Reverse Engineering the Behaviour of Twitter Bots

Bello Shehu Bello, Reiko Heckel, Leandro Minku
Department of Informatics, University of Leicester, UK
{bsb14, reiko.heckel, leadro.minku}@le.ac.uk

***Abstract*—Recent research has shown significant success in the detection of social bots. While there are tools to distinguish automated bots from regular user accounts, information about their strategies, biases and influence on their target audience remains harder to obtain. To uncover such details, e.g., to understand the role of bots in political campaigns, we address three questions: Can we describe the behaviour of a bot (when and how a bot takes actions) by a set of understandable rules? How can we express bias and influence? Can we extract such information automatically, from observations of a bot?**

In this paper, we present an approach to reverse engineering the behaviour of Twitter bots to create a visual model explaining their actions. We use machine learning to infer a set of simple and general rules governing the behaviour of a bot. We propose the notion of differential sentiment analysis to provide means of understanding the behaviour with respect to the topics on its network in relation to both its sources of information (friends) and its target audience (followers). Respectively, this provides insights into their bias and the influence aimed at their target audience.

We evaluate our approach using prototype bots we created and selected real Twitter bots. The results show that we are successful in correctly describing the behaviour of the bots and potentially useful in understanding their impact.

***Index Terms*—Twitter bots, social network analysis, reverse engineering, behavioral analysis, differential sentiment analysis**

I. Introduction

The rise of social bots has introduced a new challenge in social network analysis [7]. Social bots are particularly common on Twitter. They tweet, retweet and actively participate in the formation of public opinion [12]. A recent review of Twitter accounts by Twitter Inc. shows that about 8.5% (more than 13.5 million) of Twitter user accounts are social bots [20]. Social bots are automated user accounts that interact with other users, performing a number of actions based on predefined rules [8]. Bots can perform a legitimate task such as delivering news, directing users to useful links and updating news feeds. However, bots are widely used to spread spam, mislead and manipulate the content of social media [8], [12]. Bots are deployed by political actors to attack their opponents and promote political interests. In the UK referendum on EU membership, Twitter bots were used by both sides of the debate [12]. Bots were used in the 2016 US presidential election to support candidates and attack their opponents by injecting information pointing at websites with fake news [8], [17].

Although different approaches and tools [3]–[6] were developed to detect Twitter bots, details about their behaviour remain veiled. This leaves several questions unanswered. How and when do they take actions? What are their targets? What positions do they try to promote? We try to answer such questions by reverse engineering the behaviour of social bots. To the best of our knowledge, our work is the first effort to understand the rules controlling the behaviour of Twitter Bots.

Our aim is to create a model explaining the behaviour of the bots. Once data about a suspected bot is extracted, we use feature selection to detect the attributes used by the bot to make decisions, construct decision trees to infer rules. Our novel rule-based representation simplifies and generalizes the presentation of decision trees to provide a plausible explanation of the bot's behaviour.

We use topic modelling based on LDA [2] to discover the topics of interest to a bot. For each topic, we analyse the sentiment of the tweets produce by the bot and that of the users on its network. We introduce differential sentiment analysis to understand the bot's attitude as compared to that of its users, both in relation to its sources of information (friends, the users followed by the bot) and its target audience (the bot's followers).

Our work on reverse engineering Twitter bots will help to identify their strategies, biases and influence on their target audience. This may help to detect issues such as bots calling for extremist or violent action, or spreading misinformation. We used prototype bots we created and selected real Twitter bots to evaluate our approach. We conducted a series of experiments to test how well the approach will scale on different bots with different behaviour.

The remainder of this paper is organized as follows: Section II covers related work on detection of Twitter bots. Section III describes the details of our approach. Section IV presents our experimental results. Finally, Section V concludes the paper and states our future work.

II. Related Work

Most relevant to our work is research on the detection of social bots, concerned with identification and separation of bots from humans with little or no information about their behaviour. Our work extends the identification of bots by recovering behavioural information. This will allow the analysis of their purpose and influence on social media.

In recent years, social bots have increasingly become more sophisticated, challenging existing detection strategies. A number of detection methods were introduced. The survey [8] classified bot detection systems as graph-based, crowdsourcing and feature based. While the graph-based approach focuses on the structure of the social graph to identify bots, feature-based detection systems focus on behavioural patterns to learn

978-1-5386-9588-3/18/$31.00 ©2018 IEEE

signatures of human-like and bot-like behaviour. BotOrNot, now Botometer[1] is one of the feature-based detection systems made publically available to check the presence of social bots on Twitter accounts [5]. It classifies Twitter accounts as bots by selecting and analyzing features that differentiate between human and automated users. The features are grouped into six main classes: retweet, user mention and hashtag co-occurrences are network features used to compute statistical information such as degree distribution, centrality measures and cluster coefficients. The number of followers and posts are classified as friends features which provide statistics relative to the account. Tweet rate and inter-tweet time distribution are important factors to identify automated accounts. Botometer uses these as temporal features to capture the timing pattern of content generation and consumption. Other features used by Botometer are user, content and sentiment [5]. Currently, Botometer displays a percentage of whether a given Twitter account is likely to be a bot.

In 2015 DARPA organized the Twitter Bot Detection Challenge to developed techniques for early detection of bots [20]. The features used by most participants to identify bots were similar to those used in Botometer. They concluded that, as Twitter bots are becoming more sophisticated there is a strong need for efficient ways to categorize them.

Research [9] makes an effort to analyse social bot infiltration strategies in Twitter. Despite the fact that Twitter is trying to block bots, their work shows that there are many ways Twitter bots can be created in large numbers and live a long life without being shut down. They created 120 social bot accounts with different characteristics and strategies to investigate the extent to which the bots infiltrate the Twitter network. Their results show that even social bots with simple automated mechanisms are able to infiltrate Twitter successfully. They claim that if social bots are created in large numbers, they can endanger our politics by disseminating false information with significant impact on public opinion.

Recently [10] proposed a framework for bot analysis named Stweeler. Although they intend to study the impact of bots on online social networks, they do not discover the rules controlling bots, nor details about their targets and policies. The current version of Stweeler available at github[2] serves as a platform for the collection of tweets, creating honeypots and classification of Twitter account as human or bots. Stweeler uses Bayesian text classification with ranking algorithms to detect spam in tweets and rank the credibility of information disseminated by a bot.

In our work, we propose the reverse engineering of Twitter bots as a way to discover rules and understand their behaviour in much greater detail.

III. METHODOLOGY

Twitter bots tweet, retweet, reply and send direct messages. They also undo their actions. We are specifically interested in social Twitter bots that interact with the public for marketing, political campaigns or volunteering initiatives. Bots that tweet words from a dictionary, correct spelling or report changes in Wikipedia are not the focus of our research. Hence, we define a social Twitter bot as an automated user account that spreads information and interacts with other users. Our first selection of social Twitter bots comes from a list published by [1], [19]. These are bots used during the 2016 US presidential election and the UK Brexit campaigns.

The wide availability of services for automating Twitter accounts without writing code contributes to the large number of automated accounts. We analyzed the services for creating Twitter bots, looking at their service descriptions and creating sample bots. The services analyzed includes labnol[3], botlibre[4], jetbots[5], cheapbotsdonequick[6] and botize[7]. While this gives us an idea about the possible behaviour of simple bots, more sophisticated bots can be created in a programmatic way. To analyse the latter we take two approaches: Firstly, we study the Twitter API which provides programmatic access to public tweets. This is to understand the data available to bots and possible patterns which they could use to take actions. In short, Twitter bots use the Twitter search API to find tweets or users, e.g., tweets with a hashtag #Brexit or tweets from user @realDonaldTrump with retweet count of at least 100. Details about these searches can be found in the Twitter search API page[8] [9].

Secondly, we selected real social Twitter bots and analyzed them, mining data from their accounts for patterns including the relationship between actions performed and sets of attributes of tweets or users. We identify such patterns by extracting attribute values from action occurrences, to learn when and how certain attribute values trigger an action. To represent the resulting patterns visually, we define a general notion of rule for Twitter bots.

A Twitter bot rule is composed of a pattern P and the invocation of an operation. A graphical representation of a rule is sketched in Fig. 1. The pattern of a rule can be based on attributes of tweets and/or users, but can also be more complicated, e.g., considering the follower relation.

A. Data Collection

We built a crawler that mines data from bots via the Twitter API. For each action, such as reply and retweet, we capture attribute values of the original tweets and the users who created them[10] [11]. We extract both graph-based and content features.

[1]https://botometer.iuni.iu.edu
[2]https://github.com/zafargilani/stcs
[3]https://www.labnol.org/internet/write-twitter-bot/27902/
[4]https://www.botlibre.com/forum-post?id=5015
[5]http://www.jetbots.com/twitter/account-creator-bot-2/
[6]https://cheapbotsdonequick.com/
[7]https://botize.com/
[8]https://help.twitter.com/en/using-twitter/twitter-advanced-search
[9]https://developer.twitter.com/en/docs/tweets/rules-and-filtering/overview/standard-operators.html
[10]https://dev.twitter.com/overview/api/tweets
[11]https://dev.twitter.com/overview/api/users

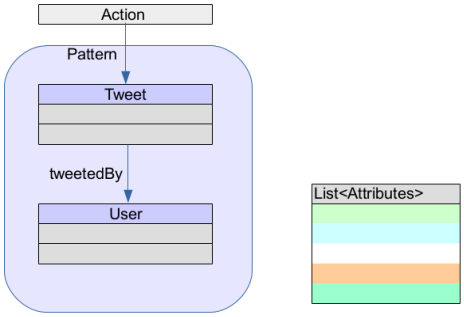


Fig. 1: A template of Twitter bot rule

B. Data Pre-processing

Once data about a suspected bot is extracted, we use the Natural Language Tool Kit (NLTK)[12] to pre-process the data, tokenize text, remove stop and stem words. We also expand shortened URLs and extract their top level domain.

C. Differential Sentiment Analysis

We introduce the notion of differential sentiment analysis to understand the behaviour of a bot in relation to the users on its network. We distinguish two forms of differential sentiment, bias and influence. For each topic, we define the bot's *bias* as the difference in sentiment between the tweets produced by the bot and the tweets on this topic issued by the bot's friends, i.e., the users the bot is following. The difference in sentiment between the tweets produced by the bot and those of its own followers defines the bot's *influence*.

Moreover, we consider the evolution of sentiment over time to detect changes in bias and influence. Recording the bots sentiments for specified time periods, we slice the set of tweets based on the given intervals and compute the average sentiment of each topic. If there are not enough tweets in a given interval, we double the length of this interval. The average sentiment of the bot's followers and friends on the relevant topics are computed for the same time spans.

We use a topic model to compute the set of topics in the tweets. The topic model is based on the Latent Dirichlet Allocation algorithm (LDA) [2] which works effectively on tweet texts [11], [21]. For the sentiment analysis, we use the VADER tool [13] to compute the sentiment of each tweet. This is a rule-based tool designed for social media text. The bot's sentiment with respect to each topic is added as a feature for the analysis.

D. Feature Selection

The aim is to select the most relevant features used by a bot to make decisions. This is an important step in understanding the behaviour. We considered the whole set of features available to Twitter bots[10,11]. For example, a bot may only reply to tweets from verified accounts or with a certain number of followers. This leads to a set of features of different types and priorities. After having performed experiments to find the best feature selection method for Twitter bots, we used l_1-regularized linear SVM an embedded method for the feature selection. Three different methods were evaluated on four different datasets, i.e., mutual information-gain, regularized linear SVM and random forest. Their performance and accuracy were evaluated using four different learning algorithms, namely IBK, naive Bayes, SVM and decision trees. We found that using l_1-regularized linear SVM leads to the most accurate model.

[12]https://www.nltk.org/

E. Machine Learning

Our interest in machine learning is to provide conceptual explanations of the relationship between the input variables (tweet and user features) and the output variables (bot actions). Conventional machine learning algorithms such as naive Bayes can classify entities and provide numerical distributions among variables, but they do not provide a conceptual description of their relationship [14]. Decision trees are a popular method to capture the relationship between input and output variables but have limited representation power. They may become very complex even for simple relationships. We propose a novel rule-based approach for representing the same information.

F. Rule Extraction

We employ graph transformation concepts to generate rules from decision trees for two purposes. First, general rules can cover unseen examples and combine leaf nodes of the same type. Second, they simplify and enhance the readability of the model. Fig 2a shows a simple example of a decision tree of a Twitter bot. We can translate it into a rule of the form `IF pattern THEN action`. This can be represent as shown in Fig 2b. Our representation combines the leaf nodes of the decision tree and generalizes the set of users to cover unseen username.

G. Visualization

We introduce a new visualization that separates the high-level representation of rules from the data. For each action, our visualization produces a graph representation of a set of rules $\{r_1, r_2, \ldots, r_n\}$ formed from the set of attributes used by the bot. A tabular representation of values associated with nominal attributes supplies possible data values. The tabular representation is colour-coded based on sentiment. The family of green colours represents positive sentiment and that of red indicates negative sentiment. The values are sorted in descending order of their distribution in the data, but we intend not to include any numerical representation of their distribution because it is independent of the behaviour of the bot. Fig 7 shows an example of our visualization.

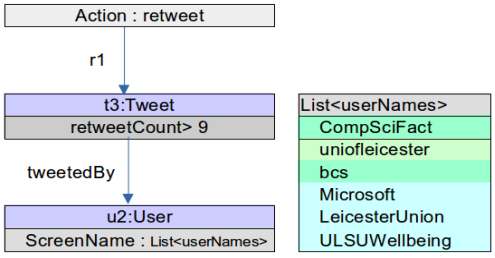
IV. Experiments

The aim of our experiments is to test our approach for the extraction of rules from Twitter bots. We consider different bots, including prototype bots we created and real Twitter bots that actively engage with real users, to validate the result of the rule extraction and sentiment analysis.

```
J48 pruned tree
------------------

T2retweet_count >9
|
|       U2screenName = LeicesterUnion: retweet (12.0)
|       U2screenName = CompSciFact: retweet (47.0)
|       U2screenName = LeicStartups: retweet (4.0)
|       U2screenName = ULSUWellbeing: retweet (3.0)
|       U2screenName = bcs: retweet (22.0)
|       U2screenName = Microsoft: retweet (14.0)
|       U2screenName = uniofleicester: retweet (44.0)
T2retweet_count <= 9
|
|       U2screenName = GaetaSusan: notRetweet (1.0)
|       U2screenName = thinkinglatina: notRetweet (1.0)
|       U2screenName = HersheSquirt: notRetweet (3.0)
|       U2screenName = Nigel_Farage: notRetweet (1.0)
|       U2screenName = FLOTUS: notRetweet (1.0)
|       U2screenName = Corrynmb: notRetweet (2.0)
```

(a) A textual representation of a simple decision tree for Twitter Bot



(b) A representation of the decision tree in Fig. 2a using the notion of our general rule for Twitter Bots

Fig. 2: Rule extraction

For the first experiments, we created prototype Twitter bots. We allowed each bot to run for three weeks and then crawled its data, including all the available features. The idea of these experiments is to challenge our feature selection method to select the smallest relevant subset in a large set of features such that the rule extraction process is able to extract accurate rules governing the bot's behaviour.

In the second experiment, we selected real Twitter bots successful in attracting human users. These include bots that were found to be retweeted by realDonalTrump, the verified Twitter account of Donal Trump. We used Debot[13], a system for the detection of Twitter bots to confirm our selection. For easy reference, we labelled the bots used in this paper as Bot1, Bot2 and Bot3. Bot1 (Don_Vito[14]) is a Donal Trump-supporting bot. It was detected as bot by Debot on 2nd November, 2016, Bot2 (natespuewell[15]) is a Clinton-supporting bot [1]. It was detected as a bot by Debot on 15th September 2016.

Bot3 (BritainStays[16]) is an account of the campaign against Brexit. It is not detected by Debot but is detected by our application that we build to catch automated accounts. We build this application in order to study the behaviour of live automated accounts. Many of the accounts that we have detected were later detected and suspended by Twitter, including @UKIPNFKN which we found to be connected with @BritainStays, reacting to and propagating similar hashtags. On analysis of BritainStays we found evidence of automation in its dataset, see also Table I.

[13]http://www.cs.unm.edu/ chavoshi/debot/check_user.php
[14]https://twitter.com/Don_Vito_08
[15]https://twitter.com/natespuewell
[16]https://twitter.com/BritainStays

A. Data Collection

In each experiment, we crawled a maximum of 3,200 tweets from each bot. These include retweets and replies. We are interested in understanding the users and tweets attractive to the bots. Hence for each retweet or reply we extracted the original tweets and the users who created them. We collected all the available features of the tweets and the users in a table of features and observed the table to established an initial understanding of the behaviour of the bot. Through this observation we learned that Bot1 is retweeting from and replying to a set of users including Donal Trump and to tweets with keywords and hashtags including AmericaFirst, makeAmericaGreat. We also found that Bot1 is replying to many tweets using identical text. This is another feature for identifying bots.

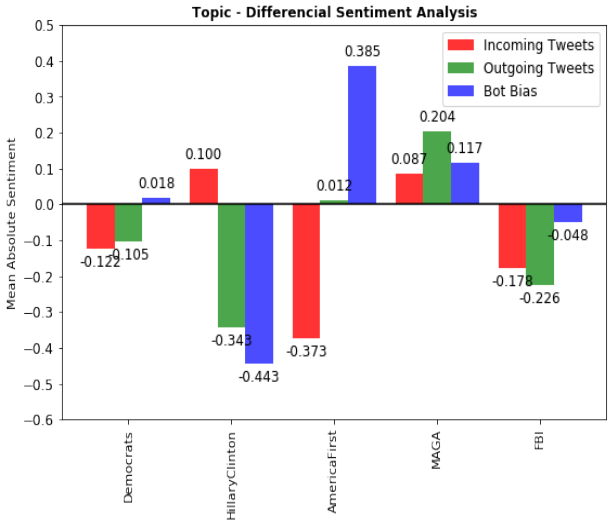
It was difficult to establish a good understanding of the behaviour of Bot2 just by looking at its dataset, so we used our approach to analyse its behaviour and then go back to the data to validate our finding. This suggests that our approach is more useful for understanding the behaviour of sophisticated bots. We found that Bot2 is mainly disseminating tweets against Donal Trump. Fig 3b shows the main topics used by Bot2 against Trump.

In the dataset of Bot3 (BritainStays) we found that its tweets contain hashtags #stopBrexit, #brexitwontwork and links to Brexit news from independent.co.uk, theguardian.com. It also retweeted many of its own tweets, possibly due to poor automation. It retweets tweets containing keywords or hashtags such as Brexit, Theresa May, StopBrexit. It made few replies, containing a well-structured text, Brexit Fact #102, 103 . . . and hashtag #FinalSay.

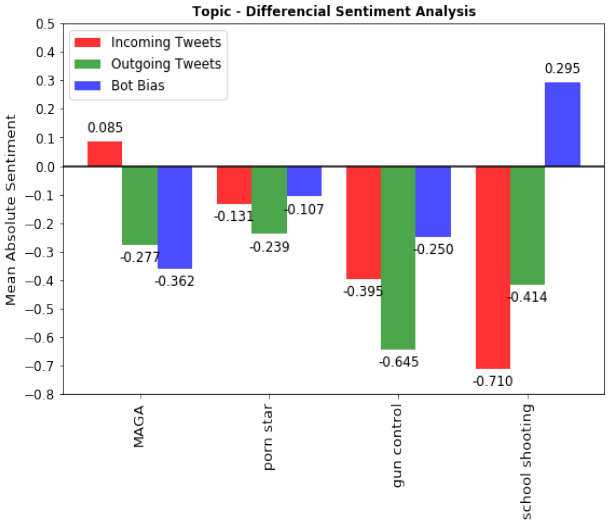
B. Topic Modeling and Sentiment Analysis

We computed the sentiment of tweets disseminated or reacted to by the bots with intensity between -1 and +1. We added the sentiment levels to the features used for rule learning. We use LDA to compute the set of topics for which the bot is positive, negative or neutral. This allows the model to capture the main topics of interest to the bot. We compute differential sentiments to provide means of understanding the behaviour of a bot in relation to the users on its network.

Fig. 3a and 3b show the resulting differential sentiments of bots 1 and 2. Bot1 is biased opposite to its friends with regards to HillaryClinton and AmericaFirst. It is promoting tweets which are negative about Hillary Clinton while users on its network are generally positive. This is seen as a clear sign of bias, and can also serve as a strategy to convert users, trying to change their opinion. Bot2 is biased opposite to its friends with regards only to MAGA (Make America Greater Again) but shares their opinion on other topics. Looking at the results of the two bots, Bot1 engages in both positive and negative campaigning, while Bot2 is mainly negative. In

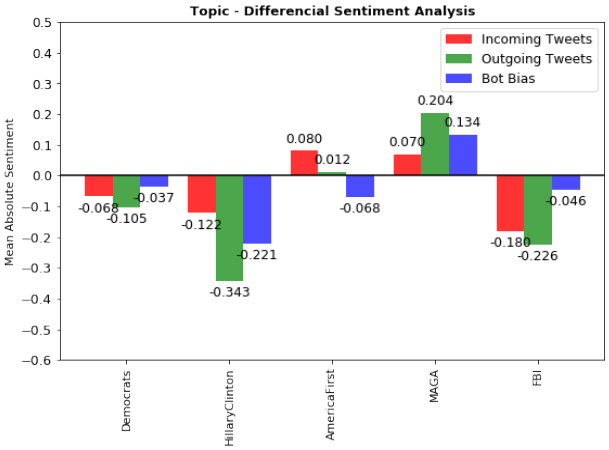


(a) Bot1 opposes its friends w.r.t. to *Hillary Clinton*, *America First*

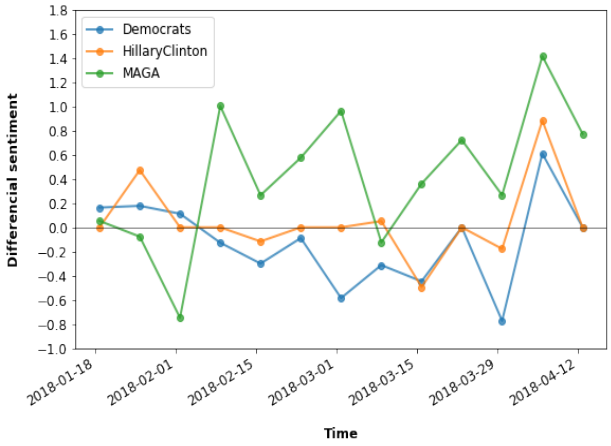


(b) Bot2 opposes its friends w.r.t. *MAGA*, but agrees on other topics

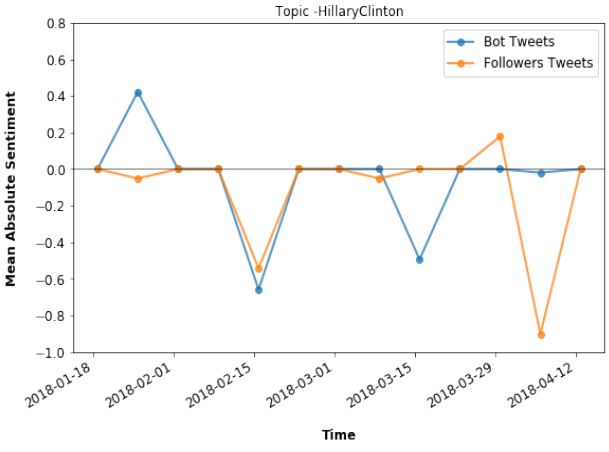
Fig. 3: Bias differential sentiment



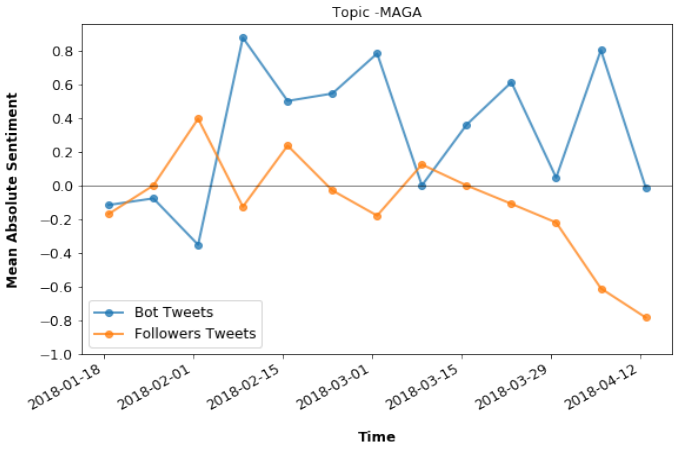
(a) Bot1 shares opinions of its followers on all topics



(b) Bot1's influence differential sentiment over time



(c) Bot1's follower's absolute sentiment over time on *Hillary Clinton*



(d) Bot1's follower's absolute sentiment over time on *MAGA*

Fig. 4: Bot1's influence, over time differential sentiment

particular, while Bot1 is promoting tags from Donald Trump, it is also campaigning against Hillary Clinton. Bot2 is mainly against Donal Trump without specifically promoting content with regards to Hillary. In addition to detecting bias, we use differential sentiment analysis to study the intended influence of the bots. The majority of followers of Bot1 and 2 have the same opinion as the bots on all topics. Fig. 4a shows this for Bot1. Even if they agree, we notice that Bot1 is more strongly against Hillary Clinton and supports MAGA. The followers promote the tag AmeracaFirst more than the bot. Differential sentiment over time (DSO) provides more insights into how the relationship between a bot and its followers evolve. Fig. 4b shows Bot1's influence differential sentiment over time.

Influence differential sentiment over time (IDSO) tells us the influence of a bot on the opinion of a network. Given the overall polarity of a bot's differential sentiment (DS), any point that is opposite to that polarity on IDS over time means the bot may have had little impact on the network with respect to that opinion at that time. For example, Fig. 4a shows that Bot1 is negative toward Hillary Clinton but if we look at its IDS over time in Fig. 4b, we notice that in week 2 (25-01-2018) and the second to the last week of the analysis (05-04-2018), the DS turns out positive. Hence Bot1 seems to have no influence against Hillary Clinton in these weeks. If we look at week 9 (15-03-2018) we notice that it is a period where Bot1 could have made a significant impact. IDS over time helps us to identify periods where a bot could have had an impact on a network, however since there are many other factors, we only observe correlations, not causal links. The absolute sentiment over time simplifies the understanding of DSO over time. Fig. 4c shows Bot1's followers absolute sentiment over time on Hillary Clinton. We can see that in the second to last week (05-04-2018, week 12) the bot is less negative against Hillary Clinton then its followers. This is a simple explanation of why the IDS over time shows up positive in that week, to which the bot has contribute less. Moreover, the absolute sentiment over time provides a view of how the sentiment of a bot and its followers change independently. Fig. 4d provides a view of how Bot1 dominated the network with tweets supporting MAGA more strongly than its followers.

C. Feature Selection

After pre-processing the data, vectorization of the remaining tweet text into bi-grams and coding the data ready for machine learning, Bot1 has 814 features Bot2 has 721 and Bot3 has 701. We used regularized l_1 linear SVM as a feature selection method to reduce the number of features for Bot1, 2 and Bot3 to 427, 291 and 84 respectively.

D. Rule Learning

We implemented rule learning based on scikit-learn [15], an open source machine learning tool supported by Google and INRIA. We treated our learning process as a binary classification problem, where in each case we either learn rules for one action (as opposed to any of the others), or between two actions. We use grid search to cover a wide range of hyperparameters to find optimal choices. They include the maximal depth of a tree, minimal number of samples per split and leaf, the penalty parameter C of the Linear SVM and class weight. We used 5 fold, 10 repeats stratified cross-validation. We used a roc_auc score as our scoring parameter. This optimizes rule learning to best separate the action regarded as the positive choice from others regarded as negatives.

E. Rule Extraction

We use a decision tree classifier to build decision trees based on the result of the feature selection process [16]. Fig 5 shows a representation of the decision tree generated from Bot1 before rule extraction. The figure indicates the limited representation power of decision trees. Looking at the visualization in Fig. 6, our approach simplifies and generalizes the decision tree using a simple but effective rule-based representation derived as follows.

First, we traverse the tree. From the root, we follow each branch and collect the set of decisions leading to each leaf node (class) in a rule instance. Then we group rule instances according to classes and remove redundant cases. For each class, we look at its set of rule instances and remove those that are already covered by other rules with a larger number of examples.

Fig. 6 visualizes the retweet behaviour of the bot. It covers unseen examples by separating structural features from attribute values. The representation includes lists of users, keywords or hashtags that are part of the rules, but were not captured in the sample data.

Rule 1 says that the bot retweets tweets containing a set of keywords with favorite count at least five. The keywords are listed in an attribute table which is colour-coded based on sentiment. Keywords such as Travel ban, Ties Russian are coloured orange because the tweets are negative while AmericaFirst is green because tweets associated with it have positive sentiment.

Rule 2 states that the bot retweets tweets containing particular hashtags that have been retweeted at least 10 times.

Rule 3 specifies that the bot retweets tweets created by a set of users including realDonalTrump and the bot itself (Don_Vito_08). This is another feature identifying bots, which retweet many of their own tweets due to automation.

Fig. 7 represents the reply behaviour of Bot1 using our approach. There are three patterns which trigger the bot's reply. First, tweets that contain specific keywords; second, tweets with user mentions realDonalTrump; third, tweets from certain users. The table below each rule provides details about the attribute values associated with the rule. The separation of rules from attribute values simplifies and generalizes the presentation.

V. Evaluation

We evaluate our approach in two ways: qualitatively, to check how well the model reflects the observed behaviour of the bots and quantitatively, to validate performance on unseen data.

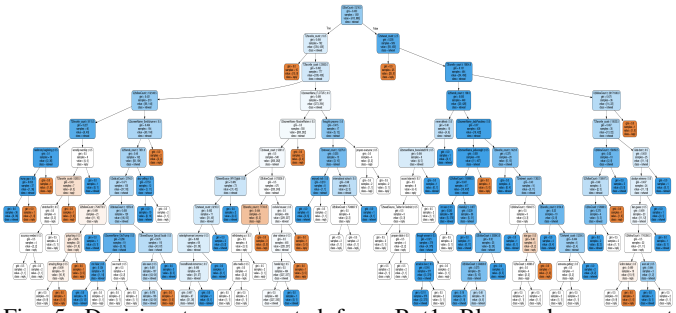


Fig. 5: Decision tree generated from Bot1. Blue nodes represent decisions leading to retweet, orange nodes indicate replies.

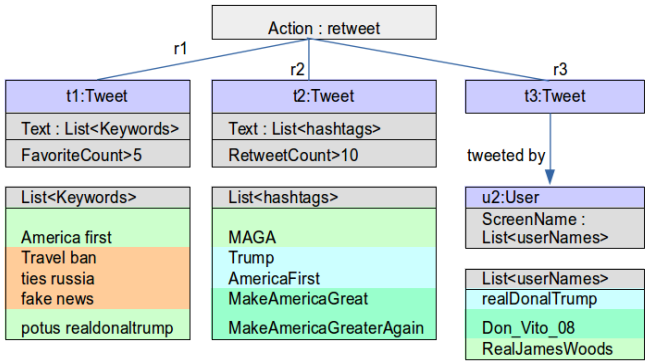


Fig. 6: Retweet rules for Bot1 generated from the decision tree in Fig. 5.

A. Qualitative

We assess the quality of the rules produced using the proposed approach. That means to assess the extent to which the results match our observations of the actual behaviour of real bots or the rules controlling our prototype bots. The prototype bots provide the first ground truth, as we are aware of the actual behaviour of these bots. We found that our approach has successfully recovered the rules. For the real bots, the behaviour observed via the dataset form the ground truth. We found that our approach recovers details that match the observed behaviour. It also captures additional information

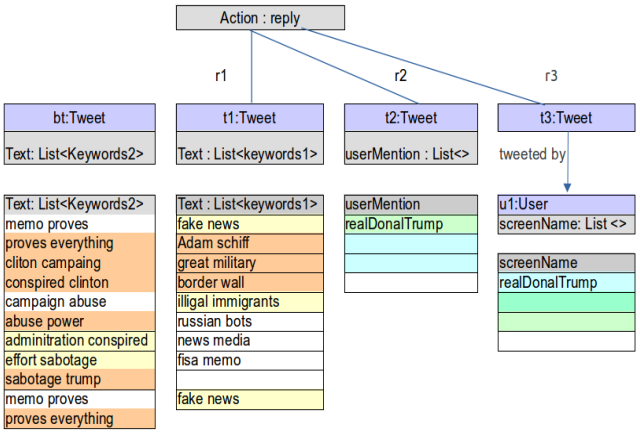


Fig. 7: Reply rules for Bot1.

that may be difficult to find through human observation. It can be seen that the retweet rule for Bot1 shown in Fig. 6 matches our initial observation that the bot retweets tweets that contain a specific keyword or hashtags related to Donald Trump. In addition, it captures the relationship between the keyword and the tweet's favourite count, hashtags and tweet's retweet count. Although we have not noticed this during the observation, the bot uses this as a strategy to select the most important tweets. Differential sentiment analysis sheds more light on the interests and target of the bots.

B. Quantitative

To evaluate the results in the case of real bots, we use standard machine learning techniques. We gather separate independent test data to check how well the model from which we generate the rules will perform on unseen data. For Bot1 and Bot2 the training data consists of 2,767 and 1,792 tweets, while the test data comprises 2,908 and 927 tweets respectively. In the case of our prototype bots, we explore different cases, such as limited data and class imbalance. Our aim is that the approach should capture patterns of behaviour within smaller representative examples. We varied the size of training vs test data, using positive and negative examples to check the quality of the decision trees generated. The two prototype bots have training date sizes of 604 and 342 tweets, test data sizes of 339 and 388. The prototype bots are called Helpdesk and CSInfo_news. The results are presented in Table I based on measures commonly used in machine learning and information retrieval, i.e., precision, recall and area under the receiver operating characteristic curve (roc_auc_score). While precision measures the exactness of the model, recall informs us about completeness. They are formally defined as *Precision = TP/(TP+FP)* and *Recall = TP/ (TP+FN)*.

The roc_auc_score is the area under the curve of the true positive rate (recall) plotted against the false positive rate (FPR). The FPR is defined as *FPR = FP/(FP+TN)*.

With regards to the classification of a bot's action as retweet or notRetweet we define the above norms as

- True positive (TP) = number of retweets that are correctly classified as retweets.
- True negative (TN) = number of notRetweets that are classified as notRetweets.
- False positive (FP) = number of notRetweets that are classified as retweets.
- False negative (FN) = number of retweets that are classified as notRetweets.

The roc_auc_score is one of the most popular measures to evaluate robustness of a classifier. Using retweet as an example, this is the probability that the classifier will judge a randomly chosen retweet action as retweet rather than a randomly chosen notRetweet as retweet. This is important in rule learning, as we want to avoid false rules while aiming to recover as many correct rules as possible. A roc_auc_score of 0.5 is considered poor while 1.0 is excellent. In predictive studies of psychology, law and human behavior, roc_auc_score of 0.70 or higher is considered strong [18]. In that sense, the

results shown in Table I show that the models from which we generated the rules are good enough, the least roc_auc_score value of natespuewell being 0.70. We noticed that some of the topics in the training data are not in the test data, but the overall behaviour does not change. This is the advantage of our rule representation. The rules will remain the same, only the tabular values will change. We achieved roc_auc_score of 0.95 and 0.98 in the Helpdesk and BritainStays datasets because of the strong pattern of the test data in the training data. This shows a strong predictive power of the models.

TABLE I: The performance result of the model on an unseen data

Data set	***Accuracy***	***Precision***	***Recall***	***F-Measure***	***AUC_Score***
Helpdesk	0.95	0.95	0.95	0.95	0.95
CSInfo_news	0.86	0.87	0.86	0.85	0.82
Don_Vito_08	0.80	0.83	0.78	0.80	0.75
natespuewell	0.90	0.91	0.90	0.88	0.70
BritainStays	0.98	0.98	0.98	0.98	0.98

It is important to note that bots behaviour is mainly event-specific. It may change over time due to shifts from one event to another. Therefore, unlike other machine learning problems, having more training data will not always lead to a good predictive model. We intent to use incremental learning to detect changes in the behaviour to allow our approach to report different rules for different behaviour.

VI. Conclusion and future work

In this paper, we presented an approach towards reverse engineering of Twitter bots. We defined a visual format for rules to represent their behaviour and discussed the experiments used to test our approach. Our visualization simplifies and generalizes the representation of bots' behaviour and helps to arrive at a plausible explanation.

We introduced differential sentiment analysis to provide means of understanding the behaviour of a bot in relation to its friends and followers. We introduced influence differential sentiment analysis over time (IDSO). While [12] find it difficult to ascertain the influence of bots on its followers due to other external factors associated with the followers, our IDSO can indicate periods where such influence could have happened. The score of the differential sentiment can be used in identifying bots that have a higher chance of making an impact. The absolute mean sentiment over time can be used to track and understand the response of a bot to an external event.

We are working towards the full automation of the process for extraction and representation of rules. Based on such automation we would like to study how a bot's rules changes over time, to detect and interpret such changes.

More generally we are looking forward to building a web-based tool for reverse engineering and analysis of Twitter bots. This will allow the extraction and representation of rules from online data as well as an analysis of their differential sentiment. The tool will also enable us to evaluate our approach and representations in a wider context.

References

[1] Alessandro Bessi and Emilio Ferrara. Social bots distort the 2016 us presidential election online discussion. 2016.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[3] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. Debot: Twitter bot detection via warped correlation. In *ICDM*, pages 817–822, 2016.

[4] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing*, 9(6):811–824, 2012.

[5] Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 273–274. International World Wide Web Conferences Steering Committee, 2016.

[6] John P Dickerson, Vadim Kagan, and VS Subrahmanian. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 620–627. IEEE, 2014.

[7] Geli Fei and Bing Liu. Social media text classification under negative covariate shift. In *EMNLP*, pages 2347–2356, 2015.

[8] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *arXiv preprint arXiv:1407.5225*, 2014.

[9] Carlos Freitas, Fabricio Benevenuto, Saptarshi Ghosh, and Adriano Veloso. Reverse engineering socialbot infiltration strategies in twitter. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 25–32. ACM, 2015.

[10] Zafar Gilani, Liang Wang, Jon Crowcroft, Mario Almeida, and Reza Farahbakhsh. Stweeler: A framework for twitter bot analysis. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 37–38. International World Wide Web Conferences Steering Committee, 2016.

[11] Liangjie Hong and Brian D Davison. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*, pages 80–88. ACM, 2010.

[12] Philip N Howard and Bence Kollanyi. Bots,# strongerin, and# brexit: Computational propaganda during the uk-eu referendum. *Available at SSRN 2798311*, 2016.

[13] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[14] Kenneth A Kaufman and Ryszard S Michalski. 2-from data mining to knowledge mining. *Handbook of Statistics*, 24:47–75, 2005.

[15] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[16] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[17] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. Detecting and tracking political abuse in social media. *ICWSM*, 11:297–304, 2011.

[18] Marnie E Rice and Grant T Harris. Comparing effect sizes in follow-up studies: Roc area, cohen's d, and r. *Law and human behavior*, 29(5):615–620, 2005.

[19] socialpuncher Sadbottrue.com. Chapter 15. make america bot again. part one sadbottrue —. [online] available at: http://sadbottrue.com/article/chapter-15-make-america-bot-again-part-one/ [accessed 15 may 2018]. 2016.

[20] VS Subrahmanian, Amos Azaria, Skylar Durst, Vadim Kagan, Aram Galstyan, Kristina Lerman, Linhong Zhu, Emilio Ferrara, Alessandro Flammini, Filippo Menczer, et al. The darpa twitter bot challenge. *arXiv preprint arXiv:1601.05140*, 2016.

[21] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.