh, avajpayee}@ets.org smara@columbia.edu 2020. A Report on the 2020 Sarcasm Detection Shared Task. *arXiv preprint arXiv:2005.05814.*

GLUE Benchmark Leaderboard. (n.d.). Retrieved November 11, 2020, from https://gluebenchmark.com/leaderboard

Hankyol Lee, Youngjae Yu, and Gunhee Kim. Augmenting Data for Sarcasm Detection with Unlabeled Conversation Context. *arXiv preprint arXiv:2006.06259.*

Ingham, F. (2018, November 27). Dissecting BERT Part 2: BERT Specifics. Retrieved November 11, 2020, from https://medium.com/dissecting-bert/dissecting-bert-part2-335ff2ed9c73

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, and Google AI Language {jacobdevlin,mingweichang,kentonl,kristout}@google.com 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805.*

Nikhil Jaiswal 2020. Neural Sarcasm Detection using Conversation Context. *Proceedings of the Second Workshop on Figurative Language Processing*

Tanvi Dadu and Kartikey Pant 2020. Sarcasm Detection using Context Separators in Online Discourse. *arXiv preprint arXiv:2006.00850.*

Xiangjue Dong, Changmao Li, and Jinho D. Choi 2020.Transformer-based Context-aware Sarcasm Detection in Conversation Threads from Social Media. *arXiv preprint arXiv:2005.11424.*

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *Conference paper at ICLR 2020*

Suleiman Khan, Ph.D. “BERT, RoBERTa, DistilBERT, XLNet - Which One to Use?” *Medium*, Towards Data Science, 17 Oct. 2019, towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8.