UMEÅ UNIVERSITY

CLASSIFYING TWITTER BOTS

A comparasion of methods for classifying whether tweets are written by humans or bots

Simon Västerbo

Bachelor Thesis, 15 hp/credits
Kandidatprogrammet i datavetenskap

2020

Abstract

The use of bots to influence public debate, spread disinformation and spam, creates a need for efficient methods for detecting the usage of bots. This study will compare different machine learning methods in the task of classifying if the author of a tweet is a bot or a human, using tweet level features. The study will look at how well the methods are able to generalize to unseen data. The methods included in the comparison are *Random forest*, *AdaBoost* and the *Contextual LSTM* model, to compare the models *Area under the receiver operating characteristic curve* and *Average precision* will be used. In the study five datasets with tweets from bots are used, and one with tweets from humans. Two tests have been used to evaluate the performance. In the first test all but one bot set is used during training, where the models are evaluated on the excluded set. The second test the models was trained on the separate datasets, and evaluated on the separate datasets. In the results from the first test, the difference in performance of the models where very low. The same was true for *Random forest* and *AdaBoost* in the second test. The *Contextual LSTM* model achieved low performance in some combinations of datasets, in the second test. The low difference in performance between the models in the first test, and between *Random forest* and *AdaBoost* in the second test, makes it hard to determine what model is best at the task. When taking the time required to train and test using the models into consideration, *Random forest* seem to be the most suitable for the task.

Acknowledgements

First I wish to give thanks to my supervisor Kai-Florian Richter. He provided valuable support during the whole process of this work, giving useful advice, constructive criticism and taking the time to answer my questions.

I would also like to give thanks the opponents of my thesis, Noradin Hagi Shire and Ken Sund. The constructive criticism and advice they provided was helpful.

Finally I wish to thank Stefan Warg, for taking the time to reading and giving feedback for several of the drafts for this thesis.

Contents

1 Introduction

The use of automated accounts, also called bots, on social media sites can have many reasons, some legitimate such as updates from a media provider. Other motivations for using automated accounts are more questionable, such as spreading advertisements or malware, or bots used to spread disinformation or political propaganda.

For example, during recent elections there have been indications of an increased use of bots to influence the outcome of the election. The model used by Ferquist et. al. classified around 6% of the accounts posting using political hashtags related to Swedish election during March to the end of May 2018 as bots.[4]Twitter bots have also been utilized as a tool to intimidate journalists in Yemen.[1] The use of bots to control the public discussion by silencing opposition or by spamming propaganda highlights the need for efficient ways to detect bot accounts and tweets. The sophistication of the bots used are also increasing as shown by Cresci et al., where some social bots use fake profile information, photos and location. The posting behavior of these bots is also increasingly similar to normal users.[3] Due to the reasons listed above, it is important to be able to identify bots, the typical way to do that is using machine learning.

1.1 Purpose and Research Questions

I will compare different methods in the task of classifying if tweets are posted by a human or a bot, based on the content and metadata of a tweet. I will try to answer how well the methods are able to generalize when classifying new types of bots.

RQ How does the models perform with *ROC AUC* and *Average Precision* as performance metrics when classifying tweets as coming from a bot or human?

In addition, I will discuss the implications that the results have on the suitability of the methods for different applications. I will also discuss the reason for the difference.

1.2 Models to include in the comparison

There exist many machine learning methods that could be included in the study, however due to the limited time available I have chosen three methods to include. In this section I will give a short motivation for including each of them, the methods will be explained in more detail in section 3.

Random forest

Random forest has achieved good performance when used to classify bots in several studies.[10][16]

Contextual LSTM

The *Contextual LSTM* model proposed by Kuduguta et al. preformed best among the models included in their comparison when classifying based on tweet information.[9]

AdaBoost

AdaBoost has achieved good performance when used to classify bots in several studies, and performed best in the non LSTM based models tested by Kuduguta et al.[9]

2 Related Work

Beskow et al. built a classifier to classify usernames based on the observation that a large number of bots used for intimidation of journalists in Yemen used strings of random alphanumeric characters as name. They estimated that using their method had a false positive rate of about 1%, and of the 100 classified accounts that they manually inspected five where suspended, eight gave no results (probably due to being canceled) and the rest showed bot like behavior.[1]

The *Botometer* bot classifier described by Yang et al. is based on *Random forest*.[17] *Random forest* was also the algorithm that performed best in the comparisons done by Rossi et al. and Novotny.[13][15]

Morstatter et al. proposed the *BoostOR* model based on AdaBoost that focused on optimizing recall. When compared to their different heuristics for classifying bots, SVM and AdaBoost, their model achieved a higher F_1 score.[12]

Wang et al. compared different models for classifying tweets as spam using features based on the content and metadata of a tweet. Of the compared models random forest performed best when looking at F_1 score. They also evaluated the importance of the included features when using random forest.[16]

Kudugunta and Ferrara investigated classifying if the poster was a bot based on separate tweets. They used a *Long short term memory* (LSTM) based approach that use the text of a tweet and the account metadata that is given with the tweet by twitters API to classify if the poster was a bot. They also tested a number of methods for account level classification, out of the box and where *Synthetic Minority Over-sampling Technique* (SMOTE) has been used to balance the data in combination with data enhancement using *Edited Nearest Neighbors* (ENN) or *Tomek links*. Used out of the box Random forest performed best, in both cases where the data had been enhanced AdaBoost performed best. In the task of tweet level classification their LSTM based model outperformed the other models followed by AdaBoost and Random forest. They also showed that including metadata for the tweet gave an improvement of accuracy compared to only looking at the text.[9]

Luo et al. proposed DeepBot that used a Bi-directional LSTM based model with tweets embedded into vectors with the GloVE representation as input. They trained their model with 144000 tweets posted by bots and 144000 by humans, using binary cross entropy as loss function and the adam optimizer. On their testing set of 62000 tweets by bots and 62000 by humans they achieved a accuracy of 0.80 and AUC of 0.87. [11]

3 Theoretical Background

In this section I will start with a description of two metrics that can be used when comparing models in the task of classification. Then I will give a description of three different machine learning algorithms that may be used to learn classification models.

3.1 Metrics

In evaluating classification tasks, two commonly used metrics are *Average precision* (AP) and the *Area under the receiver operating characteristic curve* (ROC AUC). This section will give a description of these two metrics.

Average Precision

Average precision (AP) is a way to give a single value summary of a *precision recall curve*. A precision recall curve is a plot of the *precision* and *recall* as the decision threshold is changed. Precision is defined as $P = T_p/(T_p + F_p)$ where T_p is the number of true positives and F_p the number of false positives, that is the fraction of predicted positive values that are correctly classified. The recall (or *true positive rate*) is defined as $R = T_p/(T_p + F_n)$ where T_p is the number of true positives and F_n the number of false negatives, that is the fraction of the positive values that was correctly classified.

If a vector of elements $\boldsymbol{x} = (x_1, ..., x_n)$ is ordered so that $p(x_i) \geq p(x_j)$ if $i < j$ where $p(x)$ is the predicted probability[1] of x belonging to the positive class, the average precision AP can be calculated as $AP = \sum_n (R_n - R_{n-1})P_n$ where R_i is the recall and P_i the precision when the threshold is set so that the first i elements are classified as positive.

ROC AUC

A *Receiver operating characteristic curve* (ROC) is a plot of the *true positive rate* and the *false positive rate* as the threshold is changed. The *false positive rate* (FPR) is defined as $FPR = (F_p/(F_p + T_n)$, where F_p is the number false positives and T_n the number of true negatives, that is the fraction of negative values that was incorrectly classified as positive. The *true positive rate* is the same as recall and the definition is shown above. The *ROC AUC* score is the area under the ROC curve.

3.2 Machine learning algorithms

Three algorithms that may be used in the task of classification is: *Contextual LSTM*, *Random forest* and *AdaBoost*. All three are supervised machine learning methods, meaning that they try to learn a mapping between inputs to outputs based on a set of pairs of input and output examples, or more formally, given a input $\mathcal{D} = \{(x_1, y_1)...(x_n, y_n)\}$ where each $y_i = f(x_i)$ the goal in supervised learning is to output $\hat{y_i} = \hat{f}(x_i)$ where $\hat{f}$ is an approximation of f. This section will give a description of the above mentioned algorithms.

[1] or some other scoring function

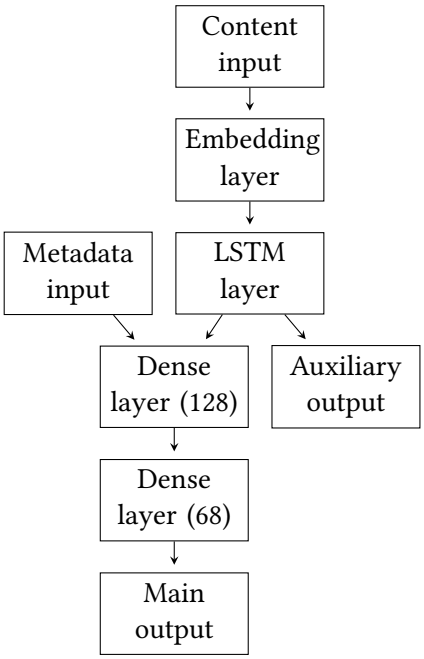

Figure 1: Outline of the Contextual LSTM network

Contextual LSTM

This section will describe the *Contextual LSTM* model. The model is based on the *Contextual LSTM* model proposed by Kudugunta et al.[9]

Network architecture The network has two input layers, one for content and one for metadata.

The content input takes integers and the embedding later maps them to word vectors. The output from the embedding is then given as input to a lstm layer. The output of the lstm layer and the metadata input is then used as input into a fully connected network with tree layers.

LSTM layer *Long Short Term Memory* (LSTM) networks is a type of recurrent neural network first proposed by Hochreiter et al. 1997.[7] The LSTM network consists of 32 memory cells.

Dense layers The last layers is a fully connected layer with 128 nodes followed by a fully connected layer with 64 nodes both using the *Rectified Linear Unit* or *ReLu* as activation function, ending with a single neuron output activated by the *sigmoid* function.

$$relu(x) = \max(x, 0)$$

$$sigmoid(x) = \frac{1}{1 + exp(-x)}$$

Binary crossentropy The *Binary crossentropy* loss function is used to estimate how bad the performance of a model is. The function gives a higher value for models with bad predictions.

$$H(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

where $\boldsymbol{y} = (y_1, ..., y_n)$, $y_i \in \{0, 1\}$ is the true values and $\hat{y}_i$ is the predicted probability of $y_i = 1$.

Random forest

Random forest is a algorithm based on building a forest of decision trees introduced by Breiman 2001[2]. The algorithm uses a technique called *bagging* that works by selecting a random subset of examples from the training set, with replacement, for every tree in the forest. Random forest also use a random subset of features when deciding the best split for the nodes in the trees. In the paper by Breiman the class is given by voting where the class with largest number of votes is selected, an alternative approach is to use the average predicted probability among the trees.

AdaBoost

AdaBoost is based on iterating over multiple weak learners on the data, where each example is weighted. For each iteration the weights of the missclassified examples are increased. The weak learners used in the experiments are decision trees with a depth of 1. The *SAMME.R* algorithm was proposed by Zhu et al. 2006, *SAMME.R* is a algorithm based on the original AdaBoost algorithm by Freund and Schapire.[18][5]

SAMME.R

It is expected that it is possible to calculate probability estimates for the classes $\{1, ..., K\}$ from the weak learners T.

1. Let $w_i = 1/n$, $i = 1, 2, ..., n$, where n is the number of examples in the training data.
2. For $m = 1$ to M, where M is the number of weak learners:
   (a) Fit a classifier $T^m(\mathbf{x})$ using the training data and weights w_i.
   (b) Let $p_k^m(\mathbf{x}) = P_w(c = k|\mathbf{x})$, $k = 1, ..., K$ be the weighted probability estimates for the classes.
   (c) Let
   $$h_k^m(\mathbf{x}) = (K - 1) \left(\log p_k^m(\mathbf{x}) - \frac{1}{K} \sum_{k'} log p_{k'}^m(\mathbf{x}) \right), \ k = 1, ...K$$
   (d) Let
   $$w_i = w_i \exp \left(-\frac{K - 1}{K} \mathbf{y_i^T} \log \mathbf{p^m}(\mathbf{x_i}) \right), \ i = 1, ...n$$
   (e) Re normalize w_i
3. Output
   $$C(\mathbf{x}) = \arg\max_k \sum_{M}^{m=1} h_k^m(\mathbf{x})$$

4 Method

In this section I will start with describing the data used in the experiments, and how I have prepared it. After that a description of the experiments will follow.

4.1 Datasets used

The data used is from the *cresci-2017* data set available at [8], the data is described by Cresci et al. 2017.[3] The data set is a collection of several data sets containing tweets and user data for the posters of the tweets. A summary of the datasets is shown in table 1.

Table 1 Data sets used.

Name	Tweets	Type	Age	Origin
social-spambots-1	1,610,034	bot	2012	Bots used to promote candidate in election for Mayor of Rome.
social-spambots-2	428,542	bot	2014	Accounts used to advertise the *Talnts* app.
social-spambots-3	1,418,557	bot	2011	Accounts used to advertise the products on *amazon.com*.
traditional-spambots-1	145,094	bot	2009	Account used to post links to malicious content.
fake-followers	196,027	bot	2012	Fake followers bought by the the researchers.
genuine-accounts	2,839,362	human	2011	Accounts that answered a question in a natural language sent to them on twitter.

4.2 Data preparation

Before using the data to train and test the models a number of modifications is done to make the data more suitable for the task.

Text preparation

To make the text in the content of the tweets more suitable for the *Contextual LSTM* model a number of modifications is done. The modifications are done before using the data to train and test the models, the reason for that is to avoid repeating the time consuming task for every test. The steps done are as follows:

1. Substitute words in the text based on the rules used in the ruby script used by Pennington et al. for preprocessing the twitter data used for training the word vectors, some examples are listed in table 2. The reason for the substitutions are to make the text conform to the texts used to train the pretrained *GloVe* vectors [6]. For example: If one where to look up the word vector for `:D` in the pretrained *GloVe* embedding, no word vector would be found as all instances of `:D` was replaced with `<smile>` during the training of the word vectors.

2. Replace all uppercase letters with lowercase and split the text on white space characters into a list of strings.

3. For each string in the resulting list: Look up the index of the corresponding word vector and replace the string with that index, or if not found with the index of the unknown token.

Table 2 Table with examples of substitutions done during the text preparation. *eyes*={8:=;}, *noses*={'`-}

Original word	**Replacement**
URL address	<url>
Word starting with @	<user>
Combination of one element in *eyes* and one of)dD possibly separated some element in *noses*	<smile>
One element in *eyes* followed p possibly separated some element in *noses*	<lolface>
One element in *eyes* followed by (or preceded by) possibly separated by element in *noses*	<sadface>
One element in *eyes* followed by one of \/|l possibly separated by element in *noses*	<neutralface>
<3	<heart>
#word	<hashtag> word
WORD	word <allcaps>

4.3 Data sampling

To train and evaluate the models, the features used in the tests can be divided into tweet specific and user specific features, where the user specific features will be constant for all the tweets posted by the same user.

The method for selecting training and testing data I used for each dataset:

For each dataset d listed in table 1:

1. Let $\mathcal{D}^d$ be the tweets in dataset d.
2. Let U be the set of user ids that have posted tweets in $\mathcal{D}^d$.
3. Let U_s be a randomly selected set of user ids from U, so that
$$\frac{\sum_{i \in U_s} c_i}{\sum_{j \in U} c_j} \approx 0.2$$
where c_i is the number of tweets in D^d posted by the user with user id i.
4. Let D^d_{test} be the tweets in $\mathcal{D}^d$ posted by user ids in U_s and $\mathcal{D}^d_{train}$ the tweets in $\mathcal{D}^d$ posted by user ids not in U_s.
5. If $|\mathcal{D}^d_{test}| > 100000$ select 100000 random tweets from $\mathcal{D}^d_{test}$ as the test set for that data set, else select all tweets in $\mathcal{D}_{test}$ as the test set.
6. $|\mathcal{D}^d_{train}| > 400000$ select 400000 random tweets from $\mathcal{D}^d_{train}$ and let $\mathcal{D}^d_{train}$ be the set of selected tweets.
7. If the data set is composed of tweets posted by bots:
   (a) $|\mathcal{D}^d_{train}| > 100000$ select 100000 random tweets from $\mathcal{D}^d_{train}$ and let $\mathcal{D}^d_{train2}$ be the selected tweets, else let $\mathcal{D}^d_{train2} = \mathcal{D}^d_{train}$

The reason for the upper limit of tweets in the training and testing sets is to decrease the time required to run the training and testing. The result of the limits is that the maximum number of tweets during training is 800000 (400000 from bots and 400000 from users) for both tests, and during testing 200000 (100000 from bots and 100000 from users). The reason for step 7 is that test 1 described in section 4.4 use tweets from four of the test sets during training, so to achieve the same total limit, the number of tweets from each set has to be decreased. The tweets selected for training and testing remain constant for the training and testing of the different models. To process and sample the data I have used the *pandas* python 3 library.

4.4 Tests

To try to answer the research question I have performed two tests. All combination of models described in section 3.2 and datasets described in section 4.1 containing tweets from bots have been tested using both tests. In both tests I will use the metrics described in section 3.1 to evaluate the models, namely *ROC AUC* and *AP*.

In the first test, the models are trained using all bot datasets except for one, and then evaluated using data from the unseen dataset. It is a more realistic scenario for the following reasons: As there exists several public datasets of tweets labeled as being posted by ether a bot or a human, it is reasonable to include data from several sources. The time required to collect and label data, combined with the appearance of new bots used to post on twitter, makes it unlikely that the data used during training is fully representative of the data that on which the model is used to classify.

The second test is done by training the models using separate bot datasets and evaluate on separate bot datasets. The goal with that test is to give some insight how the similarities and differences between the datasets affect the performance of the models. The hope is that it also can give some insight about the results from the first test.

Test 1

In the first test I train the different models using a combination of all but one datasets. The trained model is then scored based on its performance when used on the test set of the excluded dataset combined with the test set of the user set.

The test can be described as the following steps:

1. Let B be the set of datasets consisting of tweets posted by bots.
2. For each dataset $d, d \in B$:
   (a) For each compared model M:
      i. Let $B^t = \{b \mid b \in B \text{ and } b \neq d\}$
      ii. Let $\mathbb{X} = \cup_{i \in B^t} \mathcal{D}^i_{train2} \cup \mathcal{D}^u_{train}$ where u is the data set consisting of tweets from real users.
      iii. Let $p(y = 1|\mathbf{x}, \mathcal{D}, \mathcal{M})$ be the estimated probability of the tweet corresponding to the feature vector $\mathbf{x}$ is posted by a bot, given the model M trained on the training data $\mathcal{D}$.
      iv. Let $\hat{\mathbf{y}} = \{p(y = 1|\mathbf{x}, \mathbb{X}, \mathcal{M}) \mid x \in \mathcal{D}^b_{test} \cup \mathcal{D}^u_{test}\}$.
      v. Calculate the metrics described in section 3.1 based on the predicted probability's $\hat{\mathbf{y}}$ and the true labels $\mathbf{y}$.

Test 2

In the second test the models are trained using the data from one of the bot datasets at a time, combined with the training data from genuine users. The models are then evaluated using each of the test sets from the bots combined with the test set from real users.

The test can be described as the following steps:

1. Let B be the set of datasets consisting of tweets posted by bots
2. For each data set $d, d \in B$:
   (a) For each compared model M:
      i. Let $\mathbb{X} = \mathcal{D}^{d}_{train} \cup \mathcal{D}^{u}_{train}$ where u is the data set consisting of tweets from real users.
      ii. Let $p(y = 1|\mathbf{x}, \mathcal{D}, \mathcal{M})$ be the estimated probability of the tweet corresponding to the feature vector $\mathbf{x}$ is posted by a bot, given the model M trained on the training data $\mathcal{D}$.
      iii. For each d^t where $d^t \in B$:
         A. Let $\hat{\mathbf{y}} = \{p(y = 1|\mathbf{x}, \mathbb{X}, \mathcal{M}) \; | x \in \mathcal{D}^{d^t}_{test} \cup \mathcal{D}^{u}_{test}\}$.
         B. Calculate the metrics described in section 3.1 based on the predicted probability's $\hat{\mathbf{y}}$ and the true labels $\mathbf{y}$.

Training and implementation

The *Contextual LSTM* has been implemented and trained using the *keras* API included in the *tensorflow 2.2* library. The models is trained by minimizing the *Binary crossentropy* using the *Adam* optimizer. *30* is used as an upper limit for the number of epochs, if no improvement in the loss when validating is seen for three epochs in a row the training is stopped and the best performing weights are saved. The model is trained with batches of 64 tweets at the time.

The input to the embedding layer is the content of the tweets transformed into arrays of integers using the process described in section 4.2, the embedding in my tests use GloVe word vectors pre-trained on tweets made available by Pennington et al., GloVe is described in their paper.[14] In the tests pretrained word vectors is of lengths 25, 100 and 200 is used.

The experiments for *AdaBoost* use the *AdaBoostClassifier* class in the scikit-learn python library, and using the *SAMME.R* algorithm.

For *Random forest* the *RandomForestClassifier* in the scikit-learn library is used. The *RandomForestClassifier* predicts the class using the average predicted probability among the trees in the forest.

5 Results and Analysis

In this section I will present and give an analysis of the results of the two tests. First the results of the tests is discussed separatly, followed by a summary.

5.1 Test 1

The results from test 1 is presented in table 3, each column in the table represents the results when excluding the corresponding dataset from the training and testing on that dataset. The rows in the table represent the different models in the comparison.

That the difference in score is very low in general is worth pointing out. So while I describe some of the differences below, I think it is hard to draw any strong conclusion from the results.

As shown in the table there is no model that performed best in all tests. When comparing *ROC AUC* the *Random forest* based models performed best for the *traditional-spambots-1*, *social-spambots-1* and *social-spambots-2* datasets. For the *social-spambots-3* dataset the *Contextual-LSTM* model using word vectors of size 200 achived the highest score, and for the *fake-followers* dataset the *AdaBoost* model performed best.

Table 3 Results when training on all data sets except for one and evaluating on the excluded data set. , FF=fake-followers, SB1=social-spambots-1, SB2=social-spambots-2, SB3=social-spambots-3, TB=traditional-spambots-1. cl-25=contextual-lstm-25, cl-100=contextual-lstm-100, cl-200=contextual-lstm-200, ada=adaboost, rf=random-forest

	FF		SB1		SB2		SB3		TB	
	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap
ada	**0.935**	0.854	0.970	**0.981**	0.800	0.851	0.986	**0.990**	0.982	0.966
rf	0.932	**0.864**	**0.975**	**0.981**	**0.870**	0.887	0.978	0.966	**0.993**	**0.982**
cl-25	0.813	0.677	0.963	0.975	0.842	0.874	0.969	0.973	0.965	0.941
cl-100	0.854	0.748	0.969	0.979	0.846	0.870	0.959	0.946	0.958	0.936
cl-200	0.906	0.818	0.966	0.977	0.864	**0.896**	**0.989**	0.983	0.962	0.928

When comparing the *AP Random forest* performed best for the *traditional-spambots-1* and *fake-followers* datasets, and shared the top preformance with *AdaBoost* for *social-spambots-1*. *AdaBoost* performed best for *social-spambots-3*. When *social-spambots-2* was the data set to exclude and test on the *Contextual-LSTM* model using word vectors of size 200 achieved the highest score.

The random forest model achieved the highest score in 3/5 tests for both *ROC AUC* and *AP*. When looking at the results for the different datasets the score for *fake-followers* and *social-spambots-2* was lower than the rest for all models.

5.2 Test 2

The results for the second test is presented in table 4, 5, 6, 7 and 8. Due to the number of tables, they are placed in appendix A and this section will describe some general details of the result.

The model that achieved the highest score in most comparisons in *ROC AUC* was *Random forest*, with highest score in 14/25 tests. For *AP* the model that achieved the largest number of highest score was *AdaBoost* with 11/25.

For the *AdaBoost* and *Random forest* models the difference in score where quite small for most of the tests. The *Contextual LSTM* models however, achieved low scores on some combinations of training and testing datasets, in some cases not far from what is to expected from random guessing. The models that only use metadata achieved a comparatively high score in the same combinations, which may be caused by conflicting patters in the text of the tweets in the datasets.

When looking at the results for different datasets the *AdaBoost* and *Random forest* models achieved high scores for the *social-spambots-3*, *traditional-spambots-1* and *social-spambots-1*, for all training sets used. The scores when evaluating on *social-spambots-2* was in general lower than for the other datasets when using *AdaBoost* and *Random forest*.

5.3 Summary

In the first test the score between the models was very close, as for the *AdaBoost* and *Random forest* in the second test. In both tests the scores for *AdaBoost* and *Random forest* was in general lower when evaluating on *social-spambots-2*.

6 Discussion

In this section I will first discuss the results and their practical implications, then I will compare the results with the study done by Kudugunta and Ferrara.[9]

6.1 Results

As mentioned in section 4.4, the first test seems to be the more realistic scenario. While *Random forest* performed best in test 1, the small difference in the scores makes it hard to draw any real conclusion from that. Test 1 show that all of the compared models are able to achieve good performance when classifying tweets as coming from a bot or human.

In the second tests, the results for the *Contextual LSTM* model was more varied among the datasets. As the *Contextual LSTM* model also use the text of the tweets, it is possible that the differences in the textual content was the cause of the low performance in some of the combinations of test and training sets. That the *Contextual LSTM* showed consistently good performance in test 1 indicate that the usage of data from multiple sources can help to remedy the problem.

Even if the problems that the *Contextual LSTM* has when generalizing between some combinations of datasets can be mitigated by using a combination of datasets in the training data, and test 2 seems to be less realistic than test 1, the results from test 2 is still worth taking into consideration when choosing what model to use. When taking that into account, combined with the results in test 1 where *Contextual LSTM* and the models using only metadata are quite similar, the models using only metadata even performed slightly better, the results indicate that when using both tweet and account level features the two models using only metadata is preferable compared to the *Contextual LSTM*.

When selecting what model to use for a specific task, there are several factors apart from the classifying performance to consider. Two such factors is time required for testing and time required for predicting the class of previously unseen input. During both the training and evaluating when running the tests, the fastest model was the *Random forest*, and *Contextual LSTM* was slowest. If that is taken into account when selecting a model based on the results, *Random forest* seem to be the preferable choice.

6.2 Comparison with previous work

Kudugunta and Ferrara compared several methods, including *Contextual LSTM*, *Random forest* and *AdaBoost*, on the task of classifying if the author of a tweet is a human or a bot, based on only tweet specific features.[9] In their study the *Contextual LSTM* based models outperformed the other models, achieving a *ROC AUC* of 0.964 using word vectors of size 200, compared to 0.9065 for the best non *Contextual LSTM* based model, and 0.7765 for the best non *Contextual LSTM* based model where no data enhancing had been made. They also compared *Random forest* and *AdaBoost*, among other models, when using account level features, where *Random forest* achieved an ROC AUC of 0.9845 and *AdaBoost* 0.9823. In their study a combination of the *genuine-accounts*, *social-spambots-1*, *social-spambots-2* and *social-spambots-3* datasets where used.

In their test using only tweet specific features, 6 metadata features was used, compared to 16 used in this study. An advantage of only using tweet specific features is that a twitter account may be used by both a human and bot, and if user level features is used in the decision making, the ability to differentiate between the tweets posted by bots and humans from such hybrid

accounts may decrease. Their results showed that their *Contextual LSTM*, using the content of the tweet combined with metadata can achieve an improved performance compared to both *AdaBoost* and *Random forest* when only tweet specific features are included. This study indicate that the improvement does not hold when also including account level features, however there may also be differences in how the training of the models was preformed that affected the results.

In their tests performed on account level features, *AdaBoost* and *RandomForest* achieved a higher *ROCAUC* than most of the test I performed in test 1, a reason for that may be that I evaluated the performance using the test set from a holdout dataset, where the training set from the holdout dataset was not included in training.

6.3 Limitations

The datasets used in the study is quite outdated, where the tweets included is collected between 2009 and 2014. It seems likely that the bots in use in 2020 differ from the bots used 6-11 years earlier, so the results of this study may be outdated.

No statistical analysis is done on the results, for example in test 1 the results was very similar between the models, so if some statistical test had been included to test for statistical significant differences the results would have been more informative.

6.4 Conclusions

This study has compared *Contextual LSTM*, *Random forest* and *AdaBoost*, in the task of classifying tweets as coming from bots or humans. To compare the models *Area under the receiver operating characteristic curve (ROC AUC)* and *Average precision (AP)* have been used.

To compare the models two tests have been used, one where the models are trained using four of five bot datasets, and evaluated on the excluded. The models have also been evaluated using the separate bot datasets after being trained using separate bot datasets. The features used is a combination of account and tweet specific features, where the *Contextual LSTM* also use the text.

In the first test the differences in results where very low among the different models, while in the second test the *Contextual LSTM* model showed very low performance on some combinations of datasets. The scores for *Random forest* and *AdaBoost* was very close in the second test.

The low difference in the scores of the first test makes it hard to draw any real conclusion about how the models compare in that task. The same is true for *Random forest* and *AdaBoost* in the second test. When looking at both test, one of *Random forest* and *AdaBoost* seem to be the best choice. If also taking the time required to train and test the models, *Random forest* seem to be the preferable choice.

6.5 Future Work

The collection of labeled datasets of current tweets from bots is valuable to enable research in the subject of investigating modern bots. For example looking into if the results in this study are relevant for current bots.

Further research into classifying based on tweet specific features would be interesting, due to the possibility of posting by accounts being semi automated.

References

[1] David M. Beskow and Kathleen M. Carley. "Its all in a name: detecting and labeling bots by their name". en. In: *Computational and Mathematical Organization Theory* 25.1 (Mar. 2019), pp. 24–35. ISSN: 1572-9346. DOI: 10.1007/s10588-018-09290-1. URL: https://doi.org/10.1007/s10588-018-09290-1 (visited on 04/15/2020).

[2] Leo Breiman. "Random Forests". en. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324 (visited on 05/20/2020).

[3] Stefano Cresci et al. "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 - WWW '17 Companion* (2017). arXiv: 1701.03017, pp. 963–972. DOI: 10.1145/3041021.3055135. URL: http://arxiv.org/abs/1701.03017 (visited on 03/30/2020).

[4] Johan Fernquist, Lisa Kaati, and Ralph Schroeder. "Political Bots and the Swedish General Election". In: *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, Nov. 2018. DOI: 10.1109/isi.2018.8587347.

[5] Yoav Freund and Robert E Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". en. In: *Journal of Computer and System Sciences* 55.1 (Aug. 1997), pp. 119–139. ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1504. URL: http://www.sciencedirect.com/science/article/pii/S002200009791504X (visited on 05/20/2020).

[6] *GloVe: Global Vectors for Word Representation*. URL: https://nlp.stanford.edu/projects/glove/.

[7] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[8] *Indiana University Bot Repository*. URL: https://botometer.iuni.iu.edu/bot-repository/datasets.html.

[9] Sneha Kudugunta and Emilio Ferrara. "Deep neural networks for bot detection". en. In: *Information Sciences* 467 (Oct. 2018), pp. 312–322. ISSN: 0020-0255. DOI: 10.1016/j.ins.2018.08.019. URL: http://www.sciencedirect.com/science/article/pii/S0020025518306248 (visited on 04/15/2020).

[10] Octavio Loyola-González et al. "Contrast Pattern-Based Classification for Bot Detection on Twitter". In: *IEEE Access* 7 (2019), pp. 45800–45817. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2904220.

[11] Linhao Luo et al. "Deepbot: A Deep Neural Network based approach for Detecting Twitter Bots". In: *IOP Conference Series: Materials Science and Engineering* 719 (Jan. 2020), p. 012063. ISSN: 1757-899X. DOI: 10.1088/1757-899X/719/1/012063. URL: https://iopscience.iop.org/article/10.1088/1757-899X/719/1/012063 (visited on 04/15/2020).

[12] Fred Morstatter et al. "A new approach to bot detection: Striking the balance between precision and recall". In: *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. Aug. 2016, pp. 533–540. DOI: 10.1109/ASONAM.2016.7752287.

[13] Jan Novotny. "Twitter bot Detection & Categorization - A comparative study of machine learning methods". In: 2019.

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: (2014), pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[15] Sippo Rossi et al. "Detecting Political Bots on Twitter during the 2019 Finnish Parliamentary Election". eng. In: Jan. 2020. ISBN: 9780998133133. DOI: 10.24251/HICSS.2020.298. URL: http://scholarspace.manoa.hawaii.edu/handle/10125/64040 (visited on 03/30/2020).

[16] Bo Wang et al. "Making the Most of Tweet-Inherent Features for Social Spam Detection on Twitter". In: *arXiv:1503.07405 [cs]* (Mar. 2015). arXiv: 1503.07405. URL: http://arxiv.org/abs/1503.07405 (visited on 04/16/2020).

[17] Kai-Cheng Yang et al. "Arming the public with artificial intelligence to counter social bots". In: *Human Behavior and Emerging Technologies* 1.1 (Jan. 2019), pp. 48–61. DOI: 10.1002/hbe2.115.

[18] Ji Zhu et al. "Multi-class AdaBoost". In: *Statistics and its interface* 2 (Feb. 2006). DOI: 10.4310/SII.2009.v2.n3.a8.

A Tables

Table 4 Results when training on separate data sets and evaluating on SB3, FF=fake-followers, SB1=social-spambots-1, SB2=social-spambots-2, SB3=social-spambots-3, TB=traditional-spambots-1. cl-25=contextual-lstm-25, cl-100=contextual-lstm-100, cl-200=contextual-lstm-200, ada=adaboost, rf=random-forest

	FF		SB1		SB2		SB3		TB	
	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap
ada	0.931	0.898	**0.982**	**0.986**	**0.944**	**0.946**	0.993	0.994	0.961	0.960
rf	**0.969**	**0.966**	0.976	0.972	0.806	0.771	0.996	0.997	**0.985**	**0.984**
cl-25	0.497	0.518	0.898	0.870	0.505	0.467	0.998	0.998	0.960	0.960
cl-100	0.558	0.552	0.652	0.550	0.758	0.663	0.998	0.998	0.978	0.979
cl-200	0.525	0.523	0.634	0.565	0.899	0.861	**0.999**	**0.999**	0.972	0.969

Table 5 Results when training on separate data sets and evaluating on TB, FF=fake-followers, SB1=social-spambots-1, SB2=social-spambots-2, SB3=social-spambots-3, TB=traditional-spambots-1. cl-25=contextual-lstm-25, cl-100=contextual-lstm-100, cl-200=contextual-lstm-200, ada=adaboost, rf=random-forest

	FF		SB1		SB2		SB3		TB	
	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap
ada	0.900	0.879	**1.000**	**0.999**	**0.952**	**0.901**	0.991	0.978	**0.999**	**0.998**
rf	**0.990**	**0.973**	0.992	0.974	0.933	0.827	**0.996**	**0.986**	0.999	0.994
cl-25	0.922	0.826	0.904	0.825	0.487	0.247	0.792	0.675	0.987	0.953
cl-100	0.923	0.839	0.816	0.742	0.562	0.283	0.973	0.913	0.991	0.969
cl-200	0.750	0.575	0.941	0.870	0.557	0.239	0.975	0.929	0.976	0.940

Table 6 Results when training on separate data sets and evaluating on SB1, FF=fake-followers, SB1=social-spambots-1, SB2=social-spambots-2, SB3=social-spambots-3, TB=traditional-spambots-1. cl-25=contextual-lstm-25, cl-100=contextual-lstm-100, cl-200=contextual-lstm-200, ada=adaboost, rf=random-forest

	FF		SB1		SB2		SB3		TB	
	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap
ada	0.947	0.964	0.957	0.973	0.938	**0.953**	0.967	0.980	0.952	0.968
rf	**0.966**	**0.967**	**0.974**	0.982	**0.943**	0.945	**0.975**	**0.981**	**0.969**	**0.971**
cl-25	0.942	0.938	0.964	0.978	0.772	0.802	0.833	0.839	0.961	0.967
cl-100	0.937	0.939	0.961	0.977	0.657	0.692	0.922	0.933	0.964	0.966
cl-200	0.853	0.850	**0.974**	**0.985**	0.654	0.674	0.908	0.920	0.944	0.947

Table 7 Results when training on separate data sets and evaluating on FF, FF=fake-followers, SB1=social-spambots-1, SB2=social-spambots-2, SB3=social-spambots-3, TB=traditional-spambots-1. cl-25=contextual-lstm-25, cl-100=contextual-lstm-100, cl-200=contextual-lstm-200, ada=adaboost, rf=random-forest

	FF		SB1		SB2		SB3		TB	
	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap
ada	**0.955**	0.898	0.925	0.866	0.917	**0.881**	**0.953**	**0.877**	0.880	0.789
rf	0.952	**0.924**	**0.939**	**0.877**	**0.918**	0.813	0.891	0.802	0.795	0.694
cl-25	0.807	0.708	0.579	0.444	0.738	0.456	0.896	0.717	0.906	**0.826**
cl-100	0.884	0.816	0.592	0.345	0.750	0.554	0.908	0.762	0.881	0.804
cl-200	0.741	0.609	0.537	0.418	0.802	0.560	0.886	0.683	**0.909**	0.804

Table 8 Results when training on separate data sets and evaluating on SB2, FF=fake-followers, SB1=social-spambots-1, SB2=social-spambots-2, SB3=social-spambots-3, TB=traditional-spambots-1. cl-25=contextual-lstm-25, cl-100=contextual-lstm-100, cl-200=contextual-lstm-200, ada=adaboost, rf=random-forest

	FF		SB1		SB2		SB3		TB	
	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap	auc-roc	ap
ada	0.789	**0.842**	0.785	0.832	**0.900**	**0.910**	0.800	0.854	0.787	**0.825**
rf	**0.826**	0.818	**0.857**	**0.879**	0.803	0.804	0.841	0.855	**0.813**	0.803
cl-25	0.781	0.757	0.652	0.626	0.869	0.864	0.813	0.788	0.738	0.718
cl-100	0.805	0.789	0.757	0.803	0.878	0.876	**0.862**	**0.867**	0.781	0.749
cl-200	0.794	0.760	0.608	0.628	0.877	0.868	0.789	0.757	0.767	0.732

UMEÅ UNIVERSITY