Online Twitter Bot Detection: A Comparison Study of Vectorization and Classification Methods on Balanced and Imbalanced Data

Yicong Chen and Jiahe Ling

Abstract— **In this study, we aim to classify whether a Tweet comes from a human or a bot. We are particularly interested in comparing the performances of different word embedding methods and classification models under both imbalance and balanced data through the f1-score and confusion matrix. Text data preprocessing methods tokenization, stop words & punctuation marks removal, and stemming are performed. Text embedding models including Bag-of-words (BoW), TF-IDF, Doc2Vec, BERT, and fastText are used for feature extraction. The classification models including Support Vector Machine, Logistic Regression, and Naive Bayes are also used. The results suggest the power of Transformer based vectorization methods including Doc2Vec, BERT, and fastText when handling imbalanced data.**

I. Introduction

With the development of Artificial Intelligence, it becomes more and more important to detect text from AI for the following reasons. Firstly, the presence of bots on social media platforms can significantly impact public opinion, political discourse, and social interactions. Understanding and accurately identifying bot-generated content can help mitigate the spread of misinformation and manipulation in online spaces. Moreover, such research aids in preserving the authenticity and trustworthiness of online conversations, enabling users to make more informed decisions about the information they consume and share.

In this study, we aim to classify whether a Tweet comes from a human or a bot. We are particularly interested in quantifying and comparing the performances of different word embedding methods including Bag of Words (BoW), TF-IDF, Doc2Vec, BERT, and fastText, and classification models including Support Vector Machine (SVM), Logistic Regression, and Naïve Bayes Classifier under both imbalance and balanced data.

The paper is structured as follows. Firstly, a comprehensive literature review on vectorization methods and classification models is presented. Subsequently, the data sources and models employed in this study are described. Then, the results and figures are presented. Finally, the paper concludes by discussing the research implications, acknowledging the study's limitations, and proposing potential avenues for future research.

Y. C. is with the Department of Computer Science, University of Wisconsin Madison, Madison, WI, 53706 USA (corresponding author to provide phone: (949) 558-4534; e-mail: ychen2229@wisc.edu).

J. L. is with the Department of Computer Science, University of Wisconsin Madison, Madison, WI, 53706 USA (e-mail: jling9@wisc.edu).

II. Literature review

A. Preprocessing

Tokenization is used to break the whole text into small chunks to make it easier to assign meaning (Solangi et al., 2018). Also, Stemming directly converts words into root form to reduce inflectional forms of words in the text. The commonly used models are Affix Removal Stemming, N-gram Stemming, Table Lookup Stemming, etc. In comparison, Lemmatization converts various inflected forms of words into meaningful forms based on the consideration of context (Asghar et al., 2014). Besides, since punctuation marks could not provide any information for analysis, they are removed through techniques named Punctuation Marks Removal (Etaiwi & Naymat, 2017).

B. Vectorization

The vectorization step extracts vectors from the text so that models could use extracted vectors for training and classification. Vectorization could be done by statistical approaches and deep learning. The traditional non-Neural Network vectorization methods commonly used are Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), which generate feature vectors based on the number of the appearance of each word in the text (Liang et al., 2017). Moreover, some neural network-based models like Word2Vec and Doc2Vec are also used for vectorization (Singh & Shash, 2019). Also, some researchers use Bidirectional Encoder Representations from Transformers (BERT), commonly used for sentiment feature extraction (Heidari & Jones, 2020). Finally, designed to be efficient and scalable, fastText is a vectorization method that generates feature vectors based on the internal structure of words, known as subword information (Joulin et al., 2016).

C. Classification Models

Based on the literature, researchers tend to use content, users' profiles, and account usage to predict whether the account is a social bot (Rodríguez-Ruiz et al., 2020). The widely used machine learning methods for the detection of social bots on Twitter include both neural network and non-neural network models: Naive Bayes (NBC), Support Vector Machine (SVM), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long-Term Short-Term Memory (LSTM), etc. (Alothali et al., 2018). Also, if there is no human-bot indicator label for the given text, some researchers would use Botometer to generate the human-bot indicator (Duan et al., 2022). In comparison, some researchers would use unsupervised learning methods, especially clustering methods like DenStream, StreamKM++, etc., in their studies (Khan et al., 2016).

D. Performance Evaluation

Finally, since the detection of social bots on Twitter is a classification problem, the Confusion Matrix with Precision, Recall, Accuracy, F-measure calculated, ROC, and AUC are commonly used to evaluate the performance of models (Alothali et al., 2018).

III. Data

The dataset is obtained from the research group of "Algorithmic Agents in the Hybrid Media System: Social Bots, Selective Amplification, and Partisan News about COVID-19." The tweets are selected based on a word list containing 181 keywords about Covid-19 among all the tweets from March 1, 2020, to May 31, 2020 (Duan et al., 2022).

The data contains 7 variables and 117,940 observations. In this study, we only use 2 variables, which are the "Content" and "SourceatBot7." We consider "SourceatBot7" as a true label since it is not possible to do manual labeling on 10k tweets, and Botometer is an advanced real-time Twitter bot detection tool based on an account's profile and activities. We choose "SourceatBot7" over "SourceatBot5" because a threshold of 0.5 for Botometer would increase the cases that human accounts are mistakenly labeled as bots, and we choose "SourceatBot7" over "SourceatBot9" because a threshold of 0.9 would label much fewer true bots than the threshold of 0.7 (Grinberg et al., 2019).

Since the original data is not balanced (Human: 91,229 vs. Bot: 26,711), we also create a balanced data frame with 53,422 observations by randomly selecting 26,711 observations from all observations with associated with Human tweets. Both imbalance and balanced data are partitioned into 80% training data and 20% testing data. The data preprocessing methods are tokenization, which converts a tweet into a vector of words, removing stop words and punctuations, and stemming (snowball stemming), which reduces a word to its word stem.

IV. Methods

After data preprocessing, we performed embedding of tweets through 5 different methods: Bag of Word (BoW), Term Frequency–Inverse Document Frequency (TF-IDF), Doc2Vec, BERT, and fastText. Then, we trained three classification models including Support Vector Machine (SVM), Logistic Regression (LR), and Naive Bayes (NB).

In specific, since the size of the data is large and limited RAM and time, for 3 neural network-based embedding methods (Doc2Vec, BERT, fastText), we used pre-trained models. The pre-trained Doc2Vec and fastText are Feed Forward Neural Networks (FFNN) with 3 layers (1 hidden), while the BERT is a Transformer-based Neural Network.

For Doc2Vec, the model runs efficiently with epochs = 10 (with default learning rate). For fastText, the learning rate = 0.025 and epochs=5 is a desirable parameters combination.

Due to the limitation computation recourses, for training of BERT, only 700 observations, which is randomly selected, are used. For training of other embedding methods, the whole data is used.

After embedding, both BoW and TF-IDF produced a vector of length 154,011 for each tweet, both Doc2Vec and fastText produced a vector of length 300 for each tweet, and BERT produced a vector of length 768 for each tweet. Then, we trained classification models with the feature vectors we obtained. Besides, parameter tuning is performed on the SVM and NBC, and the SVM with Linear kernel and Bernoulli NBC turned out to have the best performance.

Finally, for each combination of embedding and classification model, the f1-score and confusion is calculated matrix to compare the performance. F1-score is used for two reasons: 1) the f1-score is a popular performance measure for classification, and this study is a classification problem; 2) the f1-score provides robust results for both balanced and imbalanced datasets, which makes it possible to directly compare the performance of models under imbalanced and balanced data through f1-score (Allwright, 2022).

V. Results

A. Results for Imbalance Data (n=117,940)

- Bag of Words

TABLE I. F1-score and Confusion Matrix for BOW

	SVM			Logistic Regression		NBC	
F1-score	0.3851			0.3561		0.3581	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	1831	3525	1486	3870	1614	3742
	Human	2321	15911	1504	16728	2043	16189

- TF-IDF

TABLE II. F1-SCORE AND CONFUSION MATRIX FOR TF-IDF

	SVM			Logistic Regression		NBC	
F1-score	0.3516			0.2409		0.3581	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	1399	3957	812	4544	1614	3742
	Human	1201	17031	572	17660	2043	16189

- Doc2Vec

TABLE III. F1-SCORE AND CONFUSION MATRIX FOR Doc2Vec

	SVM			Logistic Regression		NBC	
F1-score	0.6739			0.6739		0.6742	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	0	5356	0	5356	5	5351
	Human	0	18232	0	5356	10	18222

- BERT

TABLE IV. F1-SCORE AND CONFUSION MATRIX FOR BERT

	SVM			Logistic Regression		NBC	
F1-score	0.6677			0.6835		0.6318	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	5	31	4	32	18	18
	Human	8	95	2	101	36	67

- fastText

TABLE V. F1-SCORE AND CONFUSION MATRIX FOR FASTTEXT

	SVM			Logistic Regression		NBC	
F1-score	0.6842			0.6855		0.6294	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	118	5213	140	5191	3028	2303
	Human	126	18131	159	18098	7199	11058

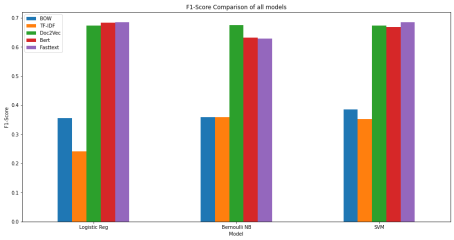

Figure 1

B. Results for Balanced Data (n=53,422)

- Bag of Words

TABLE VI. F1-SCORE AND CONFUSION MATRIX FOR BoW

	SVM			Logistic Regression		NBC	
F1-score	0.6388			0.6437		0.6578	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	3437	1967	3413	1991	3445	1959
	Human	1920	3361	1787	3494	1626	3655

- TF-IDF

TABLE VII. F1-SCORE AND CONFUSION MATRIX FOR TF-IDF

	SVM			Logistic Regression		NBC	
F1-score	0.6520			0.6578		0.6578	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	3494	1910	3481	1923	3445	1959
	Human	1820	3461	1698	3583	1626	3655

- Doc2Vec

TABLE VIII. F1-SCORE AND CONFUSION MATRIX FOR Doc2Vec

	SVM			Logistic Regression		NBC	
F1-score	0.5388			0.5394		0.5280	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	2808	2596	2941	2463	2468	2936
	Human	2330	2951	2459	2822	2080	3201

- BERT

TABLE IX. F1-SCORE AND CONFUSION MATRIX FOR BERT

	SVM			Logistic Regression		NBC	
F1-score	0.4571			0.4509		0.5019	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	25	38	26	37	40	23
	Human	38	39	40	37	46	31

- fastText

TABLE X. F1-SCORE AND CONFUSION MATRIX FOR FastText

	SVM			Logistic Regression		NBC	
F1-score	0.6161			0.6191		0.5910	
Confusion Matrix		Bot	Human	Bot	Human	Bot	Human
	Bot	3272	2132	3268	2136	3526	1878
	Human	1970	3311	1933	3348	2476	2805

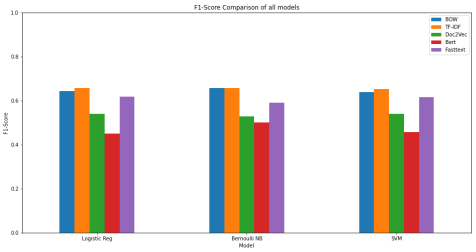

Figure 2

C. Results for Imbalanced Data (n=53,422)

This table displays the F1-score for an Imbalanced Data with the same size as the Balanced data.

TABLE XI. TABLE TYPE STYLES

	SVM	**Logistic Regression**	**NBC**
BoW	0.3800	0.3525	0.3435
TF-IDF	0.3150	0.1867	0.3152
Doc2Vec	0.6707	0.6707	0.6051
BERT	0.6677	0.6835	0.6318
fastText	0.6803	0.6804	0.6294

VI. DISCUSSION AND CONCLUSION

A. Comparison between NN-based and non-NN-based Embedding Models

Comparing **Figure 1** and **Figure 2**, we observed that non-Neural Network-based embedding methods (BoW, TF-IDF) perform much worse in imbalanced data than in balanced data. We first considered the size of the data could contribute to such a difference. However, after controlling the size of imbalanced data the same as balanced data, we still observed that non-Neural Network-based embedding methods (BoW, TF-IDF) performed worse in imbalanced data than in balanced data. We then conclude non-Neural Network embedding method is not capable of handling imbalanced classes.

We would like to further explore the reason behind such a phenomenon. Madabushi et al. conducted research on the BERT, which is a Neural Network-based embedding method, and concluded that if the data is not sufficiently dissimilar, the BERT is capable of handling imbalanced classes with no additional data augmentation (2020). This provides a possible explanation for the phenomenon we observed since all tweets in the data are about Covid-19, which suggests our data is not sufficiently dissimilar so that Neural Network-based embedding methods are capable of handling imbalanced classes with no additional data augmentation. Besides, Padurariu and Breaban conducted a similar study comparing embedding and classification models under imbalanced and balanced data for job classification based on Curriculum Vitae (2019). However, their study does not suggest that non-Neural Network-based embedding methods perform much worse in imbalanced data than in balanced data, which is different from ours.

We believe the methods of obtaining balanced data contribute to such differences. Padurariu and Breaban generate balanced data through oversampling methods (2019). In comparison, we obtained balanced data through an

undersampling method (keeping all the minority classes and decreasing the majority class). In the future, we would further explore whether the performance of BoW and TF-IDF is different in under-sampled data from that in over-sampled data.

B. Comparison among Classification Models

In general, under either balanced or imbalanced data, there is no significant difference in the performance of Logistic Regression (LR), Naive Bayes Classifier (NBC), and Support Vector Machine (SVM). Besides, Figure 6 suggests for balanced data, the best performance of SVM (f1=0.6520), LR (f1=0.6578), and NBC (f1=0.6578) is achieved with the TF-IDF embedding. In contrast, Figure 5 suggests for imbalanced data, the best performance of SVM and LR is achieved with the fastest embedding, and the best performance of NBC is achieved with Doc2Vec embedding.

C. Comparison between Imbalanced and Balanced Data

By calculating the sensitivity, the NN-based embedding methods are more likely to be influenced by imbalanced data. Among all NN-based embedding methods, Doc2Vec seems to be more vulnerable to imbalanced data than other embedding methods. For example, the sensitivity of predicting Bot is nearly 0 for Doc2Vec. Besides, among the 3 classifiers we used, Naive Bayes seems to be less sensitive to imbalanced data compared with the other 2 classifiers (the average sensitivity of Bot prediction of Naive Bayes is greater than the other 2 classifiers). However, for balanced data, there are no significant differences between embedding and classification models.

D. Comparison among non-NN-based Embedding Models

The Bag of Words is the simplest embedding method that generates the vector based on the number of appearances of the word. TF-IDF is like Bag of Word embedding but also takes how rare the word in the text into account. Our study suggests that BoW performs better than TF-IDF under imbalanced data, while TF-IDF performs better than BoW under balanced data.

E. Comparison among NN-based Embedding Models

Doc2Vec learns vectors for words in a way that captures their meaning and the relationships between words in a document. BERT uses a transformer architecture and is trained on a large corpus of text data to learn general-purpose language representation. fastText uses a shallow neural network to learn word vectors in a high-dimensional space, considering sub-word information.

Based on the results, we believe for classifying tweets from Bots and Humans, Doc2Vec and fastest have similar performance, even though they generate the feature vector based on different algorithms. In comparison, even though more complex than Doc2Vec and fastText, BERT seems to be more powerful. Due to the limitation of RAM size, we only trained BERT with 560 (700 * 0.8) observations, while training other embedding methods with over 10k observations. However, the performance of BERT is more significant than other embedding methods. So, we believe if given enough RAM and trained BERT with full-size data, BERT could result in the best results.

In conclusion, BERT is a more powerful and complex method for text embedding, even though it takes a long runtime and consumes a huge amount of RAM since it is based on a deep neural network architecture (transformer). In comparison, Doc2Vec and fastest do not take a large running time while resulting in an acceptable performance. For choosing neural network-based embedding methods, people should balance between running cost (time, RAM) and the model performance.

F. Future Improvements

First, if given enough RAM, we could train BERT with more training data, which may result in a better performance. We would also further explore whether the performance of BoW and TF-IDF is different in under-sampled data from that in over-sampled data. Besides, in this study, we only focus on the comparison of embedding methods. In the future, we could also implement neural network-based classification models and carry the comparison between each combination of embedding method and classification model. Moreover, we could also introduce more variables like users' social network, gender, age, etc. to identify whether the tweet is from a Human or a Bot. Finally, some data augmentation methods for text could be explored to handle the imbalanced data.

REFERENCES

[1] Allwright, S. (2022). What is a good F1 score and how do I interpret it? Retrieved from https://stephenallwright.com/good-f1-score/

[2] Alothali, E., Zaki, N., Mohamed, E. A., & Alashwal, H. (2018, November). Detecting social bots on twitter: a literature review. In 2018 International conference on innovations in information technology (IIT) (pp. 175-180). IEEE.

[3] Asghar, M. Z., Khan, A., Ahmad, S., & Kundi, F. M. (2014). A review of feature extraction in sentiment analysis. Journal of Basic and Applied Scientific Research, 4(3), 181-186.

[4] d'Sa, A. G., Illina, I., & Fohr, D. (2020, February). Bert and fasttext embeddings for automatic detection of toxic speech. In 2020 International Multi-Conference on:"Organization of Knowledge and Advanced Technologies"(OCTA) (pp. 1-5). IEEE.

[5] Duan, Z., Li, J., Lukito, J., Yang, K. C., Chen, F., Shah, D. V., & Yang, S. (2022). Algorithmic Agents in the Hybrid Media System: Social Bots, Selective Amplification, and Partisan News about COVID-19. Human Communication Research.

[6] Etaiwi, W., & Naymat, G. (2017). The impact of applying different preprocessing steps on review spam detection. Procedia computer science, 113, 273-279.

[7] Heidari, M., & Jones, J. H. (2020, October). Using bert to extract topic-independent sentiment features for social media bot detection. In 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) (pp. 05420547). IEEE.

[8] Khan, W., Daud, A., Nasir, J. A., & Amjad, T. (2016). A survey on the state-of-the-art machine learning models in the context of NLP. Kuwait journal of Science, 43(4).

[9] Liang, H., Sun, X., Sun, Y., & Gao, Y. (2017). Text feature extraction based on deep learning: a review. EURASIP journal on wireless communications and networking, 2017(1), 1-12.

[10] Madabushi, H. T., Kochkina, E., & Castelle, M. (2020). Cost-sensitive BERT for generalisable sentence classification with imbalanced data. arXiv preprint arXiv:2003.11563.

[11] Rodríguez-Ruiz, J., Mata-Sánchez, J. I., Monroy, R., Loyola-González, O., & LópezCuevas, A. (2020). A one-class classification approach for bot detection on Twitter. Computers & Security, 91, 101715.

[12] Padurariu, C., & Breaban, M. E. (2019). Dealing with data imbalance in text classification. Procedia Computer Science, 159, 736-745.

[13] Singh, A. K., & Shashi, M. (2019). Vectorization of text documents for identifying unifiable news articles. International Journal of Advanced Computer Science and Applications, 10(7).

[14] Solangi, Y. A., Solangi, Z. A., Aarain, S., Abro, A., Mallah, G. A., & Shah, A. (2018, November). Review on natural language processing (NLP) and its toolkits for opinion mining and sentiment analysis. In 2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS) (pp. 1-4). IEEE.