Documentation for Text Classification Project

I. Overview

This program implements a model for detecting sarcasm in text. The training data consists of twitter feeds having context text, where the response to the context is labelled as sarcasm or not sarcasm. The objective is to define a model based on this training data that detects sarcasm in text, and use it to classify the responses in the test data as sarcasm or not sarcasm.

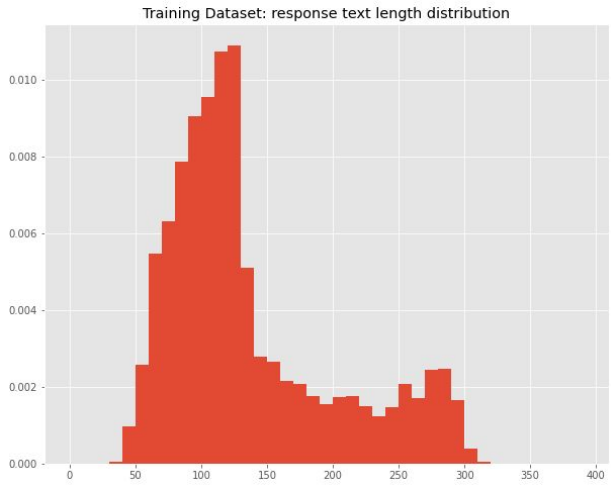
II. Data Profiling

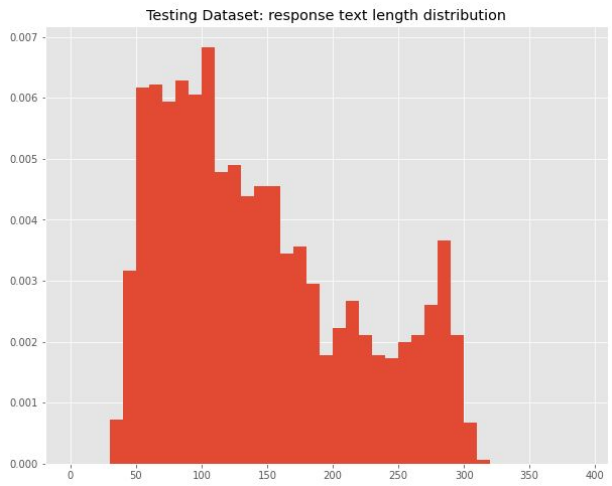
We conducted data profiling on both the training data and test data. The datasets provided are in json format. The training data contains the following columns:

- **Label**: Indicates whether the response is sarcasm or not sarcasm.
- **Response**: A string which contains the Tweet response to be classified.
- **Context**: A list which contains the conversation history in context of response. The order of the list is the same as the order of the dialogue (i.e. response directly replies to the last element of the list)

The training dataset contains 5000 Tweets in total and 2500 are labeled as sarcasm and the rest are labeled not. Therefore, the training dataset is considered as a balanced dataset and no oversampling is required.

We also conducted analysis on the response text length on both the training set and the testing set. The distribution shows that the training set and the testing set are similar in terms of response length distribution, and therefore, models based on the training set might be suitable for predicting the testing set.

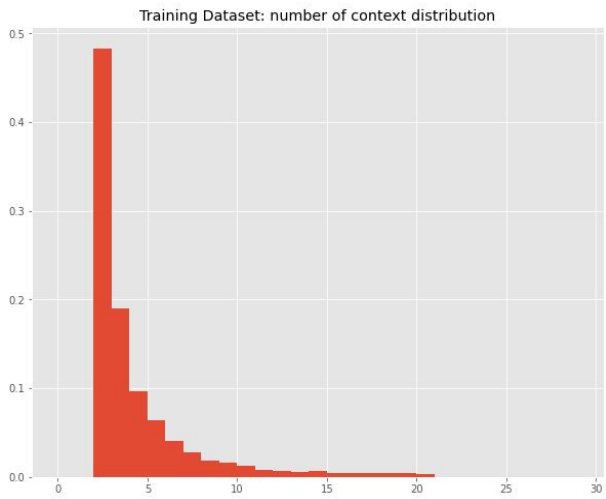


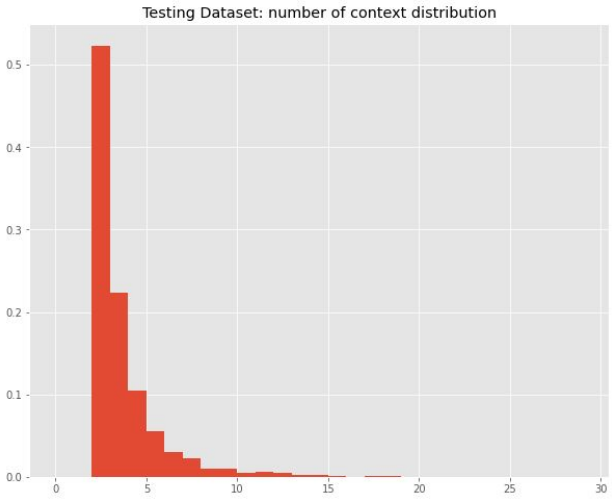


Percentile	Characters(Train)	Characters(Test)
0.1	71.0	59.0
0.2	85.8	76.0
0.3	97.0	92.0
0.4	108.0	108.0
0.5	117.0	127.0
0.6	126.0	149.4
0.7	145.0	176.0
0.8	190.0	214.0
0.9	254.0	265.0

In addition to response length, we also conducted an analysis on the number of contexts. The distribution shows that for both the training set and the testing set, the majority of the responses have two corresponding contexts. Based on our calculation, the average number for contexts for the training set is 3.87 and for the testing set is 3.16.

During our data preprocessing step, we figured that keeping all contexts will draw a warning message which indicates that “Token indices sequence length is longer than the specified maximum sequence length for this model”. Therefore, we decided to use the last three available contexts.





III. Data Cleaning

The following approaches were tried to clean up the data to see if it improved computation efficiency and\or classification accuracy:

1. Convert all text to lowercase
2. Remove all punctuations (except apostrophe)
3. Remove stop words
4. Stemming of words

A separate script **clean.py** was written to process the input data and write the cleaned up data in the jsonl format. However, we found that the cleaning of the data actually resulted in a loss of accuracy as compared to the original data. This could be because the cleanup process affected the sarcasm detection learning model. Therefore the decision to clean the data must be carefully weighed to see if it adversely impacts the learning model, and experimentation with\without cleaning is essential.

IV. Model Architecture

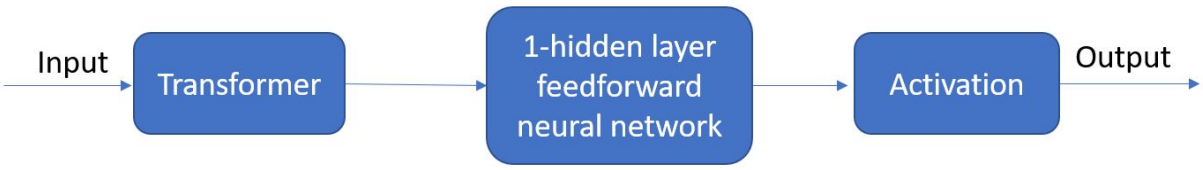
There are multiple ways to build contextual sentiment analysis models. Our final decision is to use the simple method: combine contexts with the response to form a single string as input to the model(Amardeep Kumar, 2020).

Throughout the process, we have considered the following models:

1. Fasttext
2. Roberta + 1-hidden layer neural network + ReLU
3. Roberta + 1-hidden layer neural network + Softmax

4. XLMRoberta + 1-hidden layer neural network + ReLU
5. Albert + 1-hidden layer neural network + ReLU

We also tried different learning rates for these models. The result shows that model #2 combined with a learning rate of 1e-5 gives us the best performance on the testing set with a f1 score of 0.787. The model architecture is as below:



V. Implementation

The program uses Google Colab + Jupyter notebook to take advantage of the Google GPU for accelerated data processing. The training and test data is stored on Google Drive.

The PyTorch library is used, which enables the usage of GPU as well as the use of various implementations of state-of-the-art NLP transformer libraries i.e. Pre-trained Language Models (PLMs).Transformers allow for parallelization, which makes it possible to efficiently train very big models on large amounts of data on GPU clusters.Transformer-based PLMs use much deeper network architectures, and are pre-trained on much larger amounts of text corpora to learn contextual text representations by predicting words based on their context.

One of the most popular transformers is BERT, developed by Google. RoBERTa (developed by Facebook) is an extension that is more robust than BERT, and is trained using much more training data and dynamic masking.

To investigate a lighter model, we researched and tested out Albert, which is essentially a lite-BERT. The benefits for training are evident; it provides two parameter reduction techniques to improve memory usage and BERT speed. According to Hugging Face, these techniques are forming two smaller matrices from an embedding matrix and providing repeated layers split on groups. The experience with this consisted of beating the baseline, but not being one of our best models.

The RoBERTa PyTorch transformer library is selected in our implementation as our top model.

The data is loaded using a batch size of 16. The model is trained on the training data set using epoch count of 12. The resulting model is serialized and saved in the PyTorch model format.

The test data is then evaluated against the model to classify the tweets as sarcasm\not sarcasm (stored in the generated answers.txt file).

VI. Instructions to Run Code

1. In Colab Notebooks under Google Drive, create a directory called *TextClassification*.
2. Within the *TextClassification* directory, create another directory called *data.*
3. Include the data files (**test.jsonl** and **train.jsonl**) in this *data* directory.
4. From the GitHub repository (https://github.com/samphadnis/TeamTextDragons), open **roberta_no_pretrain.ipynb** from the *code* directory into Colab Notebooks.
5. Since the saved model is too large to be uploaded to Github, we created a link to google drive and shared it on Github readme page. Download the model and save it under the *TextClassification* directory.
6. If you want to run the whole model, including the training portion, in the navbar, navigate to “Runtime” and select “Run all”. The pipeline should run, and the results **answer.txt** should be generated under Colab Notebooks.
7. If you only want to generate the results with the saved model, you may skip the **Model Training** portion of the code.

VII. Conclusion

The program is able to beat the baseline. The state-of-the-art Pre-trained Language Models such as RoBERTa are powerful tools for text classification.

VII. Team Contributions

Sameer Phadnis (phadnis3@illinois.edu): Team Leader
- Worked on and tested variations of data cleaning processes and documentation

Chen Yuan (cheny9@illinois.edu)
- Led the development work by setting up project infrastructure, implementing data profiling, classification pipeline, presentation and documentation

Abhishek Shinde (ashinde2@illinois.edu)
- Worked on training and testing various models, and improving the classification pipeline and documentation

VIII. References

- Kozlov, Alexander. “Fine-Tuning BERT and RoBERTa for High Accuracy Text Classification in PyTorch.” *Medium*, Towards Data Science, 7 Sept. 2020, towardsdatascience.com/fine-tuning-bert-and-roberta-for-high-accuracy-text-classification-in-pytorch-c9e63cf64646.
    - The code repository we relied on can be found here: https://github.com/aramakus/ML-and-Data-Analysis/blob/master/RoBERTa%20for%20text%20classification.ipynb
- “Transformers.” *Transformers - Transformers 4.0.0 Documentation*, huggingface.co/transformers/index.html.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov: “RoBERTa: A Robustly Optimized BERT Pretraining Approach”, 2019; http://arxiv.org/abs/1907.11692 arXiv:1907.11692
- Amardeep Kumar, Vivek Anand: “Transformers on Sarcasm Detection with Context”, 2020; https://www.aclweb.org/anthology/2020.figlang-1.13.pdf