Inferring the 2016 USA election foreign state-sponsored accounts on Twitter

Anis Myriam 2571698 s8myanis@stud.uni-saarland.de
Sourial Maggie 2571492 s8masour@stud.uni-saarland.de
Ramb Yannick 2550216 s9yaramb@stud.uni-saarland.de
Tikhonov Peter 2562202 s8petikh@stud.uni-saarland.de
Tabachnikov Roman 2565209 s8rotaba@stud.uni-saarland.de

***Abstract*—Since the dawn of internet, social media has been strongly affecting our personal lives. The world surrounding us has been changing to adapt to this virtual space. With every passing year, we see more and more how our physical and digital lives are merging together. Nevertheless, the 2016 USA elections have been a turning point in proving the effects of social networks on our every day lives. After the dust settled at the end of the 2016 elections, a USA based social network has come under scrutiny for allowing hostile agents to exploit the sites communication channels to subvert the general discussion and change public opinion. According to an ongoing congress investigation, Russia is being blamed for sponsoring malicious twitter accounts with the intend to cause disinformation and hinder the democratic process to their advantage. In this study we construct a method of inferring such state-operated accounts based on publicly available information using two different machine learning techniques. Based on a publicly released list of 2,752 now-deactivated Twitter accounts that were officially identified as working on behalf of Russias Internet Research Agency. We reconstructed over 200k posts belonging to almost 900 of those accused accounts which were posted during September 20 and October 12, 2016. Together with 200k of unique posts from the same time period which were submitted by benign accounts not included in the list we we successful in processing this information into a machine-terms using "doc2vec". Finally we used an array of already-proved pre-detemrined models and one closed neural-network to deduce the troll accounts from the benign one with an accuracy of more then 80% based only on their corresponding tweet content.**

1. INTRODUCTION

The Internet allows us to connect with people all around the world, if it be in commerce, research or simple human interaction. In fact nowadays in most developed countries it is considered as one of the basic human rights and is being protected by multiple laws (Net Neutrality to name one). Nonetheless, just like any other tool it can be used for good or for malicious purposes and considering that this mean of communication does not let itself to such easy definition and governess as firearms or heavy-machinery; this can be easily exploited by malicious agents. One of the biggest stories of such an act occurred during the recent presidential elections in USA, where attackers used 'fake' Twitter accounts for the means of propaganda to sway public opinion and hinder the democratic process. During the investigation of the US congress in the end of 2017 with the help of specialists and the team at Twitter, as many as almost 3000 accounts were manually classified as 'fake' and deleted from the platform for good. It is assumed by the specialists presenting those evidences [18] before the congress, that the true scope of this incident is far greater then they ever expected. Being one of the largest online social platforms with around 330 million monthly active users and 500 million tweets per day[1], Twitter clearly doesn't lend itself to the same manual approach to uncover the rest of the misbehaving accounts. However, based on this list it is possible to construct a more effective way of discerning those rouge accounts from benign ones. Thus creating a solution for further investigation into the meddling of Russian in the 2016 elections, but most importantly creating a tool which could be used in future research of similar instances in other couturiers and in other platforms.

Our general idea to solve this problem involves a binary classifier with one output class; either the tweet is a 'troll' (bot) or not. We will use the words 'troll' and 'bot' interchangeably throughout this report since a troll can be an automated bot or a human. We decided to go with predetermined models and neural networks, two very different techniques, which will let us reference and compare our results over multiple possible implementations. Similar topics were already discussed in several publications[12], [13] but usually they deal with more accent on the influence of the 'trolls' on the public opinion concerning the two political parties using twitter as gauge instead of inferring the culprits in the crowd. Moreover, we feel that the data which those researchers use in their papers is far from realistic and is easily altered, like time of tweets or the location of the user. Our approach is based on the tweet content itself, which makes it also possible to detect 'trolls'

1. Tweeter statistics https://www.omnicoreagency.com/twitter-statistics/

that register brand new account or use any other mean to pretend to be a regular person. In addition, we also briefly discuss if hash-tags can be a good indicator or not and why meta-tags are such a bad idea.

The paper is organized into several parts. First we discuss the dataset we used as well as the pre-processing steps to bring it to a more easily workable state. Then we evaluate different inference techniques including the pre-determined models and the neural network. In the end we discuss how our approach may be used and the ways it can be further improved in future research.

2. Datasets

2.1. Troll dataset

During the investigation of the House Intelligence Committee into the way Russia allegedly interfered with the 2016 US Elections, US officials were successful in producing a list of Twitter user names[3] which were tied to the Russian government's "Internet Research" division, a propaganda outfit founded in 2013. Shortly before being published as part of the evidence presented to the US jury, those accounts and their corresponding tweets were deleted from the Twitter servers by Twitter.com representatives in a bid to avoid confusion and prevent social backlash against the platform. After the list was published for the world to see, Ben Popken and Ej Fox from NBV News reconstructed part of the deleted content using Web caches that saved the deleted account's data and published their findings in a series of publications. Most importantly they published their dataset to Kaggle.com as open-sourced reconstruction[4], consisting of 870 different usernames and more then 200 thousand unique tweets. Lastly we verified that this dataset contained only names from the US-Congress provided list.

2.2. Benign dataset

To be able to conduct our experiment, we still needed a dataset with users we know weren't Russian bots, while still being in the similar time-span and similar thematic content. Luckily enough, the elections in 2016 were very well documented and we succeeded in finding a dataset of a raw stream of tweets which were published on the day of the elections. In 2016 Chris Albon scraped and later posted the tweet Ids and posted them on his GitHub page[2]. This in turn was parsed and reconstructed using twitter's API by Ed King and published as an open-source dataset on kaggle.com [5]. This dataset included more than 400 thousand unique tweets by almost as many unique users dealing with, but not limited to, the 2016 election day. After making sure that the list did not contain any names from the list of accused twitter user names we checked the number of accounts in the list which were "verified". Verified Twitter accounts go through an additional verification process which prevents malicious users from pretending to be an official spokesman or stealing the identity of someone famous. We assumed that state-controlled accounts wouldn't care for such additional authentication and the probability of a verified user being controlled by a rouge-state would be close to zero. Currently there are 300k verified users worldwide (according to the official @verified [6] twitter account, which follows such users). Because our set contained only a handful of such users, we had to think of a better plan to fulfill our requirement of a "troll-free" dataset. According to twitter's statistics [7] currently there are 69 million active accounts in the US. Assuming a worst case scenario where 250 thousand of those are foreign-controlled accounts we get a 1/276 ratio; which would mean almost 1500 our of our dataset could be trolls. Taking into account our approaches - we concluded that this is negligible considering our first step of scanning for the well-known troll names. Finally, even though our benign dataset included tweets dealing not only with the elections of that same day, after going through the trolls tweets we found out that they also were posting about issues non-related to the elections. We can only assume that the accounts tried to blend in and avoided posting only political content in fear of being called out or to seem more relatable to a wider audience.

Datasets	tweet-troll.csv	tweet-benign.csv
Rows	203k	413k
Columns	16	28
Users	870	370k

3. Pre-processing

The pre-procesing work done on the dataset can be divided into 2 main parts: feature cleaning and encoding, tweet vectorization and leverage removal.

3.1. Features Cleaning and Encoding

The dataset used initially contained the following features: `user_id`, `user_key`, `tweet_id`, `created_at`, `retweet_count`, `retweeted`, `favorite_count`, `hashtags`, `mentions and other features` .
To handle dataset biases, features that can cause memorization and overfitting of our models such as: user_id, user_key, tweet_id were directly removed.
Also features that contained more that 20% NA values were directly removed (e.g retweeted).
Also, instead of directly using the time of posting a tweet as a feature, we added a new feature: time_difference. Time_difference considers per user posting behaviour. It indicates the time the user usually take to post a new tweet. The main target was to investigate if there is a pattern or a difference in the posting behaviour between benign users and trolls.
Moreover instead of using the mentions directly, which can

2. Chris Albon, https://github.com/chrisalbon/election_day_2016_twitter, March 2018

be subject to memorization and overfitting, we only used the number of mentions per tweet. The final set of meta features can be found in the following table.

Feature Name	Description
time_difference	Time taken between the user's current tweet and his previous one
retweet_count	Number of retweets for a specific tweet
favorite_count	Number of favorite counts for a specific tweet
hashtags	Number of hashtags in the tweet
mentions	Number of mentions in the tweet
created_at	Time of posting of tweet

3.2. Tweet Vectorization

In the original dataset, tweets were reported as texts. To be able to feed them into a learning model, these texts were converted into vectors. The Doc2Vec python library was used to reach this target. Initially, a Doc2Vec model was trained on all tweets (trolls and non-trolls) creating a model that would output a vector of 100 dimensions for each tweet. Then each tweet vector was appended to its corresponding metadata explained in section 3. To produce the final dataset, an equal number of troll and non-troll tweets was chosen and randomly shuffled.

3.3. To meta or not to meta

For our learning algorithms, we tried to use a dataset with and without meta-features.
Our goal was to reduce any bias effect and ensure that our approach can detect the behaviour of bots and generalize to new unseen data and scenarios with good accuracy without memorizing the given dataset.
In order to make sure that no biases were induced in the data due to leverage points which could be due to recording of faulty data, leverage points were removed by calculation of removal of points having standard deviation more than 3 standard errors from the mean of the data.

3.4. Final words on datasets

It goes without saying that we balanced our dataset; forming one which contained 400 thousand tweets with a shuffled 50-50 balance between the troll and not-troll accounts. Choosing the 200 thousand out of the 400 thousand benign examples was random. Furthermore, we decided to break it into smaller pieces to help us in the process of pinning down the best approach, while still preserving the shuffled 50-50 balance within them.

4. Inferring trolls

Our work can be divided into 2 main parts: predetermined models and neural network.

4.1. Predetermined Models

The learning methods implemented can be divided into 3 types:

1) **Linear:** logistic regression and linear discriminant analysis (LDA).
2) **Quadratic:** Quadratic discriminant analysis (QDA).
3) **Highly non-linear:** K nearest neighbours (KNN), splines and support vector machines(SVM). For KNN, multiple K were tried in the range from 5 to 25 and the one that reported the highest accuracy was chosen.
   For SVM, multiple kernels were also tested: linear, quadratic and radial, and the one with the highest result was used.
   For splines, different knots in the range from 3 to 15 were tried.
   For random forests, different numbers of trees ranging from 400 to 800 trees, reporting best results at 500 trees which was later used.

Since we care about the interpretability of our model as we care about the flexibility and the accuracy, the linear model was tested first. Linear models reported acceptable but not good enough accuracy as we will describe more detailed in the next section.
The low accuracy in the linear model indicated that the data is not linearly separable so we moved to a more complex model: KNN. KNN performed worse than linear models indicating that the data is not locally similar.
Quadratic models were then introduced which caused a relative increase in the accuracy. This motivated us to use higher order models such as splines but the accuracy didn't improve much and with larger subsets of the dataset, SVM and splines performed even worse indicating that they cause overfitting of the data.
Since it is a classification problem, random forests is known to be a very efficient method for such problems, which motivated us to also try random forests.

4.2. Implementation and Evaluation

All the models described in this section were implemented in R.
We tested all of these models on 2 datasets: one with the tweet vectorization and meta-features and another one only with tweet vectorization.
Also to save training time, we first tried on a subset of the large dataset. This subset consisted of 3000 tweets divided equally between trolls and non-trolls to prevent any biases. The top 3 models that achieved the highest accuracy were

then used to train on the full dataset. To test the generalization power of the models, the dataset was divided into training (80%) to train the model and testing (20%) to evaluate the performance of the models.
Since our data is highly dimensional, especially because of the vectorization of the tweets, PCA was also used to test if dimensionality reduction would enhance the accuracy. Finally for our evaluation metrics, we used misclassification rate and area under the curve (AUC).
On the dataset subset, the results were as follows:

Model	accuracy without meta-features	AUC without meta-features	accuracy all features	AUC all features
KNN	0.519	0.51	0.6	0.589
Logistic	0.597	0.5959	0.71	0.709
LDA	0.597	0.59589	0.633	0.632
QDA	0.7887	0.7888	0.973	0.972
Splines	0.676	0.629	0.74	0.7398
SVM	0.559	0.556	0.637	0.6388
Random Forests	0.710	0.709	0.902	0.904

We also tested the models with PCA with the 60 principal components since as shown in the following graph, would explain more than 90% of data variance.

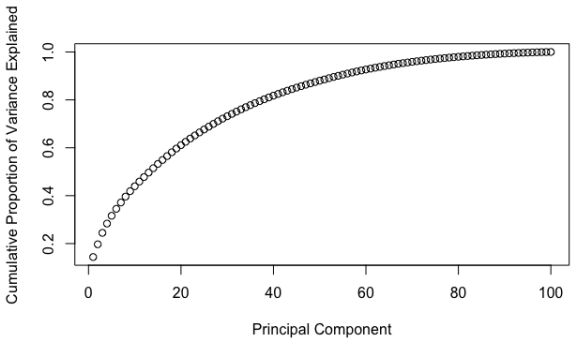

Figure 1. Variance explained by principal components

Model	accuracy without meta-features	AUC without meta-features	accuracy all features	AUC all features
Random Forests	0.697	0.696	0.732	0.732

As shown in these results, PCA did not enhance the accuracy of random forests, although random forests had very high accuracy without it, hence it was not applied on the full dataset.
The very high results reported in case of QDA with meta-features indicated a possible bias so for the full dataset testing we only tested our models against the dataset of tweet vectorization. And the results were reported as follows:

Model	Accuracy	AUC
Logistic	0.605	0.602
LDA	0.605	0.603
QDA	0.787	0.7868
Random Forests	0.755	0.756

4.3. Neural-network Model

Back when we still considered using the meta-tags as features; a hybrid of a simple linear (for the meta-tags) and a neural network (for the vectors) was the logical solution. Changing our feature-set meant that we could focus entirely on the neural network to produce a binary classifer using a closed neural network. Based on the highly recommended Tensorflow [9] package and the Keras API [10]; we divided our work into two solutions. The first approach was to build our own Classifier using the Keras API with Tensorflow backend [11], while the second approach was to use a pre-made DNN Classifier from the Estimator API[8] in Tensorflow as a reference.

As with our Predetermined Models; we started with the smaller datasets and made our way into longer runs using the results to adjust our hyper-parameters. Finally we used the biggest dataset of 400 thousand tweets divided into 80% training (20% of which we used as evaluation to improve goodness of our fits) and 20% test, all balanced over the label's binary class.

For our first approach, we decided to use the Keras API, since it allowed us to build and evaluate custom configurations of neural networks. Furthermore, this allowed us to use sklearn's API for gridsearch [15] and k-fold cross validation [16], which proved to be very handy in figuring out the configuration to achieve the best model performance. While building and testing our model we encountered many problems. First of all, it turned out that we didn't have much of a choice concerning the shape and size of the layers. We decided, according to a rule of thumb [17], that our model should have the same number of neurons in the input layer as we have input features and only one neuron in the output layer, since we used the sigmoid function for binary classification. Following the conventions even further, the number of neurons in the hidden layers were only values between the number of neurons in the input and output layer.

Initially, we left all model parameters on their given defaults and used gridsearch to find out which network shape (that is, the number of neurons per hidden layer) gives the best results. We discovered very quickly, that bigger (deeper) networks gave useless results as they did not learn anything at all. Their accuracy during the training phase

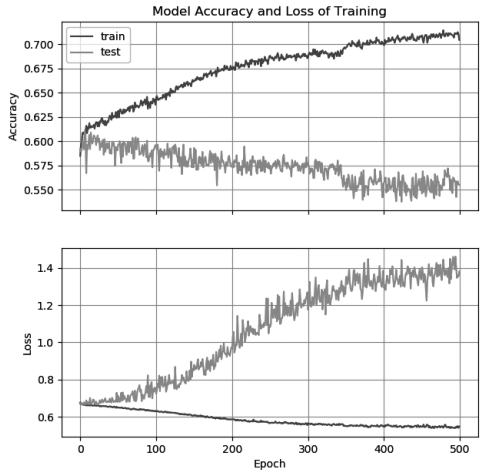

Figure 2. diverging training and evaluation graph using RMSprop

never grew higher than 50%. Therefore, we used only one or two hidden layers for future tests. We used the same approach of trial and error to figure out the best optimizer for our case. The best ones were RMSProp, SGD and Adagrad [14] and although RMSProp achieved the highest accuracy at first (71%), when plotting accuracy and loss per epoch the training accuracy increased while the evaluation accuracy decreased at the same time (vice verca for training and evaluation loss) 2. With SGD we initially achieved a disappointing 49% accuracy, which left us with Adagrad as our optimizer of choice, giving us about 60% accuracy.

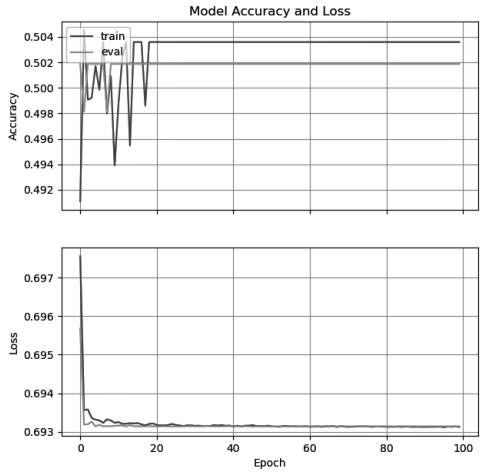

Figure 3. result only tweaking learning rate

Moving to further hyper parameters left to tweak; learning rate, dropout and batch-size, with the first and second achieving best results set to 0.01 or 0.02 and 0.1, respectively. Especially the learning rate had a big influence on our model as with higher learning rates, it quickly came to oscillation and caused our model to stagnate. Finally, our model preformed best with a batch-size of 16 or 32, which we figured by comparing runs on the small dataset. Once we built our model baseline, we decided to fit the hyper parameters into a wide array of different configurations. Similar to above, we tried to optimize only one hyper parameter at the time, leaving the others on the mentioned ”best” values. By taking the best result per hyper parameter we hoped to get an overall better performing model. The results were mostly unusable, presumably because the parameters are very much dependent on each other such that we could not freely tweak one hyper parameter at a time without throwing the system off-balance 3. This approach showed that the only way to improve our model further is to tweak all hyper parameters at the same time, which was regarding running time of the program, only feasible for a small number (e.g. 2, or 3) of different values per hyper parameter.

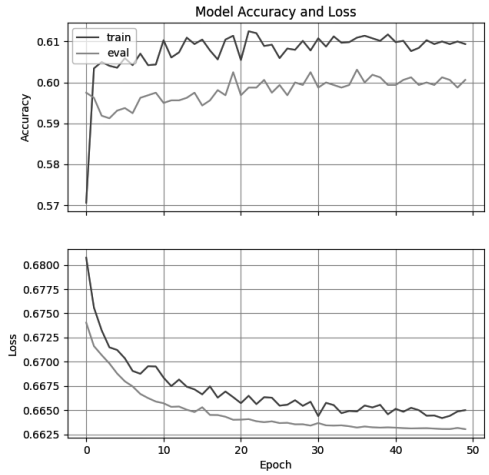

Figure 4. Accuracy and Loss using Adagrad on small dataset

Overall, we could not achieve better results than around 60% accuracy on the smaller datasets. We plotted loss and accuracy per epoch while fitting our model, using Adagrad as optimizer. It could be seen, that the curve flattened at accuracy of roughly 60%, indicating our model reached its learning peak and won't produce any significantly better results 4.

However, when fitting and testing the model on the biggest dataset consisting of 400 thousand tweets, the very same implementation reached about 87% accuracy 5, coinciding with the result of the pre-made DNN Classifier we used later in Tensorflow. The accuracies in the table are the result of testing the model using the depicted optimizers, learning rate as well as units [100, 40, 1], dropout of 0.1 and batch-size of 16 on the small and on then the big dataset (acc_{small} and acc_{big}, respectively). Interesting enough; even with SGD, which performed so badly on the small dataset, we achieved a similarly high

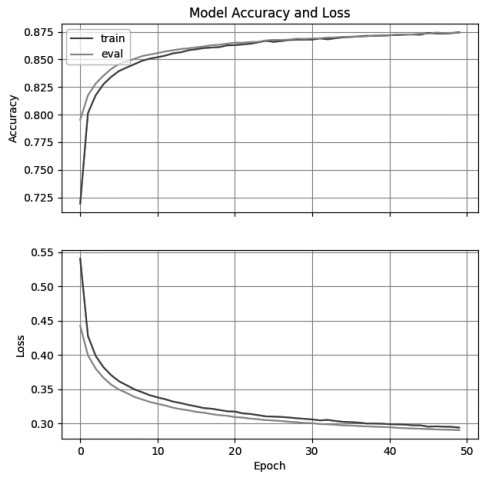

Figure 5. Accuracy and Loss using Adagrad on big dataset

result of about 86% on the big dataset.

Optimizer	lr	acc_{small}	acc_{big}
SGD	0.01	0.489875	0.854653
SGD	0.02	0.495750	0.860250
Adagrad	0.01	0.602875	0.869171
Adagrad	0.02	0.603000	0.874293

Working side-by-side with our Keras implementation, the DNN Classifier which is shipped in-the-box with tensorflow was used mainly for reference and making sure we were on the right way. And although its limitations, the premade model has proved to be very competent in our case; even without any tweaking on short runs we would get almost 80% accuracy using a simple [100, 75, 50, 25] layers setup. We assume that the initialization of the model and the way it handles hyper-parameters behind the scenes is the main reason behind such positive results, showing again how far neural networks advanced in the last decade. The level of adjustments which are covered in the Estimator API is not comparable with what we could achieve with Keras, letting us tweak just a few hyper-parameter at a time. Nonetheless, just like in the previous case, the number and form of the hidden units had a big influence on the end results and we went on to try different combinations, some which followed the standard norm [**?**], and some which were chosen by intuition.

Hidden units	Accuracy
[100, 80, 60, 40, 20]	0.88659376
[100, 75, 50, 25]	0.8824375
[100, 80, 70, 50, 40, 20 ,10]	0.7976875
[100, 90, 70, 90, 75, 50, 35, 15]	0.882375
[100, 50, 10]	0.7654
[100, 85, 70, 55, 40, 25, 10]	0.8819375
[100, 75, 50, 90, 70, 90, 75, 50, 50, 25, 10]	0.72893
[100, 75, 20]	0.73546875

Finally we we're able to take our three best configurations and let them run on longer runs resulting in 88 percent accuracy. We were glad to find that those were very similar to our final Keras outcomes, which supports our assumption concerning the initializing of the premade model and the validity of the first model's results.

5. CONCLUSIONS

To sum up, we started our research in bid to infer accounts basing our ground-truth on the list used by the US congress in their investigation into this case. With the help of other researchers who collected the corresponding data, we were able to construct a sizable amount of examples even compared to other publications on this matter. After we cleaned it up and processed that data using a widely accepted method of vectorizing text, we were left with 400 thousand of unique examples. By splitting our approach into two; the pre-determined models and a neural network, we were able to compare our results and support their validity. By getting rid of the meta-tags and proceeding only with the content of the posts we can say with confidence that our results are as general as possible without being too vague or dismissive, even compared to similar publications on the subject. As we cared about the interpretability of the data, we used predetermined models to help in understanding the shape of the data. We trained using KNN, Logistic regression, LDA, QDA, Splines, SVM and random forests. QDA achieved the best results with 78.7% accuracy indicating that the data is quadratically separable. Increasing the model degree beyond quadratic did not enhance the accuracy which can indicate a possible overfitting.

To handle overfitting and increase the interpretability, we tried to represent the data using less features by introducing PCA but it did not enhance the accuracy.

In the second part we approached the problem using one of the most used frameworks for neural networks - Tensorflow, showing that this result is easily reconstructed for inferring similar state-sponsored agents and further adjusted to a wide array of settings. Sectioning our work again into two, we implemented our main model with Keras while the Tensforflow's Estimator API model was used as a point of reference. Besides the convenience of accessing more configuration parameters, Keras let us run completely different setups more easily. Lastly, we were pleased to find that both of our models produced similar results with 88% being the

best one yet, even compared to the predetermined models. As time goes by and neural networks will become even more widely used, we foresee many improvements in the generation process and the implementation of such filters in our social networks. Considering the magnitude of variables and the fact that the users we were trying to discern were actively trying to blend into normal discussion, a result of almost 90 percent is far from the results we saw in similar research[13], [19]. We believe that our results can be improved even further by continuing the work that Ben Popken[2] has started to collect even more examples for our models to train from. Furthermore we would like to try other means of data pre-processing to see if we can adjust the data to be even more representative of our final goal. Taking our approach as a proof-of-concept this same technique could be adjusted to discerning other binary classes for example Republican and Democratic users. Furthermore we can foresee this being used to infer other sets of features of users; such as niche communities or acute personality traits. This in turn can be used to filter and monitor users on social networks for ab-normal activity; for example deducing a suicidal tendency in a user; filtering his content from suicide related content and forwarding him to the corresponding professional help.

Although the list of 2975 accounts we took as ground-truth was used in US court as evidence, we can't avoid asking how exactly those users were chosen from the 69 million active US twitter users. Unfortunately this list was given as-is and is mentioned only once in the published hearing without much explanation as to how the filtering was conducted. Assuming that the list was conducted based on a list of features, it could mean that our models are over-fitted to this exact list. To check this we would have to conduct a new research into a similar instance of social opinion manipulation, for example the Venezuelan elections of 2018 [1].

Nevertheless, we are satisfied with our results and we believe our model can be used as a proof-of-concept for bot classification using only tweet vectorization without adding any possible biases.

References

[1] Roger D. Harris, The U.S. is Meddling in Venezuelan Election, https://consortiumnews.com/2018/05/18/us-not-sitting-idly-by-on-eve-of-venezuelan-election/ ,May 18, 2018

[2] Ben Popken, Twitter deleted 200,000 Russian troll tweets. Read them here, https://www.nbcnews.com/tech/social-media/now-available-more-200-000-deleted-russian-troll-tweets-n844731, Feb.14.2018

[3] US Congress, exhibit_b; names released by the Congress, https://democrats-intelligence.house.gov/uploadedfiles/exhibit_b.pdf, Oct.24.2017

[4] Ben Popken, Russian Troll Tweets, https://www.kaggle.com/vikasg/russian-troll-tweets/home, March 2018

[5] Ed King, Tweets scraped from Twitter on November 8 2016, https://www.kaggle.com/kinguistics/election-day-tweets, March 2018

[6] https://twitter.com/verified, March 2018

[7] Number of monthly active Twitter users in the United States from 1st quarter 2010 to 1st quarter 2018 (in millions), https://www.statista.com/statistics/274564/monthly-active-twitter-users-in-the-united-states/, March 2018

[8] https://www.tensorflow.org/api_docs/python/tf/estimator/DNNClassifier, March 2018

[9] https://www.tensorflow.org, March 2018

[10] https://keras.io/, March 2018

[11] Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale, Orlando De JessNeural Network Design 2nd Edition, March 2018

[12] Analyzing the Digital Traces of Political Manipulation: The 2016 Russian Interference Twitter Campaign, Badawy, Adam and Ferrara, Emilio and Lerman, Kristina, arXiv preprint arXiv:1802.04291, 2018

[13] Disinformation Warfare: Understanding State-Sponsored Trolls on Twitter and Their Influence on the Web, Zannettou, Savvas and Caulfield, Tristan and De Cristofaro, Emiliano and Sirivianos, Michael and Stringhini, Gianluca and Blackburn, Jeremy, arXiv preprint arXiv:1801.09288, 2018

[14] https://keras.io/optimizers/

[15] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV

[16] http://scikit-learn.org/stable/modules/cross_validation.html

[17] https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw#1097

[18] United States House of Representatives Permanent Select Committee on Intelligence, Testimony of Sean J. Edgett, Acting General Counsel, Twitter, Inc. https://intelligence.house.gov/uploadedfiles/prepared_testimony_of_sean_j._edgett_from_twitter.pdf November 1, 2017

[19] Leo G. Stewart, Ahmer Arif, Kate Starbird, Examining Trolls and Polarization with a Retweet Network https://faculty.washington.edu/kstarbi/examining-trolls-polarization.pdf