A Four Feature Types Approach for Detecting Bot and Gender of Twitter Users

Notebook for PAN at CLEF 2019

Johan Fernquist

Swedish Defence Research Agency
johan.fernquist@foi.se

Abstract. The main ideas of our classification model used in the PAN Bot and Gender profiling task 2019 was to combine different feature types with the ambition to detect different styles in writing to distinguishing bots, females and males from each other. We included both word and character TF-IDF features together with compression and tweet features. As classification algorithm we used the CatBoost method. We trained two models, one for the English data and one for the data in Spanish. We achieved highest accuracy with our English model. Both models performed better in distinguishing bots and humans rather than distinguishing females and males. For both languages we achieved an higher accuracy of the bot or human classification rather than the female or male classification.

Keywords: Bot detection · Gender profiling · Twitter

1 Introduction

For several years, bots have been used for a large variety of purposes. Initially their purpose were to automate otherwise unwieldy online processes which could not be done manually, and have now become known commonly for mostly being used for commercial purposes such as directing Internet users to advertisements and posting spam in different social media channels. Bots are also often used to further illegal activity such as collecting data from users for criminal gain. Bot detection is therefore important for a variety of security purposes. Bot detection has for example been used when monitoring large events such as elections, with the aim to prevent influential operations [4]. Gender profiling from text is an important step in author profiling and can also be used for marketing and commercial purposes.

In this notebook, we will present the necessary steps for reproducing our model used in the *PAN*[2] 2019 *Bot and Gender profiling*[13] task. We also briefly describe what we hope to capture with the different types of features. The concept of the model is illustrated in figure 1.

1.1 Previous work

There is a lot of research on bot detection such as [5] where a large variety of features which have seen to perform well in previous researches are combined. In [6], two type of features are used - meta-features and tweet features. In [15] a total of 1,150 different features are used to train a supervised machine learning model to bots. For example, the features consists of part-of-speech-tags (POS), time features such as the statistics of times between consecutive tweets, retweets, and mentions and entropy of words in a tweet. We have taken these feature types in consideration while we developed our own model. Gender classification from text is a well-researched problem. In [7], it is clear that POS-tags is an important type of features when doing gender classification.

2 Model

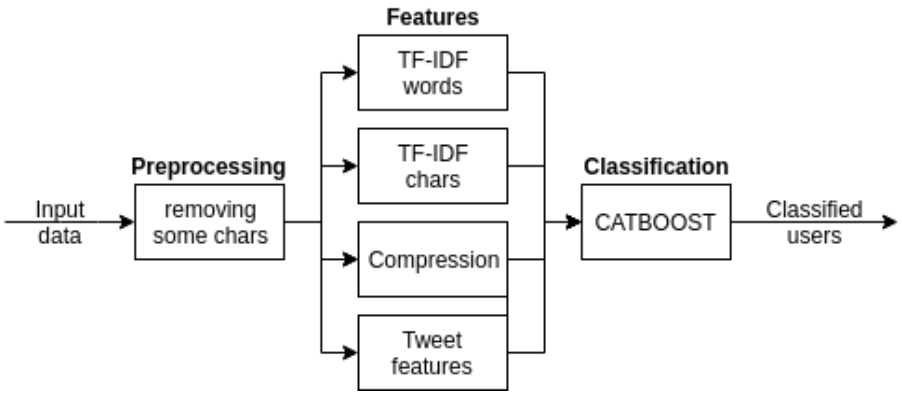

Fig. 1. Work flow of model

The model uses four different feature types: Term frequency-inverse document frequency (TF-IDF) on both word and character level, compression features and tweet features. The different feature types and the used supervised classification model are described below.

After doing some tests with a two step classifier (first bot or human, and then female or male) we decided to create a classification model which only has one classification step and classified bot, female or male directly. This decision was made since we did not want to train several models and during our testing phase we did not see any performance improvements using the two step classification model.

We created individual TF-IDF models (both for chars and words) for the English and the Spanish dataset. The list of pronouns used in the twitter features were also language dependent.

2.1 Data preprocessing

Both training and test data for the task was stored in an xml file. Every tweet for every user was preprocessed by removing all markers for tabs and citation characters (").

2.2 Features

We calculated the feature types for the training and testing users and then concatenated feature vectors for every feature type and user. Each of the feature types are described in detail below.

Term frequency-inverse document frequency (words) TF-IDF is a statistical model which calculates the importance of words in a corpus and valuates words that occurs more often in fewer documents higher. For a complete description of TF-IDF, see [11]. For this task we trained a TF-IDF model with all our training data. Then, for each user in the training data, we concatenated all their tweets into one string and calculated the TF-IDF values for every training and testing user. The TF-IDF model saved n-grams from 1 to 3 and due to time efficiency a maximum of 2000 features were used. Since it seems unlikely that the occurrence of a term is as significant as it's importance, sublinear term frequency was applied, as well as smooth inverse document frequency (meaning that every inverse document frequency is increased by 1). The TF-IDF features were used with Python's Scikit learn[9]. With the TF-IDF on words features, we hope to capture that bots, females and males care to discuss different type of topics and that bots might have a more compressed feature vector i.e. uses several terms more often and have a decreased variety of words used compared to humans.

Term frequency-inverse document frequency (characters) For the TF-IDF features weighted on characters, the approach of the TF-IDF model is the same as described above but instead of calculating the importance of words, the model is calculating the importance of combination of characters. We included character n-grams from 1 to 4 and we did not want to include uncommon character n-grams so we set a minimum document frequency of 20 percent with a maximum of 2000 features. By using TF-IDF on chars as features, we mainly hope to catch the different uses of blank space, and different symbols in conjunction with letters and digits.

Compression features Compression features were used by compressing the concatenated tweets of each user and do different statistical calculations of the compressed tweets. The reason for including compressing features was based on the assumption that bots might communicate in a more monotonous and repetitious way compared to humans. Especially spambots are more likely to just post the same tweet over and over again maybe not changing the content

at all. We wanted the compression features to catch that kind of behavior by detect a difference in compression ratios between human and bot accounts.

We used Python's *zipfile* module to compress every users' own concatenated tweets into the three different compression methods *Deflated*, *BZIP2*, and *LZMA* which are all included in the module. To obtain the compression feature vector for a user, we concatenated the following entities giving us 19 features:

- Original size (size of all concatenated tweets of a user before compression)
- Compression size for every compression method
- Mean, median, popularity standard deviation, standard deviation, max value and min value for the compression sizes
- Normalized compression (each compression size divided by original size)
- Mean, median, popularity standard deviation, standard deviation, max value and min value for the normalized compressions

Tweet features The tweet features consist of a variety of features connected to the attributes of a user's way of tweeting and the content of the tweets. We have already done some classification regarding bot detection on tweets in [5], but in this task we have no time stamps for the tweets or meta data of the users, and therefore some of the features differs from our previous method. We have also included some additional features such as part-of-speech tags and pronouns.

Several of the attributes calculated for a user consist of vectors, and these vectors have been represented as features by calculating statistics of the vector. The statistics calculated for the vectors are always mean, median, popularity standard deviation, standard deviation, maximum value and minimum value. All tweet features are listed below:

- Retweet ratio (number of tweets that are retweets divided by number of posted tweets)
- The character length of all tweets concatenated
- Shannon entropy[14] of all tweets concatenated
- Number of unique words for all tweets
- Number of tweets that have been truncated during the crawling process. They are always finished with a character showing three dots (...).
- Number of different characters the tweets are started with
- Number of unique starting character (including only letters and numbers)
- Whether or not the user always starts the tweet with a mentioning of another user
- Number of different characters the tweets are finished with
- Number of tweets mentioning the word *bot*
- Number of unique hashtags used divided by the total number of used hashtags
- Number of unique hashtags used divided by the total number of tweets
- Number of unique users mentioned divided by the total number of mentioned users
- Number of unique users mentioned divided by the total number of tweets

- Number of unique tweets published divided by the total number of tweets
- Number of unique 30 character beginnings of tweets
- Number of unique 8 character beginnings of tweets
- Number of tweets without including any hashtags, mentioning and URL:s or being a retweet, divided by the total number of tweets
- Number of unique emojis used
- Number of unique emojis used divided by the total number of emojis used
- Number of unique characters to end tweets with
- Number of unique URL:s in tweets
- Number of unique URL:s in tweets divided by the total number of URL:s in tweets
- Number of unique domains linked to
- Number of unique domains linked to divided by the total number of linked domains
- Statistics of number of URL:s per tweet
- Statistics of length of tweets
- Statistics of number of mentionings per tweet
- Statistics of Shannon entropy per tweet
- Statistics of number of hashtags per tweet
- Statistics of number of words per tweet
- Statistics of number of pronouns per tweet
- Statistics of number of upper case letters per tweet
- Statistics of number of lower case letters per tweet
- Statistics of number of blank space per tweet
- Statistics of number of digits per tweet
- Statistics of number of row breaks per tweet
- Statistics of number of tweets between two tweets including a hashtag
- Statistics of number of tweets between two tweets including a URL
- Statistics of number of tweets between two tweets including a mentioning
- Statistics of number of tweets between two tweets including a question sign
- Statistics of number of tweets between two tweets being retweets
- Statistics of Levenshtein distance between every following tweets. Read more about the Levenshtein distance in [1]
- Statistics of number of Part-of-speech (POS) vector where every element in the vector corresponds to the occurrence of a specific POS-tag. POS-tagging is done with the Natural language toolkit[8].

Some features' denominators are increased by 1 to prevent division by zero. The feature vector for the tweet features consist of 139 features.

With the tweet features, we hope to distinguish the bots and humans from each others in many ways. We went through the labeled data manually and could for example see that there often were accounts which always started their tweets with a mentioning, or always retweeted someone. In our labeled data we also saw that women were using emojis more frequently which motivated us to implement the features regarding emojis. With the hypothesis that a bot wants to contact and be seen by as many users as possible (for commercial purposes for example)

the features concerning the use of mentionings and hashtags are important. If an account is used for generating traffic to a website (which could be likely for a spambot), the number of different URL:s posted would be reasonably small, but should occur in several tweets. The Levenshtein feature, the statistics of number of tweets between two tweets including hashtags, URL:s etc. and the entropy features are all used for detecting the content is changed between tweets. It seems more reasonably that a bot would not change the content of the tweets as a human.

2.3 Classification algorithm

Initially we used the Random forest algorithm for classification, but we later discovered that CatBoost gave us better performance. The CatBoost algorithm is based on gradient boosting over decision trees. The CatBoost classification algorithm is further described in [3].

For the CatBoost classifier, we used our training data for training the model, and to prevent the model from overfitting, we used the test set as validation data. Since the evaluation metric for the PAN Bot and gender task would be accuracy, we chose accuracy to be the metric to select the best final model after a total of 5000 iterations.

3 Experiment and results

We parsed every tweet for every user training and test users. Our TF-IDF models were trained with the tweets from our training users, and then calculated the TF-IDF, compression and tweet features for all of our users. This gave us a total of 4158 features calculated for each of the users. We let the CatBoost model learn for 5000 iterations training on our training users and validating on our test users. We then saved the model giving us the best accuracy for the validation set. This was done for English and Spanish separately and resulted in two different models. These two models were then used for the dataset provided in the TIRA[10] environment and the results are shown in table 1.The Low Dimensionality Statistical Embedding (LDSE) baseline described in [12] is also included in the table.

Table 1. Result of bot and gender classification

	English		**Spanish**		
	Human/bot	**Female/male**	**Human/bot**	**Female/male**	**Average**
Own model	0.9496	0.8273	0.9061	0.7667	0.8624
LDSE	0.9054	0.7800	0.8372	0.6900	0.8032

It is clear that our model performs better on the English data compared to the data in Spanish. It might be several reasons for this. Since the TF-IDF

features are the only language dependent features, the signals of the English language regarding gender profiling might be harder to catch in Spanish. There might also be the case that the data set in Spanish is more complex, making that problem harder to solve. The bot or human classification seems to be an easier classification task for our models for both languages compared to the gender classification. It is also clear that our model performs better on all of the different tasks compared to the LDSE baseline.

References

1. Black, P.E.: Dictionary of algorithms and data structures. National Institute of Standards and Technology Gaithersburg (2004)
2. Daelemans, W., Kestemont, M., Manjavancas, E., Potthast, M., Rangel, F., Rosso, P., Specht, G., Stamatatos, E., Stein, B., Tschuggnall, M., Wiegmann, M., Zangerle, E.: Overview of PAN 2019: Author Profiling, Celebrity Profiling, Cross-domain Authorship Attribution and Style Change Detection. In: Crestani, F., Braschler, M., Savoy, J., Rauber, A., Müller, H., Losada, D., Heinatz, G., Cappellato, L., Ferro, N. (eds.) Proceedings of the Tenth International Conference of the CLEF Association (CLEF 2019). Springer (Sep 2019)
3. Dorogush, A.V., Ershov, V., Gulin, A.: Catboost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363 (2018)
4. Fernquist, J., Kaati, L.: Online monitoring of large events. In: 2019 IEEE International Conference on Intelligence and Security Informatics (ISI). IEEE (2018)
5. Fernquist, J., Kaati, L., Schroeder, R.: Political bots and the swedish general election. In: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI). pp. 124–129. IEEE (2018)
6. Gilani, Z., Farahbakhsh, R., Tyson, G., Wang, L., Crowcroft, J.: Of bots and humans (on twitter). In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017. pp. 349–354. ASONAM '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3110025.3110090, http://doi.acm.org/10.1145/3110025.3110090
7. Isbister, T., Kaati, L., Cohen, K.: Gender classification with data independent features in multiple languages. In: 2017 European Intelligence and Security Informatics Conference (EISIC). pp. 54–60. IEEE (2017)
8. Loper, E., Bird, S.: Nltk: The natural language toolkit. In: In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics (2002)
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
10. Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF. Springer (2019)
11. Rajaraman, A., Ullman, J.D.: Mining of massive datasets. Cambridge University Press (2011)

12. Rangel, F., Franco-Salvador, M., Rosso, P.: A low dimensionality representation for language variety identification. In: International Conference on Intelligent Text Processing and Computational Linguistics. pp. 156–169. Springer (2016)
13. Rangel, F., Rosso, P.: Overview of the 7th Author Profiling Task at PAN 2019: Bots and Gender Profiling. In: Cappellato, L., Ferro, N., Losada, D., Müller, H. (eds.) CLEF 2019 Labs and Workshops, Notebook Papers. CEUR-WS.org (Sep 2019)
14. Shannon, C.E.: A mathematical theory of communication. Bell system technical journal **27**(3), 379–423 (1948)
15. Varol, O., Ferrara, E., Davis, C.A., Menczer, F., Flammini, A.: Online human-bot interactions: Detection, estimation, and characterization. CoRR **abs/1703.03107** (2017), http://arxiv.org/abs/1703.03107