Language-Agnostic Twitter Bot Detection

Jürgen Knauth
Institute for Computer Science
University of Göttingen
jknauth@uni-goettingen.de

Abstract

In this paper we address the problem of detecting Twitter bots. We analyze a dataset of 8385 Twitter accounts and their tweets consisting of both humans and different kinds of bots. We use this data to train machine learning classifiers that distinguish between real and bot accounts. We identify features that are easy to extract while still providing good results. We analyze different feature groups based on account specific, tweet specific and behavioral specific features and measure their performance compared to other state of the art bot detection methods. For easy future portability of our work we focus on language-agnostic features. With AdaBoost, the best performing classifier, we achieve an accuracy of 0.988 and an AUC of 0.995. As the creation of good training data in machine learning is often difficult - especially in the domain of Twitter bot detection - we additionally analyze to what extent smaller amounts of training data lead to useful results by reviewing cross-validated learning curves. Our results indicate that using few but expressive features already has a good practical benefit for bot detection, especially if only a small amount of training data is available.

1 Introduction

While at the end of the 90s digital communication was still handled exclusively via well-defined Internet protocols, a completely new form of communication has established over the decades using web-based technology: Social media. These are web-based platforms that allow users to publish and exchange messages with each other after registration. These platforms are highly attractive: on the one hand, they establish a form of social community that enables people to enter into new relationships with each other or to maintain existing relationships. On the other hand, these platforms are designed in such a way that their users can easily find new content based on algorithmic recommendations.

Today, these social media platforms cover a wide variety of media. This work focuses on Twitter[1] - a service that enables the publication of short text comments.

As information and opinions are widely disseminated and exchanged via these platforms, the rise of social media as mass communication tools has increasingly attracted the interest of different actors who try to use social media as influencing factors. Businesses recognize the potential to advertise their products in this way at a very early stage, while political actors try to promote their views.

With the rise of such platforms to mass media and the desire to influence public perception and opinion, a comparatively new phenomenon has developed: The use of bots. In addition to normal human users, some social media platforms also feature computer programs that generate and provide content. This phenomenon has been found on Twitter as well as other social media platforms.

Some of these bots are indistinguishable from normal users on a superficial level. This is desired by their creators: Though there are legitimitate applications that justify the use of bots, bots are also used for malicious purposes. E.g. bots can be used for spamming to advertise products or for drawing attention to certain political statements.

As technical methods seem to make it possible to promote certain opinions, there exists at least a

[1] https://twitter.com/

Proceedings of Recent Advances in Natural Language Processing, pages 550–558,
Varna, Bulgaria, Sep 2–4, 2019.
https://doi.org/10.26615/978-954-452-056-4_065

theoretic danger that democratic processes might be in influenced by bots. Badawy et al., 2018 analyzes a set of Russian bots and how their information is picked up by American twitter users related to these bot accounts. Their research shows that information is introduced selectively into social media such as Twitter. Badawy et al., 2018 show that this information gets picked up and is spread - by users and other bots. The extent to which such influence can actually influence public opinion is not yet clear and requires further research. However, the very fact that such information is disseminated makes detecting Twitter bots an important issue to address.

2 Related Work

Detecting Twitter bots has been addressed as a research topic for quite some time. Wang, 2010 reconstructed a graph model based on crawled Twitter accounts and tweets exploiting follower and friend relationships. They use their data to implement a spam detection approach using classic Bayesian classication to distinguish between normal behavior from suspicious behavior on Twitter.

In 2012 a large scale classification study was conducted by Chu et al., 2012 to detect bots, humans and "cyborgs" in Twitter data. The term "cyborgs" is used here for accounts that are a symbiosis of humans and bots, making this a multinomial problem. Chu et al., 2012 used a dataset of 2000 manually classified Twitter accounts and achieved an accuracy of about 96% with a random forest approach.

Clark et al., 2015 uses a different approach. They not only crawl Twitter directly to extract tweets but additionally use data from a honeypot. They provided a setup with Twitter accounts being bots themselves. Twitter users following these accounts for no real reason are considered as being bots. Clark et al., 2015 focus on content of the tweets and use three different features to measure tweet similarity and hyperlinks. Based on this they propose a classifier to distingusish between "organic" and "robotic" texts achieving an accuracy of 90.32% of the "organic" and 95.2% of the "robotic" accounts. They additionally identify different behavior among these accounts indicating that there are several different classes of spammers and spam bots.

Cresci et al., 2015 and Cresci et al., 2017 manually analyzed and annotated over 8000 account data records and conducted an even more in depth analysis of the phenomenon of Twitter bots. Building a classifier they achieved an accuracy of 95% (Cresci et al., 2015). In their 2017 paper they discovered a new generation of social bots being more sophisticated compared to bots already known. They improved their performance by using advanced clustering algorithms and a large set of 126 features and were able to detect up to 97.6% of Twitter bots on one of their data sets.

Newer work uses sophisticated machine learning techniques. Cai et al., 2017 detect bots by modeling users and their behavior using deep learning by focusing on topic, latent sentiment and temporal aspects. Their CNN-LSTM model learns from user content as well as user behavior. With this approach they achieve an F1 score of 88.30% percent for their deep learning model outperforming reference models based on content or behavior alone.

Efthimion et al., 2018 use an approach with support vector machines. They make use of the Levenshtein distance to detect similar posts for their classification approach. This is noteworthy, as we make use of Levenshtein distance as well, but in contrast to Efthimion et al., 2018 use it to detect similarities not among tweet content but on account level. (See below.) Efthimion et al., 2018 use the same data as we use in our work and achieve and accuracy of 95.77% (which we outperform in our work).

Lundberg et al., 2018 provide two classifiers that are used to not only build an offline classifier, but a system that performs online analysis of tweets as they emerge from the Twitter community. They achieve an accuracy of about 98%.

3 Contribution and Outline

3.1 Research Questions

According to the importance of social media and Twitter bots outlined in the introduction, the following scientific questions arise regarding the recognition of Twitter bots:

- Is it possible to detect bots at account level only, without taking into account the content provided by these accounts?
- Does bot detection benefit from content analysis?
- Is it possible to reliably detect bots in a way that avoids language-specific features?

- Creating training data for bot detection is difficult. How large does such a data set have to be in order to achieve useful training results for machine learning? Can a small data set be sufficient?

4 Methodology

4.1 Considerations about Bots and Humans

Our approach to identifying Twitter bots is based on the assumption that bots are fundamentally different from humans in some aspects. Our consideration is that two categories should be distinguished here:

- Technical differences
- Purpose-related differences

4.1.1 Technical Differences

Due to the fact that bots are computer programs, they are not subject to certain human limitations. Computer programs can act instantly in contrast to humans who need time to reflect and are often occupied with other tasks of daily life and work. It can therefore be assumed that human behaviour is different from bot behaviour with regard to timing and the orientation of published content.

Our considerations are also based on the assumption that it is difficult for computer programs to imitate human behaviour. While it is possible to simulate human inadequacies, for example by delaying reactions, it would be difficult for bots to accurately mimic human behavior: This would require extensive statistical analysis of the behaviour of Twitter users. We can therefore assume that bots are created using simpler methods, such as random temporal behaviour, which only resemble human behaviour at first glance.

4.1.2 Purpose-Related Differences

Bots have clear objectives, for example spreading political messages or references to products. Bots bring specific content to attention, hashtags, URLs. So they have some kind of "agenda" which should be able to be exploited to some extent in general.

It should be noted here that the fact that we and other researchers are able to identify bots quite reliably clearly shows that these assumptions are not unfounded.

4.2 Dataset

Our work is based on the MIB dataset (Cresci et al., 2017), which contains 8375 annotated Twitter accounts.

- 3473 accounts - humans
- 991 accounts - political candidate retweeters
- 3457 accounts - paid apps spammers
- 464 accounts - amazon.com spammers
- Total number of accounts: 8375 accounts

This data set contains data records about the accounts themselves as well as tweets created.

4.3 Baseline

In the next sections we lay out our feature extraction and machine learning process. In order to compare our results with a baseline we reimplement one of the machine learning classification systems described in Kudugunta and Ferrara, 2018. This classifier is quite a high baseline as it performs very well and is - to our knowledge - the current state of the art.

4.4 Feature Extraction

For building a machine learning classifyer we require a matrix of feature values for training and - later on - classifying unseen test data. So in a first first step we extract a variety of features from the Cresci et al., 2017 datasets. These features are discussed in the next subsections.

4.4.1 Account Based Features

The first group of features is derived from account metadata. Our *simple user profile features* directly reflect values the Twitter API provides about users. We additionally derive features with some processing from the screen and user names.

Some of the features are self explanatory or explained by the Twitter API documentation. [2] Nevertheless some of these features require additional discussion.

- **Simple user profile features:** We hypothesize that metadata from the user profile provides valuable information about the user account. Some of this data is generated by

[2] https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/user-object.html

Twitter itself and sometimes difficult to control directly by users. This data contains characteristics about a user's account we can exploit for machine learning. These account features include:

  - *default_profile:* Has the user altered the profile?
  - *geo_enabled:* This feature reflects if users enable adding geographic information if they publish a tweet.
  - *protected:* When true, indicates that this user has chosen to protect their Tweets.
  - *is_verified:* This is some kind of quality marker provided by Twitter: Accounts that are run by people of public interest can be verified as being authentic by Twitter itself.
  - *friends_count:* The number of users this account is following.
  - *followers_count:* The number of followers this account has.
  - *favourites_count:* The number of tweets this user has liked.
  - *listed_count:* The number of public lists this account is a member of.
  - *statuses_count:* The number of tweets issued by this account.
  - *profile_use_background_image:* Has the user provided a background image?

- **User profile name features:** User and screen names are very much subject to a user's choice. Therefore we hypothesize that it provides valuable information that helps to distinguish bots from humans. These features include:

  - *screen_name_length:* The length of the screen name provided by a user.
  - *user_name_length:* The length of the account name provided by a user.
  - *screen_name_digits:* Number of digits in the screen name.
  - *user_name_unicode_group:* See below.
  - *screen_name_unicode_group:* See below.
  - *levenshtein_user_name_screen_name:* See below.

We use some features that are closely related to the screen and user names of accounts. In particular, we determine which of the 105 Unicode code groups an account uses in the screen and user names. This is reflected in the categorical features *user_name_unicode_group* and *screen_name_unicode_group* where we have one feature for every Unicode code group. The rationale behind this feature is that humans tend to be quite creative in their choice of names and sometimes tend to pick characters completely unrelated to the alphabet of their own language. By comparing occurrences of characters in various Unicode code groups we take this behaviour into account.

Furthermore we want to make use of possible differences between an account's screen name and user name. We model this by calculating the respective Levenshtein distance (Levenshtein, 1966). We observed that bot accounts tend to choose user names and screen names that are similar, while humans can be more creative in this respect.

4.4.2 Content Based Features

We derive additional features from the content a user provides.

For that purpose we tokenize the tweets. We tokenize by breaking the tweet text at a variety of spaces and punctuation such as commas, colons, exclamation marks, brackets and similar characters that are commonly used in sentences. This tokenization respects characters occuring in emojis as well. While tokenization in itself is not entirely language agnostic, these heuristics should nevertheless work for a fairly large set of (Latin script) languages.

Then we can extract features based on these tokens. Other features are directly derived from the metadata of tweets.

- **Behavioural features:** We hypothesize that the tweeting behaviour of bots and humans should exhibit differences. We model this behaviour by calculating statistical properties from the data such as minimum, maximum, average, mean, median, standard deviation, skewness, kurtosis, and others.

  - *time_between_retweets (distributional feature):* This set of features models time between retweeting activities of an account.
  - *time_between_tweets (distributional):* This set of features models time between tweeting activities.

- *tweet_rate (average tweet rate):* The average number of tweets per day

- **Core content features:** We hypothesize that some aspects of intention and emotions can be derived from a tweet's content. The following features honor this in a language independent way.
  - *emojis_classic (distributional feature):* See below.
  - *emojis_kaomji_faces (distributional):* See below.
  - *emojis_line_art (distributional):* See below.
  - *emojis_other (distributional):* See below.
  - *number_of_tokens (distributional):* This feature models the size of a tweet to some extent.
  - *number_of_hashtags (distributional):* The number of hashtag based references contained in tweets.
  - *n_of_tokens_wo_hashtags_urls_symbols (distributional):* A distribution based on the number of plaintext tokens in the tweets excluding hashtags, URLs and non-alphanumeric tokens.
  - *number_of_urls (distributional):* The number of URL based references contained in tweets.
  - *special_char_repeats_rate:* This feature detects sequences of question marks and exclamation marks. These are also often used for affirmation to give special weight to what has been expressed in a tweet. We want to model this aspect.

We assume that a precise sentiment analysis of the twitter text is not available for all languages tweets could be written in. Nevertheless, we want to extract and use emotional aspects from the content. In order to model some basic aspects of emotion we detect various different sets of emoticons here. The extraction is pattern based and derived from public emoticon collections as provided by Wikipedia. The four emoji features above are distributions derived on individual occurrences of emojis in single tweets.

4.5 Feature Groups

Based on these features we build sets of different features for use in different machine learning experiments.

For reasons of getting a more detailed understanding about the performance of various feature sets we group all behavioral features to a feature set named `tweet-behav`, all tweet content related features to a set named `tweet-cont` and all account related features to `account`. We use `account-lev` as a feature set that only includes the Levenshtein distance between user and screen names to test this feature specifically.

As we want to compare our work to Kudugunta and Ferrara, 2018 we reimplented all their account features mentioned in their paper. We later on refer to this feature group with `k_f_reimpl`.

Our own feature group `all` contains almost all features described above leaving out only four minor features as they don't seem to improve the quality of our classifier: Information about the default profile, location, the content of the "protected" field and information about user background image.

5 Results and Discussion

We generate feature matrices based on different sets of features as described in section *4.5*. This data is normalized by scaling each feature to the unit interval and then used to train and evaluate different machine learning models.

To evaluate the performance of our machine learning classifiers we separate our dataset into a training and validation set by an 80:20 ratio using (deterministic) random selection provided by the scikit-learn framework (Pedregosa et al., 2011).

We have experimented with classical methods such as Logistic Regression[3], Support Vector Machines[4], Random Forests[5] and Multi-layer Perceptrons[6]. The best results were achieved with *AdaBoost*[7] (Freund and Schapire, 1999). Since the training labels are not fully balanced, we follow Kudugunta and Ferrara, 2018 and resampled the training data using SMOTE-ENN (Lemaître et al., 2017).

By using AdaBoost with SMOTE-ENN we achieve the results displayed in table 1. Figure 1 contains some of the corresponding ROC curves.

The feature set **tweet-behav** contains all features related to the tweeting behavior of an account

[3] `sklearn.linear_model.LogisticRegression`
[4] `sklearn.svm.LinearSVC`
[5] `sklearn.ensemble.RandomForestClassifier`
[6] `sklearn.neural_network.MLPClassifier`
[7] `sklearn.ensemble.AdaBoostClassifier`

Feature-Set	P	R	F1	ACC	AUC_ROC
tweet-behav	0.9920	0.8888	0.9376	0.9314	0.9475
tweet-cont	0.6413	0.9701	0.7721	0.6685	0.9295
account	0.9907	0.9835	0.9871	0.9851	0.9929
account-lev	0.9481	0.9041	0.9838	0.9159	0.9604
all	**0.9958**	**0.9835**	**0.9896**	**0.9881**	**0.9959**
k_f_reimpl	0.9886	0.9804	0.9845	0.9821	0.9935
k_f-AdaBoost-SMOTEENN	1.00	1.00	1.00	0.9981	0.9981

Table 1: Classification performance of models using different feature sets. k_f_reimpl is our reimplementation of our reference classifier. Though we reimplemented all features we were not able to confirm their original results. These are provided in k_f-AdaBoost-SMOTEENN for reference.

Feature	ANOVA F-value
number_of_tokens_without_hashtags_urls_symbols_median	8255.4
number_of_tokens_median	7843.3
levenshtein_user_name_screen_name	7736.9
number_of_tokens_without_hashtags_urls_symbols_mean	7331.9
number_of_tokens_without_hashtags_urls_symbols_stdev	7157.1
number_of_tokens_stdev	7060.3
number_of_tokens_mean	7007.8
emojis_classic_skewness	5836.8
emojis_other_skewness	5836.8
number_of_tokens_min	4783.9
number_of_tokens_without_hashtags_urls_symbols_min	3236.3
number_of_tokens_without_hashtags_urls_symbols_entropy	2262.7
number_of_tokens_entropy	2257.4
number_of_tokens_without_hashtags_urls_symbols_max	2245.3
number_of_tokens_max	2237.1
number_of_hashtags_skewness	1970.1
special_char_repeats_rate	1829.5
emojis_classic_kurtosis	1558.2
emojis_other_kurtosis	1558.2
statuses_count	1171.2

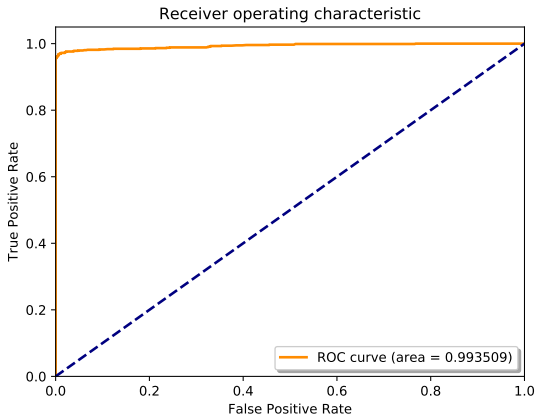
Table 2: The 20 most important features for the system "all", sorted by ANOVA F-value.

and **tweet-cont** contains all content features. Our experiments show that neither feature set is able to achieve sufficiently good results, as there is a large number of false positives.

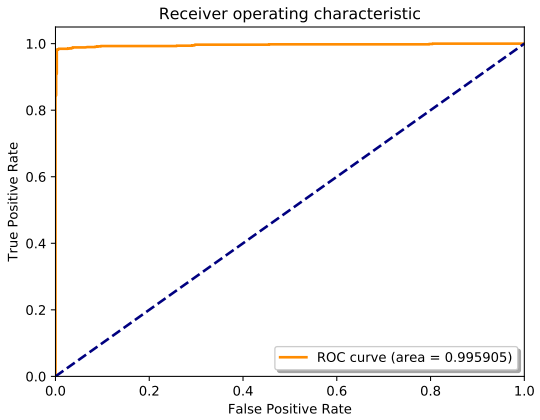
It turned out that using the Levenshtein distance as a feature provides a considerable benefit. It is the most informative feature in the **account** feature set, followed by *geo_enabled*, *statuses_count*, *user_name_length*, *screen_name_length* and features related to account metadata, such as *default_profile*, *favourites_count* and *profile_use_background_image*.

In order to get an idea of how well this feature performs we conducted an experiment using it as the only classifier. We found that by using only this features alone (the **account-lev** feature set) it is possible to achieve an accuracy of 0.8611.

In order to compare our system we reimplemented a classification system as described by Kudugunta and Ferrara, 2018, where AdaBoost with SMOTE-ENN is used to get excellent results. However, although we use the same dataset and the same features, we were not able to achieve the same results. We assume the reason for this is that the exact selection of their training data is unknown to us. For our own analysis we divide the gold standard data using deterministic random selection. Depending on the exact nature of this random selection, it is understandable that each trained system based on this selection will



(a) ROC curve for ”k_f_reimpl”



(b) ROC curve for ”all”

Figure 1: ROC curves

be slightly different and the evaluation results will therefore vary slightly.

With our feature set including features that address the tweet content as well as the the meta data, the unicode groups, the emojis in the content, the user and screen names as well as the the Levenshtein distance between both names we achieved a comparable accuracy of 0.9881 and ROC-AUC value of 0.9959. (For details see table 1.)

6 Further Experiments

In order to estimate the influence of the amount of training data on the quality of such a classifier, we conducted additional experiments and calculated learning curves for several of our systems.

To achieve representative results we use five-fold cross-validation here. Our data set contains the manually tagged data of the 8385 accounts. Therefore 1677 records will be used for validation and 6708 data records remain for training.

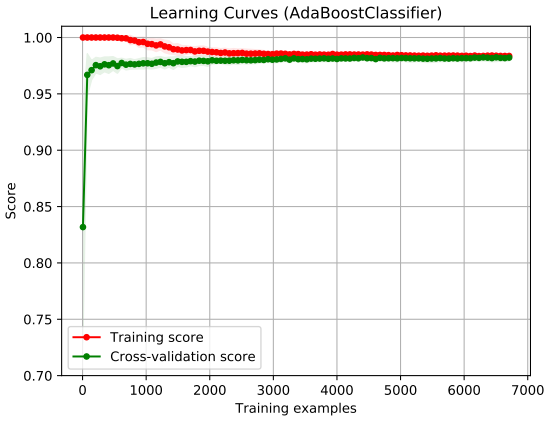
Figure 2 displays learning curves for some of our classifiers, all based on AdaBoost. Targeting one of our research questions understanding if smaller amount of training data could be sufficient for building classifiers we extracted upper left areas of some of our learning curves and present them in figure 2. These results indicate that even if data about only a very limited set of Twitter accounts would be available for training bot detection should be possible though - of course - it would not achieve the very best results. As each account in our data set in average provides 607 tweets this seems to be already a sufficient basis to learn from for practical applications.

7 Conclusion

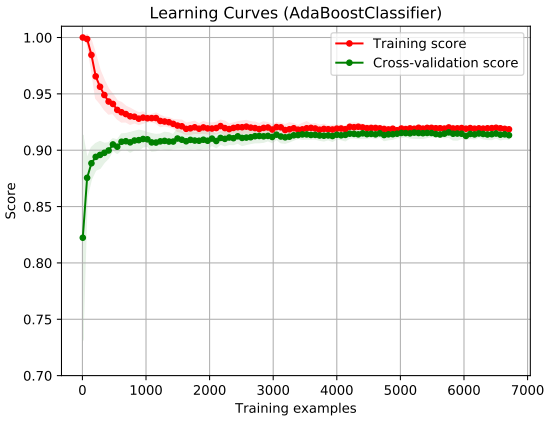
In this paper we addressed the problem of detecting Twitter bots. For easy reuse of our system we extracted various language-agnostic features including account specific features such as comparing screen name and user names as well as behavioural specific features modeling latent temporal aspects of tweeting. We have shown that with AdaBoost and SMOTE-ENN we can achieve a precision of up to 0.9969, an accuracy of 0.9881 and an AUC-ROC value of 0.9959. Additionally we calculated learning curves of several of our approaches in order to understand the impact a smaller amount of data might have on training of classifiers. We concluded that patterns existing in the data emerge quite soon during training. For practical approaches already a set of a view thousand data records will be sufficient for training. Our data set consisted of 8385 manually classified records. This is sufficient data for training classification systems of good quality.

8 Future Work

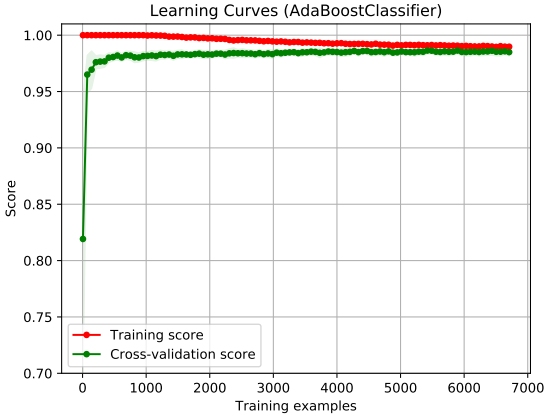
Naturally bot developers will pick up on results of researchers and improve their implementations of bots in order to perfect their disguise. Classifiers training on existing data might not work as well in future experiments on real data as bots could produce tweet content in such a way that is misleading for existing classifiers. Tweeting behavior could be modeled in such a way that it is less distinguishable from real users. Future research should focus on addressing these aspects closely.



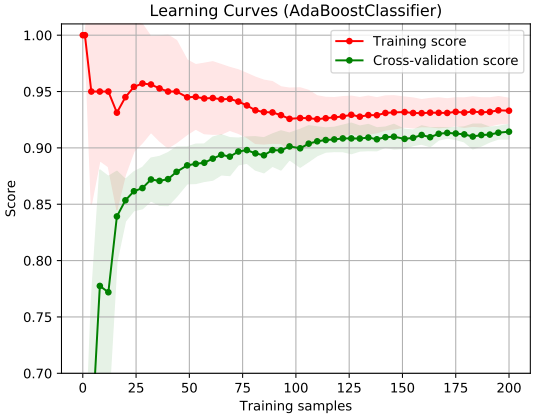
(a) Learning curve for ”k_f_reimpl”



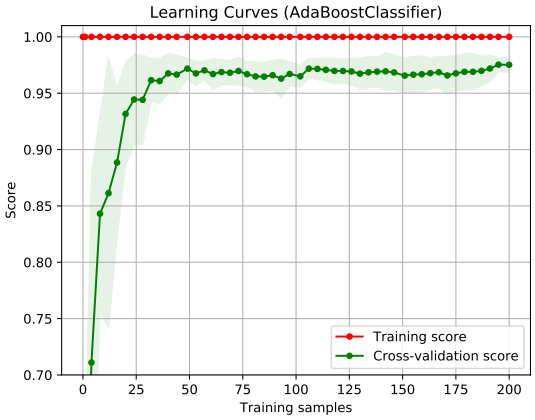
(b) Learning curve for ”account-lev”



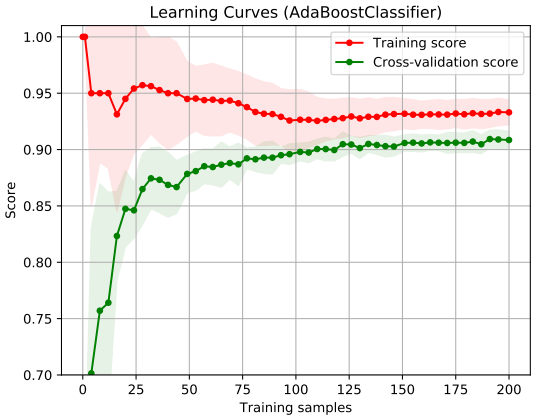
(c) Learning curve for ”all”



(d) Learning curve for ”tweet-cont”



(e) Learning curve for ”all”



(f) Learning curve for ”tweet-behav”

Figure 2: Learning curves for six of our classifiers

References

Adam Badawy, Emilio Ferrara, and Kristina Lerman. 2018. Analyzing the Digital Traces of Political Manipulation: The 2016 Russian Interference Twitter Campaign. *arXiv e-prints* page arXiv:1802.04291.

Chiyu Cai, Linjing Li, and Daniel Zeng. 2017. Detecting social bots by jointly modeling deep behavior and content information. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore*. pages 1995–1998.

Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. 2012. Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg? *IEEE Transactions on Dependable and Secure Computing* 9(6):811–824.

Eric M. Clark, Jake Ryland Williams, Chris A. Jones, Richard A. Galbraith, Christopher M. Danforth, and Peter Sheridan Dodds. 2015. Sifting Robotic from Organic Text: A Natural Language Approach for Detecting Automation on Twitter. *arXiv e-prints* page arXiv:1505.04342.

Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2015. Fame for sale: Efficient detection of fake Twitter followers. *Decision Support Systems* 80:56–71.

Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2017. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia*. pages 963–972.

Phillip George Efthimion, Scott Payne, and Nicholas Proferes. 2018. Supervised machine learning bot detection techniques to identify social twitter bots. *SMU Data Science Review* 1(2). Article 5.

Yoav Freund and Robert E. Schapire. 1999. A short introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, pages 1401–1406.

Sneha Kudugunta and Emilio Ferrara. 2018. Deep neural networks for bot detection. *Information Sciences* 467:312–322.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research* 18(17):1–5.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* 10:707.

Jonas Lundberg, Jonas Nordqvist, and Antonio Matosevic. 2018. On-the-fly Detection of Autogenerated Tweets. *arXiv e-prints* page arXiv:1802.01197.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

A.H. Wang. 2010. Don't follow me: Spam detection in Twitter. In *Proceedings of the 2010 International Conference on Security and Cryptography, Athens, Greece*. pages 1–10.