CS230

A Deep Learning Approach to Identifying Covert Disinformation Networks

Joshua M. DuFault
Department of Computer Science
Stanford University
JDuFault@stanford.edu

Abstract

Covert and organized networks can use social media platforms to launch large-scale disinformation campaigns and spread their influence undetected. This paper proposes a data organization scheme and convolutional neural network (CNN) approach to identifying these networks quickly and with high accuracy.

1 Introduction

Social media networks allow a single person to connect to millions of users instantly. While this has democratized communication to some extent, it has created unique opportunities for organizations to exert undue influence by spreading disinformation. In February of 2018, the Internet Research Agency (IRA) LLC and several of its members were charged with conspiracy to defraud the United States (US) for using "fictitious US personas on social media platforms" at the direction of a foreign government to influence US elections [1]. This was not an isolated incident, as researchers commissioned in part by the US State Department and Google LLC, reported that 22 out of 60 nations examined were found to employ covert commentators on social media [2].

In this paper, I examine the IRA's messages on Twitter to develop a deep learning model to identify user accounts controlled by an organized disinformation campaign. Here, I present a neural network architecture that accurately classifies Twitter accounts as members of the IRA when compared against non-IRA users that tweet similar content. The input is a collection of sequential tweets from a user in a character vector, which are then organized into groups. The model performs character embedding and uses a CNN to output the probability that the tweet collection is from a user account controlled by the target network.

2 Related work

This appears to be a mostly unexplored topic and a Google Scholar search for "'neural network'AND 'Internet Research Agency'" yielded no relevant articles at the time of this writing. Teams competing in a Darpa contest found that machine learning techniques alone were not able to effectively classify bots on Twitter [3]. However, this contest involved detecting individual bots rather than a connected network.

Tweets have information at the character level. For example, a given word may have a different meaning than the same word with a hash tag in front of it. There are several papers exploring language modeling and binary classification at the character level. Two types of networks gave the highest performance: a combined network with a CNN layer that fed into a LSTM layer [4], and a multiplicative LSTM [5]. There has also been work exploring the use of deep CNNs at binary classification of individual tweets [6].

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

None of these approaches directly reflected using a collection of tweets to classify a user as a member of a network. Therefore, I tested variations of all three models as well as the number of tweets from an individual user needed for accurate classification.

3 Dataset and Features

The dataset consisted of three collection of tweets along with underlying meta-data. NBC News [7] compiled the positive examples from the list of user-names released by the U.S. Congress as belonging to the IRA [8]. These tweets spanned 2014 to 2017 and were largely political. This is the Positive dataset.

I stripped all non-ASCII characters from the tweets to prevent the model from learning based on artifacts that appeared in the data during the compilation process. I then organized the dataset into groups of n sequential tweets from each user, where the value of n was determined experimentally. Finally, I added section tags for the time and text, padded the groups to a constant length, and converted the characters into a four feature embedding. Four features was the smallest embedding possible without affecting accuracy.

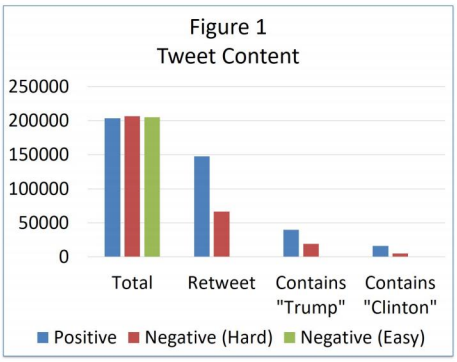
Formatting for tweet group (n = 2) before character embedding:

```
<S> <ST>11:10<ET><SW>Tweet text<EW> <ST>11:15<ET><SW>Next tweet<EW> <E>>>>>
```

The first set of negative examples is a random collection of 200,000 tweets geolocated in the US in 2016 and compiled by Twitter analytics group Followthehashtag [9]. These were on largely different topics than the Positive dataset. This will be referred to as the Easy dataset for the rest of this report.

To be useful as a detection tool, the model must be able to tell the difference between professional members of a network and other Twitter users who are not members but tweet about similar topics. For example, it should correctly identify a member of an organization created to distribute fake news from a non-member who is merely repeating the same fake news accidentally.

To test this, I designed the second set of negative examples to be largely similar in content and user makeup to the Positive dataset. Data analytics company Kaggle [10] had compiled a collection of tweets taken during the 2016 US Presidential Election. I used this dataset to select a group of non-IRA users who tweeted about political topics the same median number of times during the election as the Positive dataset. I then used this collection of users to create a dataset of tweets with approximately the same number of users and tweets per user distribution as the Positive dataset. See figure 1 for a comparison of tweet content and user distribution between the datasets.



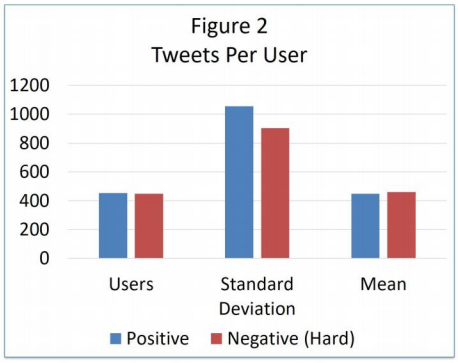


Figure 1: Dataset Comparison

The most effective value for tweets per group proved to be 20, with higher and lower numbers resulting in less accuracy. The total dataset size was 400,000 tweets, which resulted in 20,000 examples each with approximately 2,000 features. Because of the modest number of examples, I chose a 80/20 training/test split.

4 Methods

I tested five different network models. The loss function for all was binary cross entropy with Adam optimization and dropout regularization.

$$\text{Cost function: } -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

4.1 Easy Dataset

I tested five variations of the models identified in the related works section on the Easy dataset for training speed and test set accuracy.

4.1.1 Deep CNN

This featured two CNN layers each followed by a 1D max-pooling layer of four. Each CNN layer used filters of size 16 to identify relevant features in the character vector and a RELU activation function. The filter size was chosen based on the typical size of a long word in the tweets. This was to ensure that the characters could be easily mapped to words. The max-pooling layers reduced the dimensions of the large input character vectors while identifying only the most significant features in a convolved section. It was then followed by an additional max-pooling layer and a fully-connected layer with a sigmoid activation function.

4.1.2 LSTM

This network consisted of a single layer LSTM with an equal number of nodes to embedded character vectors. Each node took input from both its own character vector and the previous node. A tanh activation function was then applied to the output of each node. Each node in the LSTM was then fed into a fully-connected layer with a sigmoid activation function to compute the final probability. A bidirectional LSTM was also tested but abandoned because there was no significant improvement in accuracy but a significant increase in training time.

4.1.3 Multiplicative LSTM

The multiplicative LSTM as described by Krause, Murray, and Renals [5] combines the features of a multiplicative RNN with an LSTM using the following equations:

$$\begin{aligned} m_t &= (W_{mx}x_t) \odot (W_{mh}h_{t-1}) \\ \hat{h}_t &= W_{hx}x_t + W_{hm}m_t \\ i_t &= \sigma(W_{ix}x_t + W_{im}m_t) \\ o_t &= \sigma(W_{ox}x_t + W_{om}m_t) \\ f_t &= \sigma(W_{fx}x_t + W_{fm}m_t). \end{aligned} \quad [5]$$

As with the LSTM, each node fed into a fully-connected layer with a sigmoid activation function.

4.1.4 CNN + LSTM

This featured a single layer CNN with each layer followed by a 1D max-pooling layer and an LSTM. The LSTM then fed into a fully-connected layer with a sigmoid activation function. It was based on the model identified by Jozefowicz, Rafal, et al [4] of Google Brain. See figure 2 for a diagram.

4.1.5 Deep CNN + LSTM

This model was the same as the CNN + LSTM model except with a three layer CNN.

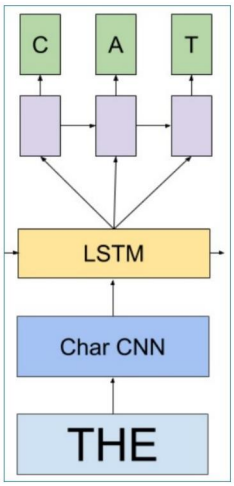


Figure 2: CNN + LSTM Diagram [4]

4.2 Hard Dataset

Following performance testing on the Easy dataset, I chose only the Deep CNN for testing on the Hard dataset and adjusted hyper-parameters to maximize performance. In testing, I found that a three layer CNN resulted in the highest performance. See figure 3 for a diagram.

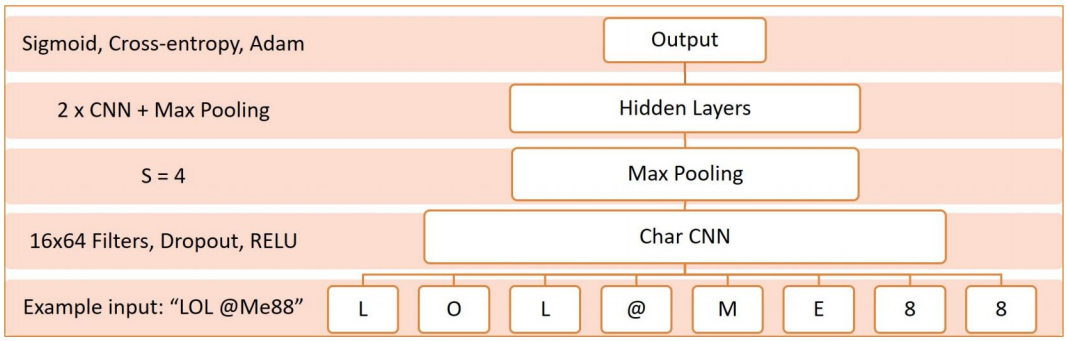


Figure 3: Deep CNN Diagram

5 Results

5.1 Easy Dataset

The LSTM and multiplicative LSTM models both did no better than chance as did the combined CNN + LSTM. I believe this was because the long character vectors made it difficult for the gradients to propagate through the time-steps correctly. I found the best results for groups of 20 tweets which resulted in vectors of approximately 2,000. This would explain why the results improved when a sufficiently deep CNN was followed by an LSTM: three max pooling layers with pool size four was enough to provide a more reasonably sized set of vectors to the LSTM.

However, this still was not enough to match the very high performance of the three-layer CNN. The CNN was significantly faster than the LSTM based models as well. The potential loss of semantic information with a CNN versus a sequential model proved to not be an issue. See table 1 for results.

Table 1: Results for each model (Easy dataset)

Model	Test accuracy	Training time (s)
LSTM	0.497	2764
mLSTM	0.497	4086
CNN	**0.976**	**60**
CNN + LSTM	0.503	698
Deep CNN + LSTM	0.806	48

5.2 Hard Dataset

Initially, I used the same hyper-parameters on the Hard dataset as I used on the Easy dataset. This resulted in an accuracy of approximately 0.80. This indicates that the Hard dataset was significantly more difficult than the Easy dataset and that the selection of the Hard dataset was successful in creating a tweet group similar to the Positive dataset.

By increasing the number of layers and epochs, I was able to increase the performance of the Deep CNN to 0.955. The training set was shuffled randomly on each run and accuracy figures changed by up to 0.02 per run based on shuffling. See table 2 for results.

Table 2: Final deep CNN results

Dataset	Training Accuracy	Test Accuracy
Easy	0.996	0.995
Hard	0.988	0.955

6 Discussion

According to Twitter [7] the number of accounts run by the IRA was on the order of thousands while the number of users who interacted with IRA accounts without realizing they were fake was on the order of millions. This would indicate that human level performance at detecting this particular network is fairly low and that this model's performance in the mid to high 90s may exceed human capabilities.

Sequential models are typically thought to be better at textual analysis because they keep semantic meaning. However, when identifying a network, semantic meaning can be a negative. For example, tweets from the creators of a fake news article and from those who unknowingly redistribute the same article will have a similar semantic meaning. The CNN proved adapt at detecting only the members of the IRA and not others with similar tweet content.

Using a character vector made it significantly easier to keep track of data that would be difficult to make a dictionary out of such as all the possible hashtags, usernames, and links that can appear in tweets. This model also has the advantage that it can be applied to any language with few adjustments.

7 Future Work

The model was designed to be mostly independent of the type of disinformation spread and the types of fake identities a covert disinformation network may use. Ideally, this means it can be applied to any such network. Currently, I have only tested it against the IRA's fake news network. I would like to try the model against other similar networks next.

References

[1] 18-cr-00032-DLF. United States District Court. 22 Feb. 2018. *Internet Research Agency Indictment.* United States Department of Justice, n.d. Web. 15 Apr. 2018.

[2] House, Freedom. "Freedom on the Net." (2018).

[3] Subrahmanian, V. S., et al. "The DARPA Twitter bot challenge." *Computer* 49.6 (2016): 38-46.

[4] Jozefowicz, Rafal, et al. "Exploring the limits of language modeling." *arXiv preprint arXiv:1602.02410* (2016).

[5] Krause, Ben, et al. "Multiplicative LSTM for sequence modelling." *arXiv preprint arXiv:1609.07959* (2016).

[6] dos Santos, Cicero, and Maira Gatti. "Deep convolutional neural networks for sentiment analysis of short texts." *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers.* 2014.

[7] Popken, Ben. "Twitter Deleted Russian Troll Tweets. So We Published More than 200,000 of Them." *NBCNews.com.* NBCUniversal News Group, 14 Feb. 2018. Web. 14 Apr. 2018.

[8] exhibit b. *United States House of Representatives Permanent Select Committee on Intelligence.* 2017.

[9] "USA - Free Twitter Dataset. 300,000 Free USA Tweets." Followthehashtag. *DNOiSE.* Web. 2 May 2018.

[10] King, Ed. "Election Day Tweets." Countries of the World. *Kaggle.* 26 Nov. 2016. Web. 10 May 2018.

[11] titu1994. "Keras-Multiplicative-LSTM." *GitHub repository.* 3 Nov. 2017. Web. 15 May 2018.

[12] keras-team. "examples." *GitHub repository.* 26 Mar. 2017. Web. 12 May 2018.