Internet Toxicology Test

Greg Eastman

Introduction

Since forums on the internet first came into being, they have been both a place of discussion and abuse. To discourage hate and toxic environments these spaces need to be moderated. Unfortunately, moderating has been outpaced by the amount of internet discourse. To address this issue, machine reading can help detect and flag problematic messages on this massive new scale. Therefore, I developed the Internet Toxicology Test. This is an encoder-decoder bidirectional long short-term memory model. It was built on comments from Wikipedia articles. In this domain it had an 86% accuracy on the testing data. Although, it can be fed new data in the form of strings, and it will update a "beta" model as a stream accordingly. Additionally, it will append the new inputs to the original dataset, so there is a record of the information and a new model can be retrained from scratch. To intake new observations, a flask app was made. It forms a web page which allows the user the option of choosing the model they want, feeding it text, and giving feedback. This all can be seen without running code in the "images of app" folder in this GitHub repository. Altogether this work should help to better moderate current and future online forums.

Data

The data for this work came from Alphabet, Google's parent company. It is over 30,000 comments left on Wikipedia articles that have been hand-coded as toxic or non-toxic. Comments on Wikipedia are messages that is not seen when reading an article. It is only seen by the people who are writing articles. They are often used to communicate information or thoughts on the topic of interest. This is a specific domain, so the model may struggle with external validity. To help address this, as well as make it more functional overall, I made the model able to accept new user input and data. This helps it learn information from other areas. To understand this, I will go over how the data is formatted.

The dataset used is a subset of the original csv file provided by Alphabet. I used only the training data provided, and even that was down sampled to get balanced outcomes. The model was built on the binary for toxic and non-toxic as well as the text of the review. The character amount was capped at 400, so any excess characters are not viewed. This is long enough that it encompasses the reviews of the original data and should be enough for any new data. Next, the text was then cleaned using classic text preprocessing techniques like casting to lowercase,

removing stop-words, etc. Although, this csv is only part of the data, as the model can also take in a stream of new observations.

The new user input values come in the form of a string submitted in a text box. The model then predicts whether that string is problematic, and then asks the user if its guess was correct. If accurate, the according prediction is added to the training data. Otherwise, the opposite value of the prediction is added. If the user does not give feedback, then neither the text nor the outcome is appended. Also, when a user gives feedback, a beta model is updated using that one observation. Therefore, the net may grow and learn as more input is given. This covers both the original data and the new data; next I will go through the methodology.

Methodology

This paper performed supervised natural language processing using a long short-term memory model. Like most AI, it was intended to mimic the human brain to get a higher-level performance. It works by having a set of cells that look at the words in the text. Then it stores information on previous words in these cells. This allows for context to be remembered when reading the entire string. This all matters because when people read, we need to be able to remember what came before a word, and it is useful to have a model that can do the same. But we as people do more than just remember what was said, our minds often fill in what is going to be said. Also, sometimes the latter part of the sentence can help to understand the beginning. To model both realities, a bidirectional call was made. This essentially runs the layer both forwards and backwards. Together this all gives the layer output. But my model was more than just a single BiLSTM, it needed to be able to better store a dictionary of words.

Word embeddings are essential to NLP as they give localized meaning to text. Good embeddings can be obtained in a variety of ways. I trained my own using an encoder-decoder. The way this works, is that the front of the model runs the data through and embedding layer. Then it goes through a BiLSTM layer. This forms the encoder. Together these create the meaning of the words. If desired, it can be removed and put onto another model, where definitions from before are preserved. This strategy is often used to get pretrained embeddings. The next part of the model is the decoder. This is made of another BiLSTM layer and a fully connected layer with a sigmoid activation function. This uses the meaning of the words from the encoder to understand the full text and determine its toxicity.

This was implemented in python. I used Keras and Tensor Flow to make the AI while NLTK helped with the data cleaning and other NLP tasks. Once I had made the model, I used Flask to create an app to run the user interface. Now that the methodology and tools are covered, it is time to check on the model performance.

Analysis

When analyzing a model, it is important to use meaningful metrics, so I looked at three different statistics. First was accuracy. This is simply the number of successful predictions over the total number of predictions. This is the most common statistic to look at, because if all that matters is guessing correct the largest proportion of times this is the baseline. The next metric I used was precision. This is a good measure of how often the model guesses a positive result correctly. It is just the number of true positives over the total number of positive predictions. To understand this in the context of this work, precision is the proportion of times that a comment was guessed toxic correctly, over the total number of toxic guesses. Finally, recall was the last metric. This is the number of times the model guessed a comment was toxic over the number of toxic comments it was fed. This metric is important if you care about having no toxic comments get through the algorithm undetected. Next, we will see the results of the model.

All three metrics could be a deciding factor for whoever is using the AI, so I am showing them side by side. Below is a table with the metric and score.

Accuracy	Precision	Recall
.8511	.8114	.8811

The model performance is strong overall. It has an over 85% accuracy just from the text alone. Although, its best value is recall. When given a toxic comment, it predicts it correctly nearly 90% of the time. So, this model may be the best choice if avoiding hateful remarks is paramount, like for websites that may have a younger audience. Last and least, the precision score is just over 80%. This means that the model tends to over-guess that things are toxic. So, if it is more important to not disrupt discourse, this may not be the best choice since it will overestimate the number of toxic comments.

It is important to note that these scores were all calculated on the testing of just Wikipedia comments. This means that on other domains the model will likely not do as well. Therefore, the beta version was created. Unfortunately, I have no metrics for that right now as there are not enough observations to give meaningful statistics. Although, if more data is fed into it by users, it will be better able to learn how these new spaces talk. So, its performance should rapidly improve.

Conclusion

This work concludes that the model it built is accurate and has great recall but is not as precise. Overall, this BiLSTM is best for environments that care more about not letting any bad language slip through, rather than over censoring discourse. Additionally, the user interface built in flask allows the model to take in text from other environments and learn from them. By running these new observations, model performance should increase, especially in previously untrained domains. Also, the new inputs are appended to the training csv, so there is both a

record and way to retrain the net. Overall, I call the model and interface the Internet Toxicology Test, and it should help to make online spaces a place of safer and more positive discourse.

References

1. "Toxic Comment Classification Challenge." Kaggle. Accessed March 12, 2021. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge.
2. Wang, Yequan. "Attention-based LSTM for Aspect-level Sentiment Classification." *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, November 2016, 606-15. Accessed February 4, 2021.
3. Mohan, Vijayarani. (2015). Preprocessing Techniques for Text Mining - An Overview.