

Charformer Performance on Categorizing Toxic Comments

Hannah Helgesen
hannahhelgesen@my.unt.edu

Michael Laufer
michaellaufer@my.unt.edu

Bryan Adams
bryanadams@my.unt.edu

Sarah Wainwright
sarahwainwright@my.unt.edu

Github: https://github.com/SarahJWainwright/Group6_CSCE5290

Introduction

Toxic comments affect the mental well-being of many internet users. Currently, there exist some tools to filter toxic comments online, but their accuracy has room for improvement, and few can differentiate multiple levels of toxicity. We seek to train a model with Charformer that can detect the level of toxicity in comments with high accuracy. This may help those constricted by negative comments to feel more comfortable browsing and sharing on the internet. We are using training data from Kaggle's "Toxic Comment Classification Challenge", with the goal of out-performing the standard Bert model, which already does an adequate toxic classification job.

Charformer, the tool we are using to categorize comments, creates a token space of the input data in the form of n-gram sub words. Some of these sub words do not make sense, but many of them have meaning. The output data is also represented using these n-grams which form unique embeddings. The authors of Charformer claim this representation yields similar accuracy to other transformer models while being end to end, meaning you can run the model on raw text rather than tokenizing first.

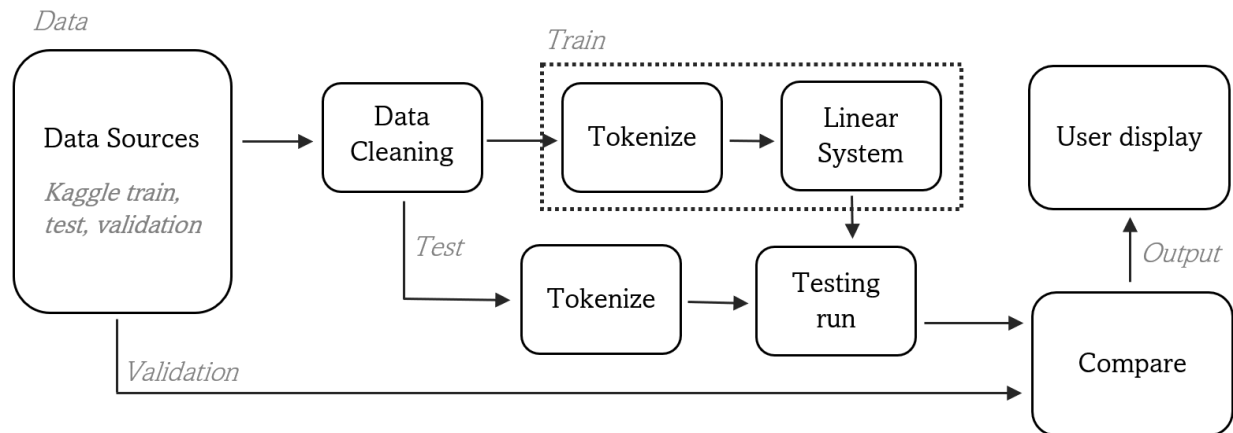
Related Work

Multiple submissions to the initial Toxic Comment Contest may be found at Kaggle's [Toxic Comment Classification Challenge](#). They strive to achieve a similar goal as our project.

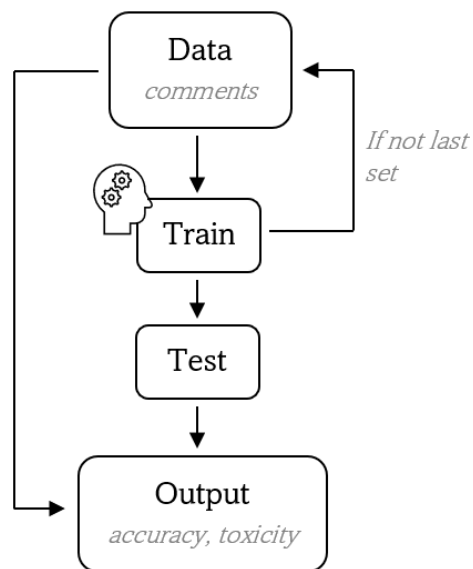
In addition, we used the following sources as reference while constructing our model:

- 1) "Yunje", and "smth". "Giving Multiple Parameters in Optimizer." *PyTorch Forums*, 4 Mar. 2017, <https://discuss.pytorch.org/t/giving-multiple-parameters-in-optimizer/869>.
- 2) Kudo, Taku, and John Richardson. "Sentencepiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing." *ArXiv.org*, Cornell UNiversity, 19 Aug. 2018, <https://arxiv.org/abs/1808.06226>.
- 3) Blob. "Sentencepiece Python Module Example." *Google Colab*, Google, https://colab.research.google.com/github/google/sentencepiece/blob/master/python/sentencepiece_python_module_example.ipynb.

Our Model



The general overview of our system was rather simple. We gathered data from a variety of sources and cleaned and separated our test and train data, which further tokenized each to fit the linear system. Following testing of the trained model, we then compared the data to our validation set and output those results to the user.



Our workflow diagram is even simpler in comparison. We hash through the 5000 units of data in sets of 20. Once training is finished, we use the test data with the trained model in order to produce our results, again following the batches of 20 over the set.

Dataset

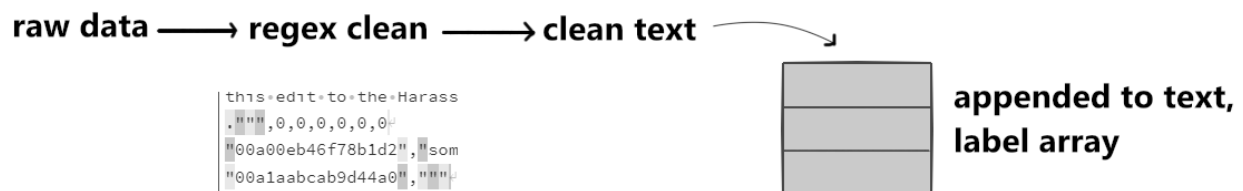
We are using a training dataset provided by Kaggle to train our model. This dataset was provided during a competition aimed at classifying the toxicity of comments. This set contains ~2 million examples of toxic comments and their labels, including toxic, severe toxic, obscene, threat, insult, and identity hate. Due to time constraints, we chose a subset of 10,000 comments with approximately the same distribution as the full dataset. The training comments originate from real Wikipedia comments labeled by human raters. This dataset may be found at [Kaggle's Toxic Comment Classification Challenge](#).

```
You are NOT an administrator. You DON'T have the authority to tell me what to do.",1,0,1,0,0,0
```

The image above is an example line from the training set. Each comment is followed by a binary representation of the toxic groups they belong to, in order of the list above. The first line of the training dataset provides the labels. The test dataset is split by comment ID and comment content. The labels are provided in a separate csv file.

Analysis of Data

The presentation of the training and test sets is very straightforward. Preprocessing includes extracting the unnecessary characters, such as quotations and other separating pieces. These cleaned and separated texts are appended to the appropriate texts, along with the labels.



This leads to training the model with the provided training data. Each comment was examined by a set of graders and given binary scores in six categories: toxic, severe toxic, obscene, threat, insult, and identity hate. One comment may fall into multiple or no categories. Comments that do not fit within any of the six categories are not toxic. In order to generate a target score for a comment, we focused on the two main categories: toxic and severe toxic, averaged across all graders. These two subscores were then averaged with weights 0.7 and 0.3, respectively, to arrive at our target score.

7%	28/400	[18:42<4:07:47, 39.97s/it]	Loss: 0.007811888121068478
7%	29/400	[19:22<4:07:00, 39.95s/it]	Loss: 0.009441673755645752
8%	30/400	[20:02<4:06:19, 39.94s/it]	Loss: 0.008434214629232883
8%	31/400	[20:42<4:05:34, 39.93s/it]	Loss: 0.011698918417096138
8%	32/400	[21:22<4:04:54, 39.93s/it]	Loss: 0.005730693694204092
8%	33/400	[22:02<4:04:11, 39.92s/it]	Loss: 0.008639584295451641
8%	34/400	[22:42<4:03:35, 39.93s/it]	Loss: 0.007104937918484211
9%	35/400	[23:22<4:02:55, 39.93s/it]	Loss: 0.008205351419746876
9%	36/400	[24:02<4:02:14, 39.93s/it]	Loss: 0.01019445899873972
9%	37/400	[24:41<4:01:27, 39.91s/it]	Loss: 0.008741533383727074
10%	38/400	[25:21<4:00:42, 39.90s/it]	Loss: 0.007198614533990622
10%	39/400	[26:01<4:00:05, 39.90s/it]	Loss: 0.010272206738591194
10%	40/400	[26:41<3:59:30, 39.92s/it]	Loss: 0.012058262713253498
10%	41/400	[27:21<3:58:57, 39.94s/it]	Loss: 0.013564079999923706
10%	42/400	[28:01<3:58:11, 39.92s/it]	Loss: 0.011097915470600128
11%	43/400	[28:41<3:57:39, 39.94s/it]	Loss: 0.008012005127966404
11%	44/400	[29:21<3:56:55, 39.93s/it]	Loss: 0.00843116082251072

The above screenshot shows some output data during the training of the model. The comments are evaluated in batches of 20, and the loss for each batch and time taken is displayed. This training occurred while we were still optimizing the model, so the loss is not necessarily accurate for the end product.

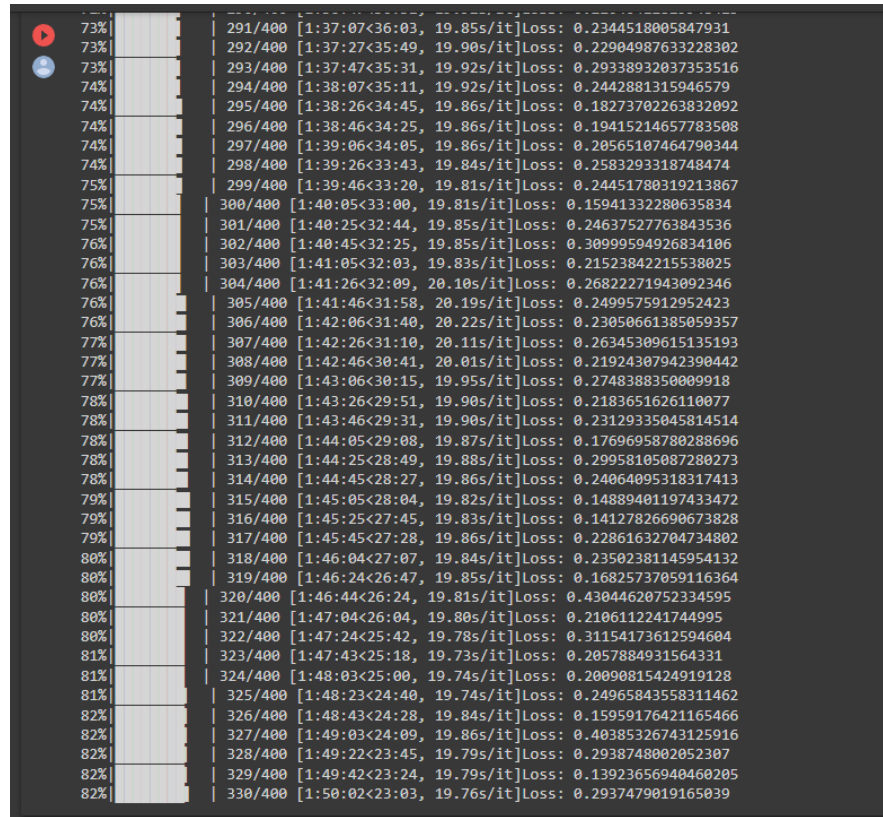
Implementation

The model follows what you would expect from a standard natural language processing model. After preprocessing data with regular expressions to remove unnecessary pieces and placing it into arrays, the model is trained on the given toxic comment data with Charformer. One of the biggest struggles in getting the model to work was matching shapes/sizes between functions within the model. When the model reached a trainable state, it was not without unfavorable loss. Each attempt at fully training the model took many hours, many of which had to be stopped prematurely to tweak values for loss optimization.

The last few days leading up to completion were spent re-running the code to reduce loss and improve accuracy. Though we had eliminated the bugs preventing the model from working, we ran into external issues, such as the program crashing a long way into the training/testing process. This was likely due to Colab being unable to handle the amount of data we attempted to use/too many epochs. Though it was not ideal, we trained on less epochs in some instances to ensure the program would not crash.

Results

Ultimately, our goal was to outperform Bert's toxic comment classifier. In the end, our model did not yield quite as great results. In a way, this was to be expected, as Bert's model uses thousands of tokens, and ours used about 200-300. In the future, our model could be improved by expanding our model's vocabulary. The following is a screenshot of some of the output from our program during epoch 6.



73%		291/400	[1:37:07<36:03,	19.85s/it]	Loss: 0.2344518005847931
73%		292/400	[1:37:27<35:49,	19.90s/it]	Loss: 0.22904987633228302
73%		293/400	[1:37:47<35:31,	19.92s/it]	Loss: 0.29338932037353516
74%		294/400	[1:38:07<35:11,	19.92s/it]	Loss: 0.2442881315946579
74%		295/400	[1:38:26<34:45,	19.86s/it]	Loss: 0.18273702263832092
74%		296/400	[1:38:46<34:25,	19.86s/it]	Loss: 0.19415214657783508
74%		297/400	[1:39:06<34:05,	19.86s/it]	Loss: 0.20565107464790344
74%		298/400	[1:39:26<33:43,	19.84s/it]	Loss: 0.2583293318748474
75%		299/400	[1:39:46<33:20,	19.81s/it]	Loss: 0.24451780319213867
75%		300/400	[1:40:05<33:00,	19.81s/it]	Loss: 0.15941332280635834
75%		301/400	[1:40:25<32:44,	19.85s/it]	Loss: 0.24637527763843536
76%		302/400	[1:40:45<32:25,	19.85s/it]	Loss: 0.3099594926834106
76%		303/400	[1:41:05<32:03,	19.83s/it]	Loss: 0.21523842215538025
76%		304/400	[1:41:26<32:09,	20.10s/it]	Loss: 0.26822271943092346
76%		305/400	[1:41:46<31:58,	20.19s/it]	Loss: 0.2499575912952423
76%		306/400	[1:42:06<31:40,	20.22s/it]	Loss: 0.23050661385059357
77%		307/400	[1:42:26<31:10,	20.11s/it]	Loss: 0.26345309615135193
77%		308/400	[1:42:46<30:41,	20.01s/it]	Loss: 0.21924307942390442
77%		309/400	[1:43:06<30:15,	19.95s/it]	Loss: 0.2748388350009918
78%		310/400	[1:43:26<29:51,	19.90s/it]	Loss: 0.2183651626110077
78%		311/400	[1:43:46<29:31,	19.90s/it]	Loss: 0.23129335045814514
78%		312/400	[1:44:05<29:08,	19.87s/it]	Loss: 0.17696958780288696
78%		313/400	[1:44:25<28:49,	19.88s/it]	Loss: 0.29958105087280273
78%		314/400	[1:44:45<28:27,	19.86s/it]	Loss: 0.24064095318317413
79%		315/400	[1:45:05<28:04,	19.82s/it]	Loss: 0.14889401197433472
79%		316/400	[1:45:25<27:45,	19.83s/it]	Loss: 0.14127826690673828
79%		317/400	[1:45:45<27:28,	19.86s/it]	Loss: 0.22861632704734802
80%		318/400	[1:46:04<27:07,	19.84s/it]	Loss: 0.23502381145954132
80%		319/400	[1:46:24<26:47,	19.85s/it]	Loss: 0.16825737059116364
80%		320/400	[1:46:44<26:24,	19.81s/it]	Loss: 0.43044620752334595
80%		321/400	[1:47:04<26:04,	19.80s/it]	Loss: 0.2106112241744995
80%		322/400	[1:47:24<25:42,	19.78s/it]	Loss: 0.31154173612594604
81%		323/400	[1:47:43<25:18,	19.73s/it]	Loss: 0.2057884931564331
81%		324/400	[1:48:03<25:00,	19.74s/it]	Loss: 0.20090815424919128
81%		325/400	[1:48:23<24:40,	19.74s/it]	Loss: 0.24965843558311462
82%		326/400	[1:48:43<24:28,	19.84s/it]	Loss: 0.15959176421165466
82%		327/400	[1:49:03<24:09,	19.86s/it]	Loss: 0.40385326743125916
82%		328/400	[1:49:22<23:45,	19.79s/it]	Loss: 0.2938748002052307
82%		329/400	[1:49:42<23:24,	19.79s/it]	Loss: 0.13923656940460205
82%		330/400	[1:50:02<23:03,	19.76s/it]	Loss: 0.2937479019165039

Project Management

Work completed

In this increment, the aim of the project required revision. We found an appropriate subject to use Charformer for, changing from Swedish and English translation to toxic comment classification. This meant we had to revamp the entire model. We successfully switched from translation to categorization. We also had to completely rewrite the documentation, as it was done incorrectly in the previous increment.

Responsibilities and Contributions

Hannah – Team leader, heavy debugging work, refined code to working order, loss optimization

Bryan – Heavy debugging work, refined code to working order, loss optimization, model tweaking

Michael - Assisting in document preparation

Sarah - Reviewing structure and code, document preparation, output/model visualization

Issues and Concerns

Initially, we planned to use Charformer to make a Swedish/English translator. It was to be trained on respected translations from parliament proceedings. While attempting to get the model in working order, we realized Charformer could tokenize the pieces of text, but could not reform those tokens back into a proper sentence or translation. Because of this, we had to rework the subject of our project. We found that Charformer works best with categorization. The toxic comment classification project falls perfectly into this. By switching the goal of our model, we were able to continue using Charformer while still properly conveying our understanding of natural language processing and machine learning. Other than that, our biggest issues consisted of a multitude of bugs/small issues that prevented us from training our model, all of which were resolved.

Bibliography

- 1) "Yunje", and "smth". "Giving Multiple Parameters in Optimizer." *PyTorch Forums*, 4 Mar. 2017, <https://discuss.pytorch.org/t/giving-multiple-parameters-in-optimizer/869>.
- 2) Kudo, Taku, and John Richardson. "Sentencepiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing." *ArXiv.org*, Cornell UNiversity, 19 Aug. 2018, <https://arxiv.org/abs/1808.06226>.
- 3) Blob. "Sentencepiece Python Module Example." *Google Colab*, Google, https://colab.research.google.com/github/google/sentencepiece/blob/master/python/sentencepiece_python_module_example.ipynb.
- 4) "Toxic Comment Classification Challenge." *Kaggle*, Google Jigsaw, 19 Dec. 2017, <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/code>.