Suicidal and Depressed Text Recognition with the BERT Language Model

Logan Ankarberg: `loganankarberg@u.boisestate.edu`

06/20/2023

Abstract

By fine-tuning a BERT model to classify text as suicidal, depressed, or neither, I was able to create an easy-to-use react application that tells the user whether the text is depressed or suicidal. The model had an 88% accuracy on the test data but had some trouble classifying suicidal text. A web scraper was used to collect the data for the model to be trained and tested, and a simple Python script received requests from the client. The client was a simple React application that displayed text, a text box, and a button. I hope that this application can prove useful to mental health professionals if they don't have time to look through text but they still want to know if it could be depressed or suicidal. This website will be a valuable tool, and in the future, I hope to improve it by collecting more data on suicidal and depressed text.

1 Introduction

The issue of mental health has become much more important in recent years. Finding early warning signs of depression and suicidal behavior is crucial in preventing any horrible outcomes. However, recognizing these signs is challenging, because people often struggle to express their emotions and might not even realize they are experiencing depression or suicidal thoughts [1].

To combat this problem, I created a web app that uses a fine-tuned BERT model to classify text as depressive, suicidal, or neither. The model was trained on data from the SuicideWatch subreddit, Depression subreddit, and four other subreddits that contain text-only posts with many words per post so the model has enough data.

My web application is built with Flask as the back end and React as the front end. Users can enter text into a simple text box on the website, which is then sent to the back end for classification with the BERT model. A response is then sent back to the front end with the classifier the model determined. Figure **??** shows how input flows from the user to the model and back.

I hope my web application can be a valuable tool for people struggling with depression or suicidal thoughts, and mental health professionals and researchers. By discovering early warning signs more quickly, we can improve the world's mental health and save lives.

I explain the background, data, and method of this project in the following sections. After that, I will explain my experiment in Section 5 where I fine-tuned a BERT model

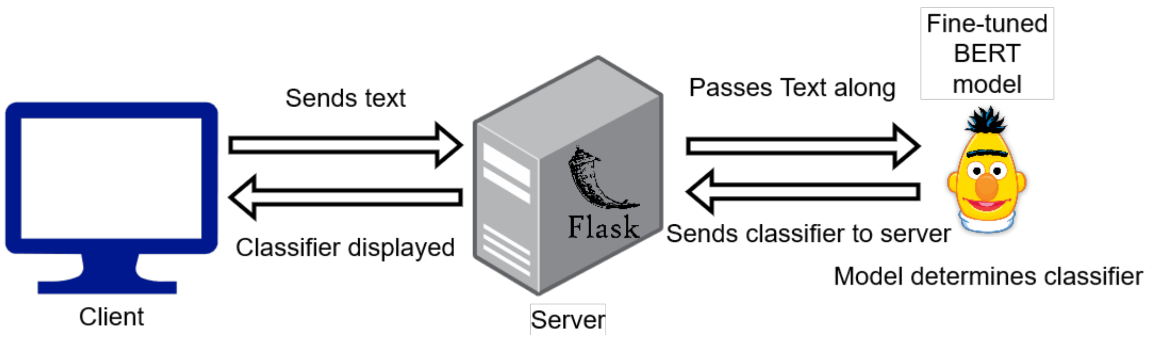

Figure 1: Flow of data between user and model

to predict labels for text. The model didn't work perfectly: metrics show values that are above common baselines, but the model was worst at correctly classifying suicidal text.

2 Background

In order to feed text from the subreddit posts into the BERT model, I used a web scraper that is an adaption from the scraper Haque et al used in a project that used BERT to identify whether the text was depressive or suicidal. Their project is similar to mine, but without the additional classifier of neither. The goal of the additional classifier is so that medical professionals who use this tool don't waste time determining why the model guessed depressed or suicidal, simply because the model didn't think it was either. [4]. My project also differs from Haque et al's because it has an easy-to-use web interface in order to interact with the model. Making this model a resource for the public would be beneficial to many professionals.

3 Data

I used the data described in Haque et al's paper[4], along with text from six more subreddits that aren't depressive or suicidal. These were the nosleep, YouShouldKnow, investing, askphilosophy, StoriesAboutKevin, and TalesFromYourServer subreddits. I collected this data by scraping it with the Reddit API using an adapted Python script from Haque et al. and storing the results in a CSV file [4]. For the text to work with BERT, it had to go through the BERT tokenizer before being fed to the model for training. After being scraped from Reddit, I had to make sure that posts were not duplicates before adding them to my data and then combining all of the data from the subreddits into one CSV.

4 Method

The question I wanted to solve was: how can a language model be used to tell the difference between suicidal, depressed, and text that is neither? In the future, could technology like this be used to flag posts on social media or search queries on Google more accurately?

If so, these platforms could recommend to the user that they call a hotline or 911. More research into this topic could save many people's lives. To model this question, I considered a few different avenues and models. I didn't want to make a neural network from scratch, as there are many models pre-trained on this task that would be easier to implement and work better. I also wanted an easy-to-use interface for the user to interact with so that normal people could use my application.

I ultimately chose BERT because it large model, and it is well suited for calculating the correct classifiers if the input is simply text. It was easy enough to grab the model from the huggingface library. The difficult part was getting the data to input into the model correctly and setting up the model properly for this task. I've used BERT in the past, so it wasn't too difficult to repurpose it for a different task. Another reason that I chose BERT is that is a transformer architecture, which means it is very good at keeping track of long-distance dependency which was the flaw of all models before it. The final reason that I chose BERT is that it only requires one additional output layer to fine-tune to different tasks [3]. For the front end, I used react because it is a very powerful way to make an application that looks amazing.

5 Evaluation

Task & Procedure I used the huggingface transformer library and I used the `bert-base-uncased` because it is the casing didn't matter too much for this task. In order to train the model, I fed all of the web-scraped data that were combined into a CSV into a BertTokenizer. I used Torch with Cuda to train it more quickly. My training data were randomly sampled from the CSV and was 80% of the data, with the other 20% being the test data. I fed the training text into the model with a max length of 128 and batch sizes of 64. Because the BERT model doesn't need many epochs when fine-tuning, my model was trained with three epochs. I used an Adam optimizer with a learning rate of 2e-05, and I printed the model's accuracy on the test data when I was done. I saved the model in a pickle file so it could easily be loaded later in the Flask server. For the front end, I had a simple web page with some text telling the user instructions for how to interact with the model, a text box, and a button. When the button is pressed, a post is made to the backend Python script that feeds the text to the model and sends it back as a response to the client. The answer is then displayed to the user below the button.

Metrics Because of the nature of the task that the model is performing, which is evaluating whether an input text is one of three classifiers, accuracy is the only necessary metric. It will tell us how well the model performs on classifying correctly. At the end of the accuracy evaluation, I created a confusion matrix which is shown in Figure 2 to determine what classifiers the model had the most trouble on. Before doing the test I assumed the model would be better at determining whether a text is neither suicidal nor depressed because it would be more of a difference in tone.

Results According to the confusion matrix that I produced, The model was able to determine a piece of text as neither suicidal nor depressed almost 99% of the time. It was

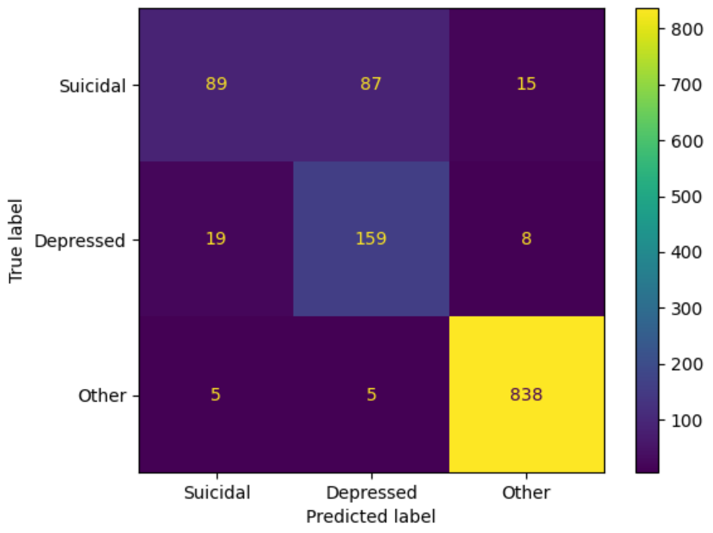

Figure 2: Confusion matrix of model on test data

able to identify depressed text correctly 85% of the time, with most of the errors being false positive suicidal classifications. The model was worst at identifying suicidal text, and it only classified suicidal text as suicidal a mere 46% percent of the time. 85% of those errors were false negatives, cases where the model thought the text was depressed, when in fact it was suicidal. These results are promising, as they show that the model is able to differentiate very well between text that is not depressed and text that is. The most common baseline for this task was 69%, which would be if the model guessed 'other' every time. The random baseline is 33% because there are three classifiers, and if one was randomly selected every time, it would be right a third of the time. The accuracy of the model on the test data was 89%, which is almost 20% above the most common baseline. According to this, my model was able to 'learn' something.

6 Conclusion

My results show that the model is not yet good at determining whether the text is suicidal. This interface will be a useful tool for medical professionals to test whether a piece of text is depressed or not, which will probably save them time. This extra time can be used to focus on what the model is weak at, which is identifying if a depressed piece of text is suicidal. I think that this project is a good step forward for machine learning models to help in the mental health field.

6.1 Future Work

In the future, the first way I would like to improve my model is by feeding it more suicidal data and depressed data. The data that I scraped from Reddit only had about 200 unique posts each. I had around 850 posts that were classified as 'other', and they were only limited to that so the model didn't choose 'other' every time. The other thing that I would probably try to improve this model is testing the latest and greatest models to see if their size would help them identify the suicidal data better. This was a very enjoyable project to work on and I hope to make progress on it in the future.

References

[1] Mental health. `https://www.who.int/news-room/fact-sheets/detail/mental-health-strengthening-our-response`. Accessed: 2023-4-18.

[2] Casper O da Costa-Luis. tqdm: A fast, extensible progress meter for python and CLI. *J. Open Source Softw.*, 4(37):1277, May 2019.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. October 2018.

[4] Ayaan Haque. SDCNL: Deep learning for suicide and depression identification with unsupervised label correction (ICANN 2021).

[5] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández Del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[6] John D Hunter. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.*, 9(3):90–95, May 2007.

[7] Wes McKinney. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*. SciPy, 2010.

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, High-Performance deep learning library. December 2019.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. pages 2825–2830, January 2012.

[10] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-Art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

7 README

This project was built and run on a Windows computer. The same procedure should work for any operating system, but the commands may be slightly different. Download and extract nlp_final_project zip file and open it. Alternatively, you can navigate to the following GitHub repository after accepting a collaboration invitation since it is a private GitHub repo: https://github.com/logananak/nlp_final_project. This project is the exact same as the zip file, but it does not contain the saved_model file. This means that if you import the repository, you have to run the create_model.py after changing the load_from_saved_model boolean to True before proceeding with the rest of the instructions. Download and use Python 3.9.13. Then navigate to the flask-server directory with cd .\flask-server. Use the command python -m venv venv to create a virtual environment. Then run .\venv\Scripts\activate to activate your virtual environment and run pip install -r requirements.txt to download the necessary packages for the project. Optionally, you can download Cuda 11.7 if you want to use your GPU for the model. Start the back end by running python server.py. To start the front end, run cd .. and then cd .\client\ and run npm start. The website should open automatically, but if it doesn't, navigate to localhost:3000. Whatever you type into the text box will be evaluated by the model when you press submit, and the output of the model will be shown below. The following are libraries that I used that made this project way easier and much more enjoyable. I used huggingface's transformers library [10], numpy [5], pandas [7], scikit-learn [9], pytorch [8], tqdm [2], and matplotlib [6].