Bot-Ani? An RWKV Based Approach For Twitter Bot Profile Detection

Aylon Feraru, Bar Riesel

August 18, 2024

Contents

0.1 Introduction

Twitter (x.com) is a social media platform in which people can manage a profile, follow and be followed others, and post texts and images. An estimated 15% of all twitter users are bots - automated accounts. While some are harmless, others spread fake news on social media, influence the outcome of elections, and propagate conspiracy theories and harmful ideologies. [1]. We care deeply about the task of bot detection, however due to the size of Twitter and the elusiveness of several bots, bot moderation must be at least partially automated.

To that end, several architectures have been proposed. In [2] David Duki ́c et al. utilized BERT contextualized embeddings to develop Logistic Regression and Deep Neural Network models to analyze tweets on an individual basis from the PAN-2019 dataset. Deriving further features from the use of emojis, mentions, etc. In [3] Shangbing Feng et al. proposed a Relational Graph based approach, in which the architecture constructs a heterogeneous graph from follow relationships and applies relational graph convolutional networks. Several Kaggle submissions have tuned DistilBeRT and tinyBeRT over a dataset containing cleaned descriptions. [4]

In this project we've chosen an adequate dataset, processed and extracted its features, and tuned a pretrained RWKV (previously unused architecture to Twitter bot classification), as well as implemented a decision head on top of it.

0.2 Motivation

0.2.1 RWKV, Briefly

The RWKV model combines the advantages of both RNNs and transformers, it's a relatively new architecture and it hasn't been deployed to this particular problem. The following is a breakdown of the model summarized from [5].

The RKWV model is comprised of stackable pairs of blocks - a channel mixing block and a time mixing block. interspersed between them are layer norm blocks and residual connections. R acts as the receiver of past information, and to what degree we wish to retain it. K and V retain the relevance of a token in relation to all others, and the values associated with it respectively. The vectors r_t, k_t, v_t in the time mixing blocks are linear combinations of a linear combination of the current and previous time input.

$$r_t = W_r \cdot (\mu_r \odot x_t + (1-\mu_r) \odot x_{t-1}), \tag{1}$$
$$k_t = W_k \cdot (\mu_k \odot x_t + (1-\mu_k) \odot x_{t-1}), \tag{2}$$
$$v_t = W_v \cdot (\mu_v \odot x_t + (1-\mu_v) \odot x_{t-1}), \tag{3}$$

as are the channel-mixing inputs:

$$r'_t = W'_r \cdot (\mu'_r \odot x_t + (1-\mu'_r) \odot x_{t-1}), \tag{4}$$
$$k'_t = W'_k \cdot (\mu'_k \odot x_t + (1-\mu'_k) \odot x_{t-1}). \tag{5}$$

where W_r, W_k, W_v and their time mixing counterparts are learnable. The output of the time mixing block is given by the following expression, where r_t serves as input to a forget gate:

$$o'_t = \sigma(r'_t) \odot (W'_v \cdot \max(k'_t, 0)^2), \tag{6}$$

The activation function ReLU^2 is used. For the channel mixing block a sequential attention-like mechanism is presented, where instead of a Query matrix a learned weights vector w is presented. Since we don't have the attention pattern that varies per token via the Query and instead of a general learned weight that varies per relative position, to add more adaptability for varying context the key is added to the exponent to modulate the weights and an exponential time decay is introduced. our model treats W as a channel-wise vector that modifies weight by relative position. u is a learnable bias. k_i modulates each token by their key - "what they have to offer". Unlike transformers that offer no decay, RWKV's attention mechanism is more focused on the present by inducing an exponential time decay on weights of past tokens $-(t-1-i)$, and unlike transformers that take on the entire sequence at once it does not incur a quadratic cost (the attention pattern is of size N^2 where N is the sentence size, instead wkv_t is a single number (per "head") that is transferred more from one time to another.

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} \odot v_i + e^{u+k_t} \odot v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}. \tag{7}$$

This state is passed from time unit to time unit recurrently and updated, and is also used to define the output of the time mixing unit:

$$o_t = W_o \cdot (\sigma(r_t) \odot wkv_t). \tag{8}$$

The structure is stackable and repeatable.

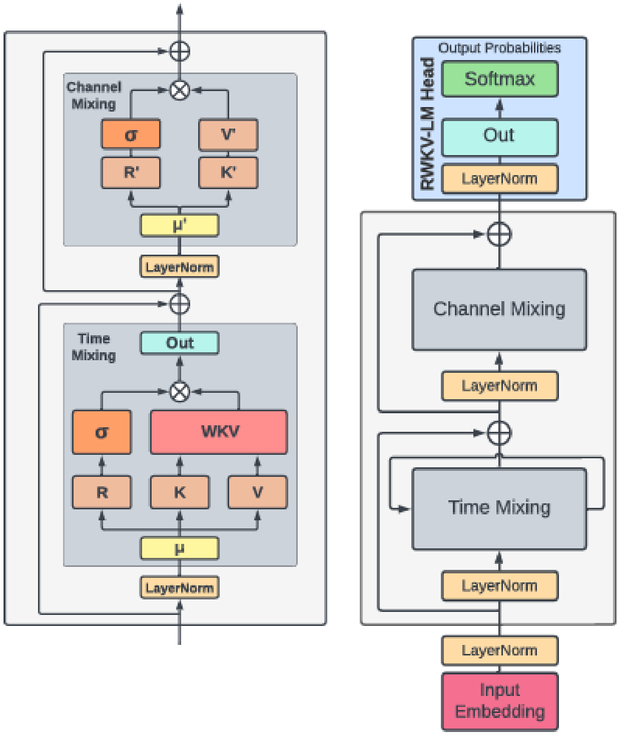

Figure 1: RWKV structure, taken from [5]

0.2.2 Modification for Classification

To achieve classification with the RWKV model, but also take into account the numerical features as well as the extracted numerical features, we needed to fit the model with a classification head. To that end we transformed the numerical and extracted numerical features to be in the same dimension of the RWKV's hidden state, concatenated them and fed them into a rudimentary Fully Connected network that weighs them and produces an output guess.

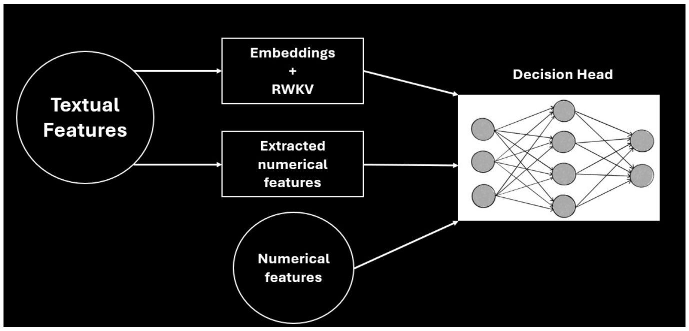

Figure 2: Classification Network Structure

0.3 Experiments and Results

0.3.1 Dataset Selection

Dataset selection proved tricky. Due to privacy concerns, there aren't a lot of Twitter profile datasets freely available. On the one hand, there's the Kaggle Twitter bot detection dataset [6] that we found limited in scope, and pre-processed with the possibility of extracting features removed. We applied for access to the TwiBot dataset [7] and were granted access, however the dataset proved too vast for the scope of the project. We eventually settled on the airt-ml twitter-human-bots dataset [8] since it both had features and the ability to extract more, as well as was about the right size to train with for the scope of the project.

0.3.2 Dataset Processing

The dataset was composed of several fields per-profile, two textual features: profile description (bio) and user-name. as well as several boolean and numerical fields such as date of creation, and average tweets.
We used pandas for the processing. We removed every object that wasn't in English from the dataset, changed the formatting of the date into 4 numerical features ('year','month', 'day', hour'). We applied textual processing by removing gmails, mentions (@username), emojis and links from the textual features, and adjoining a new boolean feature to mention whether or not these were present in the text. From [2] We have seen that these features can help the classification. As illustrated in figures 1 and 2.

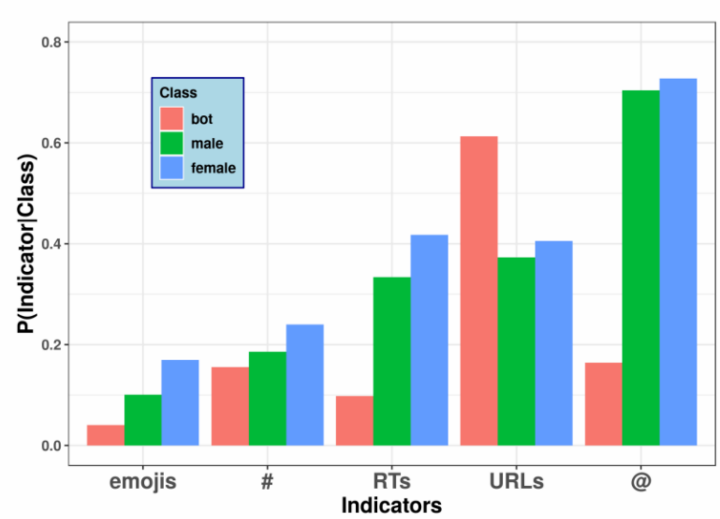

Figure 3: distribution of different indicators in each class of the train dataset. estimated via MLE, taken from [2]

Combing over our dataset for semantic features and their frequency has yielded a different distribution for bots and humans as well.

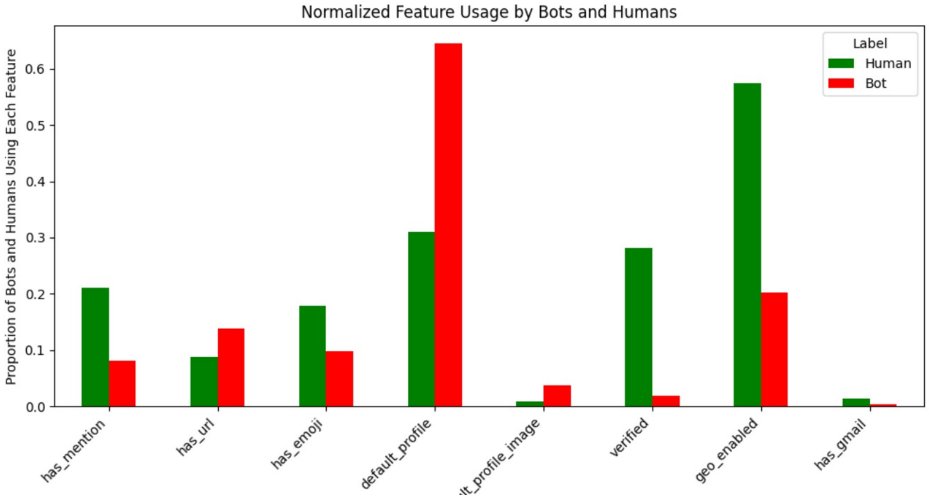

Figure 4: Percentage usage of semantic features in our dataset depending on label

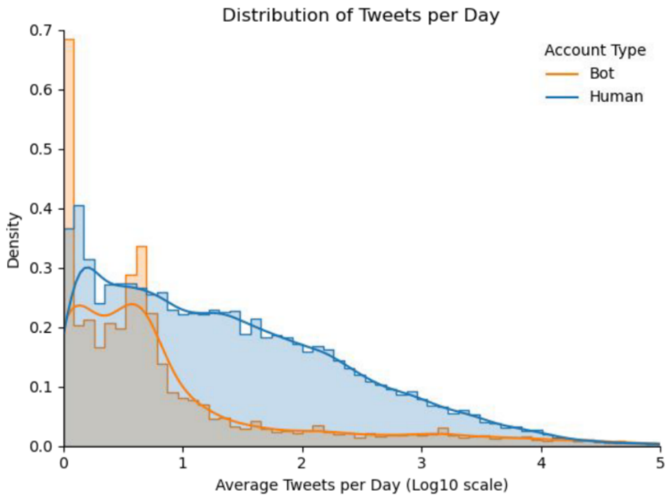

Figure 5: Average tweet distribution in LogScale shows differences between humans and bots. taken from [9]

Worth noting: the data was initially saved and loaded via csv, however the formatting seems to have been distorted upon loading it. We assumed the commas in the textual field caused the confusion, so we decided to use parquet instead - which resolved the issue (and didn't require loading in chunks).

0.3.3 Training Results

The training for classification was done via Cross Entropy loss as a surrogate for 0-1 Loss, since the latter isn't differentiable. The dataset was pre-processed as mentioned prior and then split into a training, validation and test set. For the training we tracked both 0-1 and cross entropy loss for both the training and validation sets, due to long training time we selected and trained the model with different step-sizes for 3 epochs - taking the promising step-size in terms of 0-1 loss in the validation set and training said model further.

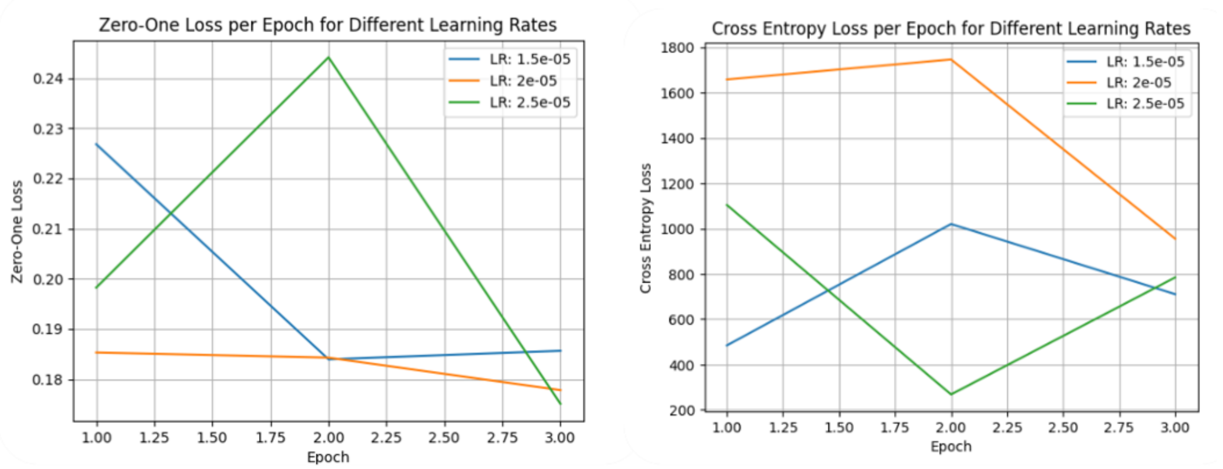

Figure 6: Training Loss After 3 Epochs

We trained the model with the chosen learning rate of $2e-5$ for 12 epochs in total - saving the weights every epoch, and chosen the weights that performed best at epoch 5.
The model with the best validation score achieved 0.81 accuracy on the test set, 72% recall and 72% accuracy. To compare these results seem to be on the same range as the BOTRGCN graph based method that scored an accuracy of 0.8462 [3], and trained on the Twi-Bot 20 dataset - while 3 percentages short, the Twi-bot dataset contains 200 tweets per labeled object whereas our model uses data found in the profile only. In [2] An LSTM tweet classifier bot that trained on the PAN2019 dataset achieved 75% accuracy and 75% precision and recall, and BERT tweet classifier scored an 82 percent accuracy. Interestingly [4] trained a distillBert on the Kaggle bot detection dataset, which contains pre-processed descriptions and some numerical features, and achieved a precision of 49% on the test set. I

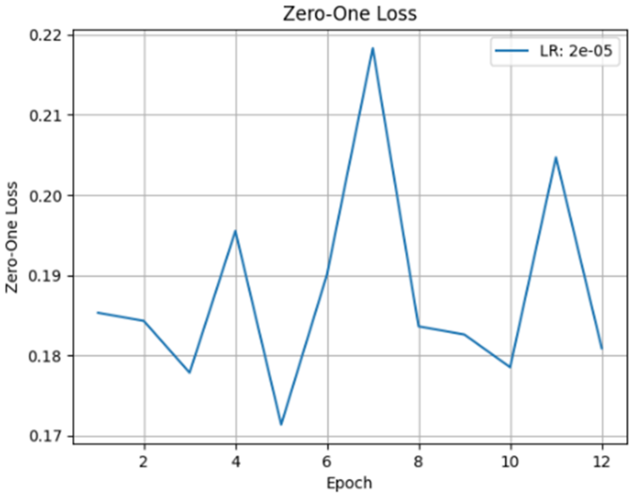

Figure 7: The complete Validation 0-1 loss at the chosen learning rate of 2e-5

think that we can gleam from this rundown that the RWKV architecture is competitive at Bot detection, however the results grow in quality as the dataset grows in size and features and that hinders direct comparison - we struggled to find other projects that used the same ”Twitter-Human-bots” dataset.

0.4 Ethics statement

Student names: Aylon Feraru and Bar Riesel
Bot-Ani? An RWKV Based Approach For Twitter Bot Profile Detection
The project aims to tune a pretrained RKWV model with a dataset we process ourselves,
and tune it for the task of classifying twitter accounts as bots and humans

2a. Twitter Users (End-users)
Social Media Platforms (e.g., Twitter)
Content Creators and Influencers

2b.
End Users: This bot detection system is designed to enhance your experience on Twitter by identifying and removing automated accounts that spread spam or misinformation.
Our goal is to create a safer and more authentic environment for you to engage in meaningful conversations.
Social Media Platforms: Our bot detection model will help maintain the integrity of your platform
by efficiently identifying and removing bot accounts that undermine user trust and spread harmful content.
This system is crucial for preserving
the quality of interactions on your platform, though it requires careful monitoring
to ensure that legitimate users are not unfairly penalized.
We recommend ongoing evaluation and user feedback to refine the model's accuracy
Content creators: This bot detection system aims to protect the authenticity of interactions on Twitter by targeting and removing fake accounts that could distort engagement metrics.
We understand the importance of your online presence,
and we are committed to minimizing the risk of your account being incorrectly flagged
as a bot. Should any issues arise, we provide clear processes for appeals
and corrections to ensure your visibility and influence are preserved

2c.
End Users: The explanation should be provided by Twitter's Trust and Safety team through transparent communication channels, such as platform updates, help center articles, and direct notifications.
Social Media Platforms: The responsibility for explaining the bot detection system to the platform lies
with the AI
development team or the vendor providing the technology.
Content creators: The explanation should be communicated by Twitter's
Support and Communications teams.
They should proactively reach out to content creators and influencers, especially those who
are high-profile or rely heavily on the platform for their livelihoods.

3a. What do you think needs to be added/changed in the Generative AI responses

I think the responses should include
motivation - to explain to the stakeholders why bots are dangerous.
I also think they should involve the Content Creators and End Users in the development and testing of the bot.
I think there needs to be countermeasures that prevent rapid false flags,
such as a trusted human employee verifying an account is a bot after it reaches
a certain number of followers before banning it.
And before banning people, sending a time sensitive verification tasks that the human
account owner could perform to avoid a ban.
(answering captchas, verifying mail, etc).

References

[1] E. Ferrara, "Disinformation and social bot operations in the run up to the 2017 french presidential election," *First Monday*, July 2017.

[2] D. Dukić, D. Keča, and D. Stipić, "Are you human? detecting bots on twitter using bert," in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 631–636, 2020.

[3] S. Feng, H. Wan, N. Wang, and M. Luo, "Botrgcn: Twitter bot detection with relational graph convolutional networks," in *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '21, ACM, Nov. 2021.

[4] G. Dutta, "Twitter bot detection using distilbert." `https://www.kaggle.com/code/gauravduttakiit/twitter-bot-detection-distilbert`, 2024. Accessed: 2024-08-13.

[5] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, X. Du, M. Grella, K. K. GV, X. He, H. Hou, J. Lin, P. Kazienko, J. Kocoń, J. Kong, B. Koptyra, H. Lau, K. S. I. Mantri, F. Mom, A. Saito, G. Song, X. Tang, B. Wang, J. S. Wind, S. Woźniak, R. Zhang, Z. Zhang, Q. Zhao, P. Zhou, Q. Zhou, J. Zhu, and R.-J. Zhu, "Rwkv: Reinventing rnns for the transformer era," 2023.

[6] A. Goyal, "Twitter bot detection dataset." `https://www.kaggle.com/datasets/goyaladi/twitter-bot-detection-dataset/data`, 2022. Accessed: 2024-08-13.

[7] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, "Twibot-20: A comprehensive twitter bot detection benchmark," in *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management*, CIKM '21, ACM, Oct. 2021.

[8] AIRT-ML, "Twitter human bots dataset." `https://huggingface.co/datasets/airt-ml/twitter-human-bots`, 2023. Accessed: 2024-08-13.

[9] Scrapfishies, "Twitter bot detection: Presentation deck," 2023.