# Automated Identification of Social Media Bots Using Deepfake Text Detection

Sina Mahdipour Saravani(✉), Indrajit Ray, and Indrakshi Ray

Colorado State University, Fort Collins, CO 80523, USA
{sinamps,indrajit.ray,indrakshi.ray}@colostate.edu

**Abstract.** Social networks are playing an increasingly important role in modern society. Social media bots are also on the rise. Bots can propagate misinformation and spam, thereby influencing economy, politics, and healthcare. The progress in Natural Language Processing (NLP) techniques makes bots more deceptive and harder to detect. Easy availability of readily deployable bots empowers the attacker to perform malicious activities; this makes bot detection an important problem in social networks. Researchers have worked on the problem of bot detection. Most research focus on identifying bot accounts in social media; however, the meta-data needed for bot account detection is unavailable in many cases. Moreover, if the account is controlled by a cyborg (a bot-assisted human or human-assisted bot) such detection mechanisms will fail. Consequently, we focus on identifying bots on the basis of textual contents of posts they make in the social media, which we refer to as fake posts. NLP techniques based on Deep Learning appear to be the most promising approach for fake text detection. We employ an end-to-end neural network architecture for deep fake text detection on a real-world Twitter dataset containing deceptive Tweets. Our experiments achieve the state of the art performance and improve the classification accuracy by 2% compared to previously tested models. Moreover, our content-level approach can be used for fake posts detection in social media in real-time. Detecting fake texts before it gets propagated will help curb the spread of misinformation.

**Keywords:** Bot detection · Deepfake text · NLP · Deep learning · Security

## 1 Introduction

Social media is extensively being used as a tool of communication and free discussion. The number of active users of Twitter, for example, has increased approximately by a factor of 11 in a period of 9 years. The huge amount of information

This work was supported in part by funds from NIST under award number 60NANB18D204, and from NSF under award number CNS 2027750, CNS 1822118 and from NIST, Statnett, Cyber Risk Research, AMI, ARL, and from DoE NEUP Program contract number DE-NE0008986.


S. Tripathy et al. (Eds.): ICISS 2021, LNCS 13146, pp. 111–123, 2021.
https://doi.org/10.1007/978-3-030-92571-0_7

being propagated world-wide through social media affects the society, public decisions, and their actions [3]. Hence, it is important to prevent malevolent parties from misusing this massive potential in their favor. Popularity of social networks has increased cyber bots and Sybils. Some studies report that 9 to 20% of Twitter users are bots and they contribute to 35% of Twitter's total contents [1,25]. These bots can be manipulated to propagate misinformation and spam, change the stock market value by trending fake information for financial gain, affect the elections for political gain, and more [12]. With the emerging progress in Natural Language Processing (NLP), bots have become more deceptive and harder to detect. Easily available bots that can be readily deployed empowers attackers so that they can perform malicious activities more easily. Consequently, bot detection is critical in social networks [16].

Most research focus on bot detection at the account-level. However, often account-level information is unavailable for privacy reasons. Also, Cyborgs (human-assisted bots or bot-assisted humans), which are common in Twitter [7], can combine normal human behaviour with malicious bot controlled activities with the help of the human operator and the advanced artificial intelligence techniques and evade account-level bot detection systems. Consequently, we focus on using the content to distinguish whether it is generated by a human or a bot.

**Problem Statement.** We present a deep neural network architecture to distinguish between bot-generated and human-generated texts. We use a real world Twitter dataset in this research and consider the detection problem as a text classification problem where given a Tweet, the objective is to determine whether it is written by a human user or generated by a cyber bot.

**Our Approach and Contributions.** Deep learning appears to be the most promising approach for textual content classification due to its automatic feature extraction and holistic language representation as demonstrated by empirical results in NLP [9]. The fact that the most advanced cyber bots have also used and benefited from the fast-paced improvements in machine learning [17] further supports our decision to focus on deep learning techniques to detect and defeat them, as they can leverage the same benefits. We employ a set of neural network architectures for distinguishing deep fake bot-generated Tweets from human-written Tweets. We also present the novel architecture by [19] consisting of BERT, BiLSTM, NeXtVLAD, and two fully-connected layers to show its performance in this classification task. Our contributions are as follows:

- Our models improve the classification accuracy of the best previous models by incorporating a domain-specific pre-trained BERT model, highlighting their efficacy.
- This improvement is achieved on a real world Twitter dataset that includes bot-generated samples that are even difficult for human readers to detect.
- We provide explanation on applying the NeXtVLAD parametric pooling layer – which has proved successful in classification and ranking tasks in computer vision – to NLP problems and assess its usability in a classification architecture for bot detection.
- Our approach can be used for real-time fake text detection.

## 2 NLP for Bot Detection

### 2.1 Challenges of NLP for Bot Detection

Since we only use textual content of posts for detection, our work falls in the category of text classification. In NLP with deep learning, text needs to be first represented by a reasonable numeric vector before any deep learning models can be applied to it. This introduces an additional complexity compared to computer vision applications wherein deep fake detection has been investigated in great depths. Even simplest fake text generation methods like search-and-replace can trick human readers [11]. The more advanced approaches, however, are much more capable and can generate totally new sentences or even interact with human users in an online conversation. The technology behind these advanced bots relies on NLP with deep learning and hence [11] suggests that the best defense against them may be NLP with deep learning itself.

An important challenge in processing social media text is its differences from traditional and formal language. Predominantly, text in social media is short in length and is informal both literally and grammatically. Also, it includes various entities such as hashtags, mention tokens, and emojis. These differences add to the complexity of providing automatic solution to any NLP problem on social media data. The language informality in social media also favors the use of automatic feature extraction and language representation models since it is almost impossible to manually engineer features that can represent words and sentences of an informal infinite natural language. In addition, machine learning can discover statistical patterns in data that are not recognizable by humans but help in detection of machine-generated text [15].

### 2.2 Dataset

Since most researchers have worked on detecting bots at account-level, the number of available fake text datasets is very limited. Cresci's dataset [8], also used in [14,18], is among the few such datasets; however, an important factor in choosing the dataset for us was the quality of the text and how similar the bot-generated Tweets are to human-generated ones. In other words, we wanted a deepfake text dataset that contains text samples generated by recent advanced deep language models. Cresci's text samples [8] follow specific patterns that can be indicator features for being generated by bots which renders them easily detectable and hence unsuitable according to our criteria.

In the deep fake text domain, language models such as GPT, RNN, LSTM, GROVER, etc. have reached the capability of generating high quality text and some studies report that the humans detection rate against these text samples is near chance [2,15]. Researchers have already studied techniques to detect bot-generated deep fake text outside social media [2,4,13,15,28], but to assess their capability in social media, we need to work with in-domain data.

**Table 1.** Example data points from TweepFake anonymized dataset.

| Tweet text | Label |
|---|---|
| The world needs more whale stories. I would love to know what whalefacts are hiding in them | GPT-2 bot |
| I will make [FOLLOWERS OF A RELIGION] victims. They come into the United States but should have been crippled so I flourish. I can do it. @USERNAME #debate | RNN bot |
| It literally what time of gucci shorts or not tolerate Libra slander on my face | Other bot |
| I think if i put my mind to it, I could put a tree in my house like they do at the Cherry hill mall | Human |

We work with the *TweepFake - Twitter deepfake text Dataset* [11] for both model training and evaluation. This dataset [11] (published in Kaggle[1]) contains annotated examples of human-generated and bot-generated Tweets. Tweets are collected from 23 different bots that imitate 17 human accounts. Table 1 shows few examples. The generator bots that produced the fake Tweets are language models such as GPT-2, RNN, OpenAI, Markov Chains, etc. and do not have the aforementioned problems. The data samples are deep fake text examples mimicking genuine Tweets and meet our criteria. This dataset includes 25572 Tweets and is balanced between the bot and human classes.

## 3 Related Work

We discuss related work along two categories: (i) bot detection at content-level, and (ii) fake text detection outside social media.

**Bot Detection at Content-Level.** Authors in [10] work on the *PAN Author Profiling* dataset [23] to detect bot-generated Tweets. Their model uses the pre-trained $\text{BERT}_{\text{Base}}$ model to get contextual embedding of the Tweet and concatenates it with emoji2vec embedding and a few binary features to feed to either a Logistic Regression classifier or a deep neural network classifier. It is worth mentioning that they do not fine-tune BERT representations in their training phase. They report a weighted $F_1$ score of 83.35 in the bot detection task by using this architecture. Authors in [18], focus on content-level classification, but not only based on the Tweet text, but also using Tweet object's metadata such as the number of Retweets and replies or favorite counts to augment the GloVe embedding features for a better classification. They use an LSTM layer to learn sequential features of the Tweet text and concatenate it with metadata features before applying fully-connected classification layers. They also calculate a classification score just by the LSTM's representation and use a weighted average loss based on the two outputs for training. Finally, the authors of [11], who have published the dataset that we use in this work, have drawn attention to detecting deep fake text in social media platforms. In addition to the published dataset,

[1] https://www.kaggle.com/mtesconi/twitter-deep-fake-text.

they also contribute by testing a set of machine learning detection methods on the TweepFake dataset. Their performance results are directly comparable with ours. For a more detailed review of social media bot detection techniques and related work, we refer the reader to [3] and [17].

**Fake Text Detection Outside Social Media.** The following studies investigate the fake text detection outside social media domain but are completely relevant to our task. Authors in [28] present a text generation model called GROVER which is based on GPT-2 and raise the concern about the need to build verification techniques against such generator models. GROVER's generated fake news is even better than human-written disinformation at deceiving human readers [28]. They train and evaluate their model with a fake news dataset that they have crawled from 2016 to 2019. In [2], authors combine available language models to generate fake reviews with desired sentiment for Amazon and Yelp. They study how human readers and machine learning generator-based classifiers perform on detecting these generated reviews. Their findings are that the human readers' performance in detecting those generated reviews was roughly equal to chance and machine learning detection mechanisms, despite performing better than humans, still need much more improvements. Authors of [15] focus on comparing humans and machines in detecting deep fake texts. They base their evaluations on GPT-2-generated text and use BERT as the primary discriminator model. They state that since text generator models are trained to fool humans, despite being successful in achieving that objective, introduce abnormalities that make the detection task easy for automatic discriminators. Their experiments also show that fake text detection is more difficult when facing short-length text.

Comparing these related studies with our work, we study the detection of short deep fake text samples, from real Twitter data, with the objective of detecting bots in social networking platforms. This work is different from detecting bots at account-level [6,7,14,16]. Similar to [10,11] we also use transformer-based models to detect fake text, but unlike [10], we fine-tune all of the model parameters in real-time at training phase. Also, we use a domain-specific pre-trained BERT model, namely COVID Twitter BERT (CTBERT-v2) [22], which is different from [11], and this results in performance improvements.

## 4 Methodology

In this section, we describe our methodology to detect bot-generated text[2]. Our model solely uses a single Tweet's text in order to determine whether it is generated by a bot or a human user.

We do not apply any text preprocessing techniques other than tokenization, as the language representation layer in our architecture, BERT, is capable of producing vector representations for *all* tokens and sub-tokens on the fly. We

[2] Our code for this paper is published in the GitHub repository at https://github.com/sinamps/bot-detection.

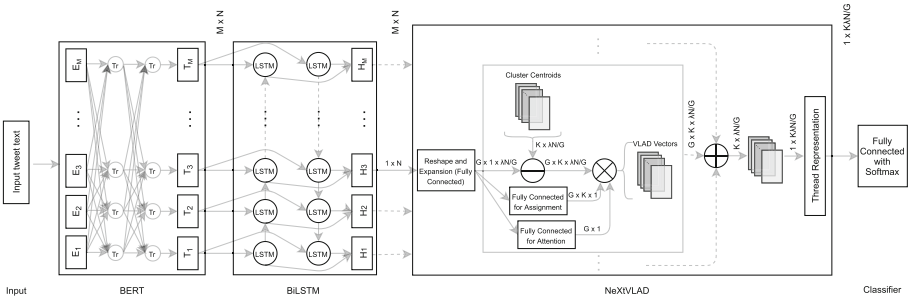


**Fig. 1.** The presented model for detecting bot-generated text content. $M$ is the number of tokens extracted from input text. $N$ is the BERT representation dimension. $\lambda$ is the expansion factor. $G$ is the number of groups to split the input after expansion in NeXtVLAD layer. $K$ is the number of NeXtVLAD clusters.

specifically use the model and tokenizer of CTBERT-v2 [22] from the Hugging Face transformers library [27]. Section 4.1 provides more details about this language model. We used the term "token" instead of "word" here to be more general and even cover strings that are not officially words, such as emojis. Examples of sub-tokens are prefixes and suffixes such as "ed" in "educated".

As practiced in the literature [24], we top this layer by a BiLSTM component to further capture temporal dependencies. These temporal dependencies refer to positional and sequential information as to where the token occurs in the Tweet. Then the outputs of the BiLSTM layer are fed into a VLAD neural component, called NeXtVLAD, for further enhancement. We chose to use NeXtVLAD as our pooling layer inspired by its promising performance in computer vision [20] and the fact that many neural network layers have empirically performed well in both computer vision and natural language processing. The last component of our model is composed of two fully-connected dense layers to perform the final classification of the feature vectors to class labels. In the following subsections, each of these model components are explained in more details. The architecture of our model is depicted in Fig. 1 and is almost identical to the neural network architecture in [19].

### 4.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a language representation model that learns a bidirectional representation from both the left and right contexts of each token and has been proven to enhance the state-of-the-art performance on eleven NLP tasks [9]. This model transforms text tokens and sentences into $N$-dimensional vectors that represent their meanings with consideration of their contexts. This calculation is based on the attention score mechanism [26] that relates the effect of each token to all other tokens and to the task objective. BERT also builds an overall encyclopedic representation for the whole Tweet.

The transformer-based models, like BERT, have been investigated and tweaked in recent years and have been proposed in various configuration and sizes. Their model parameters, that are used to calculate the numerical vector representations for words, are learnt in a pre-training phase. These models benefit from using this pre-trained parameters that are learnt in next sentence prediction and masked language modeling tasks on huge collections of unlabeled data. Although such training results in a very powerful general language model, but as language and text form differs from domain to domain, they can be pre-trained for data from a specific domain to reach even greater performance. In this work, we use a *domain-specific* $\text{BERT}_{\text{Large}}$ model which is pre-trained on COVID-19-related Tweets [22]. Our expectation of gaining performance improvements by using a model that is specifically pre-trained on Tweets is met by our observed results. This approach is used in other studies too [5].

### 4.2 BiLSTM

The BiLSTM layer is used to capture temporal relations (relations showing the sequential position of the token with respect to other tokens) in the sentence in both directions. Even though BERT itself considers both directions in capturing context information, the model may benefit from another sequence-specific component on top of it. Note that we do not encode the whole sequence of tokens into a single representation by the BiLSTM component; instead, we use it to capture the temporal features and incorporate them to update and fine-tune the vector representation of each token. About LSTM's representation for a sentence, we should mention that it updates the representation of each token based on its previous tokens and the representation of the last token is considered as the representation of the sentence itself. Hence, BiLSTM introduces a bias toward tokens that appear at two ends of a sentence. We try to remove this bias by using the VLAD component.

### 4.3 VLAD

Pooling layers intend to summarize the important information from the huge number of features that previous layers produce and remove the redundant variance in the feature space. Maximum pooling and average pooling are the most common pooling layers; however, we incorporate NeXtVLAD parametric pooling and compare it with them in this work. We start this section by presenting some fundamental information about Bag of Visual Words and build on top of it to describe Vector of Locally Aggregated Descriptors (VLAD). Then we explain how we used VLAD in NLP.

**Bag of Visual Words.** Bag of visual words is a simple approach to encode data in the computer vision domain which is very similar to the Bag of Words model in natural language processing that represents sentences as a bag of its words. The procedure in the bag of visual words model is that for all images in a dataset, first they are either partitioned into segments or transformed into

lower-dimension local features such as SIFT [21] descriptors, and then, the Bag of Visual Words model encodes each image into the frequency vector of each of those segments or features.

**Vector of Locally Aggregated Descriptors (VLAD).** Built on top of the Bag of Visual Words model, VLAD model also decomposes all data samples of the dataset into lower-dimension features or segments. However, VLAD goes beyond the feature frequency encoding. It considers a number of centroids ($K$), which is a hyper-parameter of the model, to cluster the feature set into $K$ clusters. In other words, all features from all data instances of the whole dataset are extracted and then clustered into $K$ categories. Now, for representing each data sample, first its feature vectors are extracted and assigned to their nearest cluster centroid. Then, the vector difference of these features from their corresponding cluster centroids are computed. These difference vectors are called residuals. For all feature vectors that belong to the same cluster, their residuals are accumulated together. This produces a set of $K$ accumulated residuals for each data sample which is considered as the representation of that data sample. Each residual is $N$-dimensional just like the feature vectors and hence the representation is of dimension $K \times N$ where $K$ is the number of clusters and $N$ is the dimension of each feature vector.

**NetVLAD.** The VLAD model in its original form cannot be used in a neural network architecture as it is not trainable. The reason behind that is the non-differentiable hard assignment of features to clusters. The idea of NetVLAD was to replace that hard assignment with a softmax scoring function with parameters that can be learned from labeled data. Another important aspect of NetVLAD's procedure to mention, is that by swapping the hard assignment with softmax, now model requires to compute the residuals for each feature vector from all cluster centroids and assign probability scores to them (since it does not know which cluster the feature vector belongs to beforehand). Also, the cluster centroids in NetVLAD are learnt jointly with other model parameters during the training phase. NeXtVLAD, which is described in the next section, is an improved version of NetVLAD.

**NeXtVLAD Component in Our Architecture.** NeXtVLAD first expands its input by a hyper-parameter factor ($\lambda = 4$ in our case), then partitions it into groups of smaller feature vectors and then continues similarly to NetVLAD. The other important difference with NetVLAD is that the soft assignment function includes an additional sigmoid function that computes attention scores over the groups. This scoring module intends to find the input features that are most relevant to make the correct label prediction for each data sample. The NeXtVLAD component in our architecture clusters the feature vectors (token representation vectors) that are produced by the previous layers into $K$ clusters, computes the difference of each token's feature vectors from all of the cluster centroids, and then represents the whole Tweet with these difference vectors. The cluster centroids are initialized randomly but are learnt jointly with other model parameters in the training phase. For detailed description of how NeXtVLAD

works and its mathematical formulation, we refer the reader to its original paper [20]. Comparing NeXtVLAD with NetVLAD, it requires fewer number of model parameters and is more resilient to overfitting [20]. As the output of this layer, we have a $K \times \lambda N/G$ matrix that represents the whole Tweet and we feed it to a classifier to predict the final label.

The NeXtVLAD layer, by its computations performed across all sections of its input, removes the LSTM's bias of assigning higher weights to the most recent tokens.

### 4.4 Classifier

The classification layer in our model consists of two fully-connected layers. We reduce the feature vector dimension and introduce further non-linearity by using a Leaky ReLU activation function in between the two layers. The second layer compresses the information in two nodes. We use a softmax on top of these nodes to compute the probability of belonging to each class.

## 5 Experiments and Results

We implemented our models with PyTorch and Keras frameworks and used three GeForce RTX 2080 Ti GPU cards for running the experiments. We report the details on hyperparameter tuning and model selection in the linked GitHub repository[3].

Authors in [11], in addition to publishing the dataset, have conducted experiments with a set of machine learning algorithm for detecting the bot-generated Tweets. Their results are directly comparable with our results in Table 2. The presented performance scores are computed on the TweepFake test set. As the transformer-based models had the best performance according to [11], we expanded experiments based on transformers by testing other pre-trained weights and other auxiliary model components. Table 3 shows the detailed configurations of models that we have experimented with. Our model (Cfg 1) achieves the best precision and $F_1$ score for the Human class and the best $F_1$ score for the Bot class. Also, its overall accuracy is the best value we reached in our experiments. The accuracy is a good measurement criteria in these experiments, as the dataset is balanced. Our model configurations 1 and 3 improve the accuracy by 2% over the best model from experiments in [11] which is a fine-tuned RoBERTa (also a transformer-based model).

As our results show, our model has introduced a noticeable performance improvement. A comparison of BERT (General-FT) with BERT (Domain-FT) Cfg 3 in Table 2 demonstrates that this improvement is mainly due to the domain-specific pre-training. Comparing NeXtVLAD with two very common pooling layers, average pooling and max pooling (Cfg 5 and Cfg 6), NeXtVLAD

[3] https://github.com/sinamps/bot-detection.

**Table 2.** Results obtained from our experiments for different bot detection mechanisms on the TweepFake test set (the first row is reported from [11]). *FT* means that the model is fine-tuned. *Domain* means that the model is pre-trained on domain-specific data while *General* means that is not the case. *twitter-glove-200* is the pre-trained 200-dimensional GloVe embeddings on Tweets. *Cfg* stands for configuration (the details of these configurations are provided in Table 3). Values are rounded to the nearest hundredths. This table is directly comparable with the results reported in [11].

| Model | Human | | | Bot | | | All |
|---|---|---|---|---|---|---|---|
| | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ | Accuracy |
| BERT (General-FT) [11] | 0.91 | 0.88 | 0.89 | 0.89 | 0.97 | 0.90 | 0.90 |
| LSTM on GloVe (twitter-glove-200) | 0.84 | 0.81 | 0.82 | 0.81 | 0.85 | 0.83 | 0.83 |
| BERT+BiLSTM+NeXtVLAD (Domain-FT) Cfg 1 | 0.92 | 0.91 | 0.92 | 0.92 | 0.92 | 0.92 | 0.92 |
| BERT+BiLSTM+NeXtVLAD (Domain-FT) Cfg 2 | 0.92 | 0.90 | 0.91 | 0.91 | 0.92 | 0.91 | 0.91 |
| BERT (Domain-FT) Cfg 3 | 0.91 | 0.92 | 0.92 | 0.92 | 0.91 | 0.92 | 0.92 |
| BERT+BiLSTM+NeXtVLAD (General-FT) Cfg 4 | 0.90 | 0.87 | 0.88 | 0.87 | 0.90 | 0.88 | 0.88 |
| BERT+BiLSTM+AvgPooling (Domain-FT) Cfg 5 | 0.91 | 0.92 | 0.91 | 0.92 | 0.91 | 0.91 | 0.91 |
| BERT+BiLSTM+MaxPooling (Domain-FT) Cfg 6 | 0.91 | 0.91 | 0.91 | 0.91 | 0.91 | 0.91 | 0.91 |
| BERT+BiLSTM+NeXtVLAD (Domain-FT) Cfg 7 | 0.92 | 0.91 | 0.91 | 0.91 | 0.92 | 0.91 | 0.91 |
| XLNET+BiLSTM+NeXtVLAD (General-FT) Cfg 8 | 0.86 | 0.88 | 0.87 | 0.88 | 0.85 | 0.87 | 0.87 |
| RoBERTa (Domain-FT) Cfg 9 | 0.90 | 0.94 | 0.92 | 0.93 | 0.89 | 0.91 | 0.91 |
| RoBERTa+BiLSTM+NeXtVLAD (Domain-FT) Cfg 10 | 0.89 | 0.94 | 0.92 | 0.94 | 0.88 | 0.91 | 0.91 |
| FastText's Supervised Classifier | 0.83 | 0.81 | 0.82 | 0.82 | 0.83 | 0.82 | 0.82 |

**Table 3.** Details of our model configurations. The *Model* column describes the components of the architecture. *T* stands for the Transformer component, *Bi* for Bidirectional LSTM, *NV* for NeXtVLAD, *Cl* for dense Classification layers, *AP* for Average Pooling, and *MP* for Max Pooling.

| Configuration (Accuracy) | Model | Pre-training | Pooling | Num. of NeXtVLAD clusters | Post-BiLSTM operation |
|---|---|---|---|---|---|
| Cfg 1 (0.92) | T+Bi+NV+Cl | CTBERT-v2 | NeXtVLAD | 128 | Addition |
| Cfg 2 (0.91) | T+Bi+NV+Cl | CTBERT-v2 | NeXtVLAD | 2 | Addition |
| Cfg 3 (0.92) | T+Cl | CTBERT-v2 | — | — | — |
| Cfg 4 (0.88) | T+Bi+NV+Cl | $\text{BERT}_{\text{Large-Cased}}$ | NeXtVLAD | 2 | Addition |
| Cfg 5 (0.91) | T+Bi+AP+Cl | CTBERT-v2 | Avg Pooling | — | Addition |
| Cfg 6 (0.91) | T+Bi+MP+Cl | CTBERT-v2 | Max Pooling | — | Addition |
| Cfg 7 (0.91) | T+Bi+NV+Cl | CTBERT-v2 | NeXtVLAD | 128 | Concatenation |
| Cfg 8 (0.87) | T+Bi+NV+Cl | $\text{XLNET}_{\text{Base-Cased}}$ | NeXtVLAD | 128 | Addition |
| Cfg 9 (0.91) | T+Cl | BERTweet | — | — | — |
| Cfg 10 (0.91) | T+Bi+NV+Cl | BERTweet | NeXtVLAD | 128 | Addition |

shows comparable performance. However, our intention in incorporating BiLSTM and NeXtVLAD is *not* to advocate making complex and computationally expensive pipelines without any insights or intuitions and our work on NeXtVLAD should serve as a report on its comparable performance with other alternatives. The general applicability, advantages and disadvantages of incorporating it to NLP pipelines require further analysis. The performance boost of CTBERT-v2 [22] encouraged us to also experiment with a pre-trained transformer model that is not limited to COVID-19 training data and is pre-trained on English Tweets. We chose BERTweet (Cfg 9 and 10) – a RoBERTa-based model – for this experiment. These models reached the best recall for the Human class and the best precision for the Bot class while they were 1% less accurate compared to CTBERT-v2. We also implemented the LSTM-based approach similar to [18] (second row in Table 2) to report comparable results on the TweepFake dataset, and it was way less accurate due to not using a contextualized language model.

## 6 Conclusion and Future Directions

We address the problem of detecting bots in social media solely based on their generated post content. In order to defend against bots, we studied a set of deep neural network architectures to detect whether a given Tweet is generated by a human user or a software bot. Our best models have improved the performance in terms of accuracy and average $F_1$ score by 2% compared to the previously tested and designed models. The presented NeXtVLAD layer makes our architecture more resilient against overfitting compared to fully-connected layers and is also capable of removing LSTM's bias in favor of latest tokens in text. However, the general applicability of NeXtVLAD layer to NLP problems needs further investigation. Our results also reinforced the benefits of using domain-specific language models as the best option when such a pre-trained model is available or can be produced. Our approach can be used in real-time applications for bot-generated text detection as its only cost is a feed-forward pass through the network. In future, we plan to generate very robust bot detection systems with improved performance. We need to investigate the effects of adversarial attacks on deep fake text detection models to make them robust against advanced cyber bots that may use such hidden noises to bypass the detection systems.

## References

1. Abokhodair, N., Yoo, D., McDonald, D.W.: Dissecting a social botnet: growth, content and influence in Twitter. In: CSCW, pp. 839–851 (2015)
2. Adelani, D.I., Mai, H., Fang, F., Nguyen, H.H., Yamagishi, J., Echizen, I.: Generating Sentiment-Preserving fake online reviews using neural language models and their human- and machine-based detection. In: AINA, pp. 1341–1354 (2020)
3. Alothali, E., Zaki, N., Mohamed, E.A., Alashwal, H.: Detecting social bots on twitter: a literature review. In: IIT, pp. 175–180 (2018)

4. Bakhtin, A., Gross, S., Ott, M., Deng, Y., Ranzato, M., Szlam, A.: Real or Fake? Learning to Discriminate Machine from Human Generated Text. arXiv preprint arXiv:1906.03351 (2019)
5. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: EMNLP-IJCNLP, pp. 3615–3620 (2019)
6. Chavoshi, N., Hamooni, H., Mueen, A.: DeBot: Twitter bot detection via warped correlation. In: ICDM. pp. 817–822 (2016)
7. Chu, Z., Gianvecchio, S., Wang, H., Jajodia, S.: Detecting automation of twitter accounts: are you a human, bot, or cyborg? TDSC **9**(6), 811–824 (2012)
8. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M.: The paradigm-shift of social spambots: evidence, theories, and tools for the arms race. In: WWW Companion, pp. 963–972 (2017)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
10. Dukić, D., Keča, D., Stipić, D.: Are you human? Detecting bots on Twitter Using BERT. In: DSAA, pp. 631–636 (2020)
11. Fagni, T., Falchi, F., Gambini, M., Martella, A., Tesconi, M.: TweepFake: about detecting deepfake tweets. PLoS ONE **16**(5), e0251415 (2021)
12. Gayo-Avello, D.: Social media won't free us. IEEE Internet Comput. **21**(4), 98–101 (2017)
13. Gehrmann, S., Strobelt, H., Rush, A.M.: GLTR: statistical detection and visualization of generated text. In: ACL: System Demonstrations, pp. 111–116 (2019)
14. Heidari, M., Jones, J.H.: Using BERT to extract topic-independent sentiment features for social media bot detection. In: UEMCON, pp. 0542–0547 (2020)
15. Ippolito, D., Duckworth, D., Callison-Burch, C., Eck, D.: Automatic detection of generated text is easiest when humans are fooled. In: ACL, pp. 1808–1822 (2020)
16. Jia, J., Wang, B., Gong, N.Z.: Random walk based fake account detection in online social networks. In: DSN, pp. 273–284 (2017)
17. Karataş, A., Şahin, S.: A review on social bot detection techniques and research directions. In: ISCTurkey, pp. 156–161 (2017)
18. Kudugunta, S., Ferrara, E.: Deep neural networks for bot detection. Inf. Sci. **467**, 312–322 (2018)
19. Lee, H., Yu, Y., Kim, G.: Augmenting data for sarcasm detection with unlabeled conversation context. In: FigLang, pp. 12–17 (2020)
20. Lin, R., Xiao, J., Fan, J.: NeXtVLAD: an efficient neural network to aggregate frame-level features for large-scale video classification. In: ECCV, pp. 206–218 (2018)
21. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
22. Müller, M., Salathé, M., Kummervold, P.E.: COVID-Twitter-BERT: a natural language processing model to Analyse COVID-19 Content on Twitter. arXiv preprint arXiv:2005.07503 (2020)
23. Rangel, F., Rosso, P.: Overview of the 7th author profiling task at PAN 2019: bots and gender profiling in Twitter. In: CEUR Workshop, pp. 1–36 (2019)
24. Srivastava, H., Varshney, V., Kumari, S., Srivastava, S.: A novel hierarchical BERT architecture for Sarcasm detection. In: FigLang, pp. 93–97 (2020)
25. Varol, O., Ferrara, E., Davis, C., Menczer, F., Flammini, A.: Online human-bot interactions: detection, estimation, and characterization. In: ICWSM, pp. 280–289 (2017)

26. Vaswani, A., et al.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
27. Wolf, T., et al.: HuggingFace's transformers: state-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)
28. Zellers, R., et al.: Defending against neural fake news. In: NIPS, pp. 9054–9065 (2019)