Bot On My Watch: Twitter Bot Detection using Contrastive Learning in NLP

Team 38
Shilpa Nair / ssnair@usc.edu
Zeeshan Ahmad / zeeshana@usc.edu
Rishibha Bansal / bansalr@usc.edu
Venkatesh Madi / vmadi@usc.edu
Vrundha Subramanyan / vrundhas@usc.edu

1 Introduction

Twitter is one of the most popular social media platforms worldwide, and as reported by Statista [1], there are over 397 million active Twitter users as of September 2021. In 2020, the site's user base was estimated to have grown 8.4 percent from the previous year. The growth in the user base of Twitter can partially be attributed to Twitter being free and easy to use, but another key reason for Twitter's increase in popularity is the capacity to share news on the platform. The content on social media has great potential to influence general public opinions, and this encourages individuals as well as organizations to view and publish content of interest. Besides being used by genuine users, Twitter is also home to several bots. A study on Human-Bot interactions [2] estimated that between 9 percent and 15 percent of Twitter's active monthly users are bots.

A Twitter bot is a type of bot software that controls a Twitter account via automated programs and the Twitter API. These bots can post content or interact with other users in an automated way. While Twitter bots can be helpful in unique ways, there are cases where they can be used unethically and illegally. Some Twitter bots are malicious and serve as instruments for orchestrated opinion manipulation campaigns on social media. Bots can be used to spread misinformation in online discussions of important events, including elections, to skew public opinion, as well as add confusion to debates, such as those about vaccines. With advancements in Natural Language Processing, Machine Learning, and the advent of text generation techniques like BERT [3], many bots exhibit behaviors that even humans find difficult to distinguish. Recent studies (TwiBot-20 [4]) have also shown that as the real-world Twittersphere evolves, so do the bots residing in them, making it more difficult for them to be detected by existing social media bot detection methods.

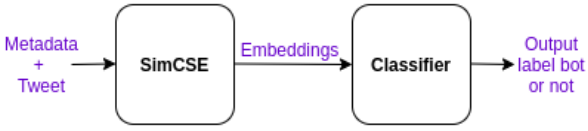

Figure 1: Overview of our end to end model

In our project, we apply the concept of contrastive learning (CL) to detect if a tweet is generated by a bot or if it is a human-generated tweet. We train a CL model using our tweet dataset to generate embeddings for tweets. We train a classifier using these embeddings while freezing the embedding parameters during training. A high-level overview of our work can be seen in Figure 1. We compare our work against two baselines. Our first baseline (baseline-1) is a classifier trained on tweet embeddings generated by the BERT-base model while freezing the embedding parameters. Comparison against baseline-1 helps us understand the impact of CL on the embeddings, and we see that our classifier (accuracy: 0.8778, f1 score: 0.8871, and MCC: 0.7675) performs significantly better than baseline-1 (accuracy: 0.6345, f1 score: 0.5675, MCC: 0.2809). We also compare our classifier against a second baseline (baseline-2) that is a classifier trained on tweet embeddings generated by the BERT-base model, but this time, we allow the tweet embeddings to learn while training the classifier. We see that our classifier still performs better than baseline-2 (accuracy: 0.8500, f1 score: 0.8521, MCC: 0.7006). Both these comparisons show that contrastive learning helps in creating good sentence embeddings and can be useful for NLP tasks such as text classification. Comparison of all the models we train are discussed in sections 3 and 4.

2 Methods

The authors of TwiBot-20 have made an unlabeled sample of their Twitter dataset publicly available through the Botometer [5] website. We contacted the TwiBot-20 team to obtain access to the entire labeled dataset, and the dataset was shared with our team via Google Drive.

The dataset we use is structured such that each instance represents account-level details. Each instance of the data includes account-level metadata, tweets associated with the account, and a bot or human label for the user account. To obtain data in the form of a tweet and bot or human label pair, while preserving the account level metadata, the dataset was re-structured by concatenating the relevant metadata to the tweet and pairing the combination with the corresponding bot or human label for the account for every account in the dataset. The metadata we use includes the screen name, protected status, verified status, the follower count, the following count, favorites count, status count, the count of public lists the account is a part of, whether the profile uses a default backdrop, and whether the profile uses a default image. The dataset was filtered to remove tweets not belonging to the English language. Further data processing involved the removal of re-tweets (when the tweet has not originated at the current account), replacing URLs with a <url> tag, spacing out emojis (to allow the identification of individual emojis), fixing contractions, and identifying extended words (words that have at least one letter repeated three or more times). The resulting dataset was sampled to obtain an even distribution of bot and human tweets to avoid an imbalance in data.

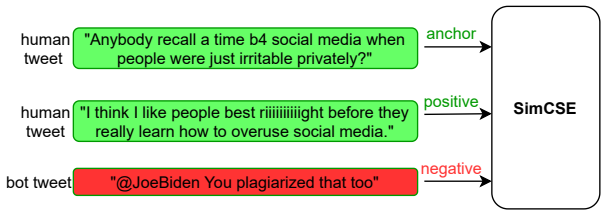

Figure 2: Training SimCSE

Our work uses the contrastive learning-based SimCSE [6] framework to build a model that generates sentence embeddings. Instead of using the models pre-trained by the SimCSE team on the NLI dataset [7] to generate sentence embeddings, we use the SimCSE framework to train our own model using the Twitter dataset we build, using the transformer-based BERT-base model (provided by the HuggingFace [8] transformers team) as our base model. Training this model involves preparing our data in a three-column format which includes "entailment" pairs as positives and a "contradiction" as its negative. Instead of directly using similar sentences as positive pairs and a dissimilar sentence as a negative (like SimCSE does), we define a positive pair as tweets belonging to the same label and a negative as a tweet belonging to a different label. For instance, if a pair of human-labeled tweets form a "positive" pair, a bot-labeled tweet would be its corresponding "negative" as shown in Figure 2. We also see that considering contextual similarity while building positive and negative pairs helps our model differentiate between a bot and a human tweet better than it could when the triplet instances were built without contextual similarity.

To build such triplets, we use the base BERT model to generate embeddings for tweets and construct two similarity matrices - all tweets against bot-labeled tweets and all tweets against human-labeled tweets - by calculating the cosine similarity between each pair of tweet embeddings for both the matrices. Since the resulting matrices are huge, storing them in memory at once would require over 1400 GB of RAM. To bypass this issue, we split the creation of these similarity matrices into batches and parallelize their construction across multiple GPUs, storing only the top two similar results per tweet as shown in Figure 3. We then load the similarity matrices, concatenate them, and build our triplet instances by combining permutations of the top two similar positive and top two similar negative pairs of tweets.

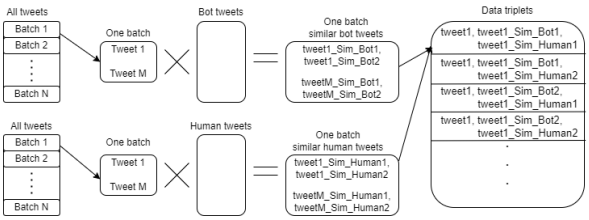

Figure 3: Triplet instance creation for tweet1 (bot tweet). For each batch, we compute the cosine similarity between every tweet in the batch and every bot tweet. Two most similar tweets are selected for each tweet. The process is repeated for human tweets and we combine permutations of the resulting similar tweets to obtain our triplet instances.

We use the resulting data to train our embedding model, and the parameters of the model are fine-tuned based on the "positive" and "negative"

pairs of the triplet data instances. We evaluate our embedding model against our validation set and can see that the embedding model learns to identify features of bot tweets and human tweets and is able to differentiate them well even though they are contextually similar. The embeddings of the positive pairs (same label data) lie closer in vector space than the embeddings of the negative pairs (data with different labels). We train several other embedding models experimenting with different learning rates, and by using models pre-trained by the SimCSE team on the NLI dataset as our base model. We discuss the results in section 4.

Next, we train a classifier to predict if a tweet is created by a bot or a human. The embedding model trained in the previous step is used to generate sentence embeddings which are inputs to the classifier along with the label of the tweet. We freeze the parameters of the embedding model while training the classifier. We evaluate the classifier on our validation and test set by feeding the data into the classifier and comparing the predicted label against the actual label. We compute metrics like accuracy, F1 score, and MCC. We train a classifier for each of the embedding models we create. We discuss the results of this stage in section 4.

3 Experiments

In contrastive learning, the training quality is largely dependent on the quality of similar and dissimilar pairs fed to the model. Initially, all human tweets were considered to be equivalent to each other, and all bot tweets were considered to be equivalent to each other. This meant that there was a possibility of creating a very large number of positive pairs and negative pairs such that any bot tweet could be paired with any human tweet as a negative pair or any bot tweet as a positive pair, and vice versa for human tweets. Training on such a huge dataset would be ideal but was infeasible, hence a random sample of pairs was chosen for training. 30,000 such instances were created as input and this sample was preprocessed by fixing contractions, accurately identifying emojis and replacing urls, hashtags, user handles and numbers with relevant tags that represent each of these categories. But the resulting model was not very promising. When tested against the dev set, it could not differentiate between the bot and human tweets.

The next attempt entailed a more sophisticated approach to identify positive and negative pairs. To make any pair, the focal tweet would be paired with a tweet that was similar to a certain degree to ensure that the context of the tweets in any pair would be as close as possible. This way, any differences between tweets of negative pairs would be attributed to the differences between bots and humans in creating tweets and similarities between tweets of positive pairs would be attributed to the patterns common to that class (bot or human). Since the end goal is to capture human and bot tweeting patterns, retweets can confuse the model if bots and humans end up circulating the exact same text. Retweets are tweets which are reposted by an account as is, and as the text is not generated by the account re-posting them, they are not reflective of that user's manner of tweet creation. Retweets make up 32.38% of training data, 31.67% of dev data and 32.7% of test data. Hence, the process of sampling for all three datasets takes care not to select retweets and also to select only English tweets. For this attempt, 50,000 tweets were sampled and their top 2 similar tweets from each class were found, resulting in 200,000 instances for training the model. The results were significantly better than results of the model based on random pairing.

Filter	BOT - Test	HUMAN - Test	BOT - Dev	HUMAN - Dev	BOT - Train	HUMAN - Train
Total tweets	102927	96926	212950	188575	756619	641791
Retweets	39970	25391	78591	48586	289171	163623
Tweets after retweet filtering	62957	71535	134359	139989	467448	478168
Tweets after filtering for English	43812	44223	95016	84611	325841	297623

Figure 4: Dataset distribution

Based on the superior results achieved when account level meta data was taken into consideration by the other researchers working on bot classification, the next iteration of changes involved concatenating account metadata to the tweet text in a way that was compatible with contrastive learning. This was achieved by fixing the first 10 word indices for 10 metadata fields, followed by the tweet text, which resulted in one long string containing both metadata and tweet. The new pairs of similar and dissimilar tweets were chosen based on this combined knowledge. Additionally, the preprocessing on user names, hashtags and numbers was reverted as these tokens could be of more value and also because replacing them with tags that only highlight the category could misdirect

the similarity evaluation. Also, "extended words" were identified as those words that are exaggerated by having at least one letter repeated three or more times, and marked appropriately as such writing styles may be significant features in distinguishing between bots and humans. This attempt sampled 360,000 tweets and by following the same similarity based pairing, 1,440,000 instances were obtained for training. This dataset is used to train the SimCSE framework on BERT models with different hyper parameters and RoBERTa[9] models with different hyper parameters and the best model is selected from each of these sets. The accuracy values of the similarities of these models on the dev dataset are shown in Figure 5, and the accuracy values, F1 scores and MCC values seen on the test dataset are shown in Figure 7.

4 Results & Discussions

After training several models with varying parameters, it has been found that the classifier trained on top of our constrastive learning model using SimCSE RoBERTa model as base, with 1 epoch and learning rate of 5e-5 outperformed the other models by a considerable margin(as seen in Figure 7). Figure 5 summarizes the accuracy scores of dev dataset on various models trained for constrastive learning. Figure 6 shows accuracy, F1 and MCC scores on dev dataset for the classifier that was trained on the embeddings produced by our contrastive learning model. The test set was evaluated against the top two models(i.e., Bert-2epoch-LR5e-5 and Roberta-1epoch-LR1e-5) with the best accuracy, F1 and MCC scores. Test scores are comparable to dev scores, which shows that the contrastive learning based models generalize well.

Model	Base model	Epochs, LR, Optimization steps	Accuracy
Bert-1epoch-LR5e-5	bert-base-uncased	1, 5e-5, 45000	0.9251
Bert-2epoch-LR5e-5	bert-base-uncased	2, 5e-5, 90000	0.9268
Bert-1epoch-LR1e-5	bert-base-uncased	1, 1e-5, 45000	0.9331
SimCSE-Bert-1epoch-LR5e-5	princeton-nlp/sup-simcse-bert-base-uncased	1, 5e-5, 45000	0.9155
SimCSE-Roberta-1epoch-LR5e-5	princeton-nlp/sup-simcse-roberta-base	1, 5e-5, 45000	0.9050
SimCSE-Roberta-1epoch-LR1e-5	princeton-nlp/sup-simcse-roberta-base	1, 1e-5, 45000	0.9176

Figure 5: Dev scores based on contrastive learning similarity

Baseline1 shows the test accuracy, F1 score and MCC of the classifier built with bert-base-uncased transformer model by freezing its parameters (embedding of tweets are not modified by the classifier). This score shows that contrastive learning contributes significantly to the learning curve of the classifier by providing better sentence embeddings to the classifier. On the other hand, if the same baseline model is allowed to fine-tune the bert-based-uncased transformer model(unfrozen parameters), as depicted by Baseline-2, contrastive learning still outperforms the baseline.

Model	Accuracy	F1 Score	MCC
Bert-1epoch-LR5e-5	0.8381	0.8577	0.68204
Bert-2epoch-LR5e-5	**0.8460**	**0.865**	**0.6994**
Bert-1epoch-LR1e-5	0.8448	0.8625	0.6944
SimCSE-Bert-1epoch-LR5e-5	0.8365	0.8592	0.6868
SimCSE-Roberta-1epoch-LR5e-5	**0.8537**	**0.8728**	**0.7176**
SimCSE-Roberta-1epoch-LR1e-5	0.8587	0.8785	0.7311

Figure 6: Dev scores based on classifier prediction

Model	Accuracy	F1 Score	MCC
Bert-2epoch-LR5e-5	**0.8606**	**0.8683**	**0.7273**
Roberta-1epoch-LR1e-5	**0.8778**	**0.8871**	**0.7674**
Baseline-1	0.6345	0.5675	0.2809
Baseline-2	0.8500	0.8521	0.7006

Figure 7: Test scores based on classifier prediction

5 Conclusions

The results show that contrastive learning can boost the accuracy of normal classifiers and hence, provides more reliable embeddings that are representative of the differences between the two categories, bots and humans. This also indicates that contrastive learning has the potential to be applicable to different NLP problems like text classification, especially where more complex representations, such as those of entire sentences, are needed.

Although this model is built to differentiate between bots and humans, it has also learned to produce representations that can categorize the

bots themselves. If there is a bot detected either through automated or manual means, our model can produce embeddings that can be used to identify similar bots on twitter. Our model will perform significantly better at these unseen situations than a simple supervised language model. Since we are learning to recognize similar bots/ humans, we can use hierarchical clustering or t-SNE to visually represent the embedding space and understand the different types of bots on twitter.

YouTube link: https://youtu.be/qUb9wme36PI GitHub link: https://github.com/vrundha/bot-on-my-watch

References

[1] Statista - *https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/*

[2] Onur Varol; Emilio Ferrara; Clayton A. Davis; Filippo Menczer; Alessandro Flammini. 2017. *Online Human-Bot Interactions: Detection, Estimation, and Characterization*

[3] BERT *https://huggingface.co/docs/transformers/model_doc/bert*

[4] Shangbin Feng; Herun Wan; Ningnan Wang; Jundong Li; Minnan Luo. 2021. *TwiBot-20: A Comprehensive Twitter Bot Detection Benchmark*

[5] Clayton A. Davis1; Onur Varol1; Emilio Ferrara; Alessandro Flammini; Filippo Menczer. 2018. *Botometer*

[6] Tianyu Gao; Xingcheng Yao; Danqi Chen 2018. *SimCSE: Simple Contrastive Learning of Sentence Embeddings*

[7] Alexis Conneau; Douwe Kiela; Holger Schwenk; Loïc Barrault; Antoine Bordes. 2017. NLI datasets. *Supervised learning of universal sentence representations from natural language inference data.*

[8] Hugging face Transformers *https://huggingface.co/*

[9] 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*