Fake News Detection Using Recurrent Neural Network

Latifah Alhaura
Faculty of Computer Science
University of Indonesia
Indonesia
latifah.alhaura@ui.ac.id

Mohamad Ivan Fanany
Faculty of Computer Science
University of Indonesia
Indonesia
ivan@cs.ui.ac.id

***Abstract*— Detecting fake news is very difficult. To overcome this problem, we propose a model that detects false information and news with the help of Deep Learning and Natural Language Processing (NLP). In this work, we have successfully used the two Recurrent Neural Networks (RNN) architecture: Long Short Term Memory (LSTM) and Grated Recurrent Units for classification. We create four models and evaluation of the model shows that an accuracy score of up to 96 percent can be achieved using the LSTM model. The results show that the proposed framework is able to present accurate results.**

Keywords—Fake News, Depp Learning, NLP, RNN, LSTM, GRU

I. INTRODUCTION

The information we obtain from the Internet and the Web is not reliable. Over the last few years spreading rumors and false information have reached a tipping point and have begun to influence social and political problems as well. The amount of time spent on social media platforms and online news sites has increased at an alarming rate. Thus most of the information they have is obtained from these sources. Even though social media can be accessed from anywhere and at any time and it's free but it gives anonymity while expressing our opinion, therefore, it leads to a lack of accountability which greatly reduces the authenticity of the data received from them compared to newspapers or other trusted news sources. The lack of continual supervision and regulatory authority has allowed criminals to rampage and spread false information. Fake news is deliberately falsified news sent to deceive people and make them believe in wrong information [1].

The most difficult part comes in the term that the human eye and understanding cannot clearly distinguish between real and false news because they are usually intertwined. One study found that 75% of the time people found fake news rather or very accurate. The circulation of fake news on social media, lately, has been devastated by the continuing government advertising and growing concern about the spread and negative effects of fake news. In the previous year, in January 2017, a German official even accepted the dilemma and cited the spread of false news. They stated that "they are dealing with dimensional phenomena that have never been seen before" in reference to the spread of false news [19]. So, detecting fake news is the most important part that we need to do to sort life [2].

In this work, we have proposed a model for distinguishing between real and fake news articles. The dataset was collected from original sources and used for training and testing purposes. The Long Short-Term Memory Model and the Gated Recurrent Unit model are used for classification purposes. Data is taken from a trusted site[1], and is pre-processed first using Natural Language Processing. NLTK is used to removing stop-words and lemmatization. The Word Index from the tokenized dataset is proposed as the initial weighting for ANN. We have used RNN to create and update this neural network. After training NN, we will have word embedding for all n-grams given in the training dataset. The description of the loss was also carried out. Basic losses are based on Binary Cross-Entropy Loss. This is reduced using RMSprop Optimizer. The model presents accuracy up to 96%.

II. RELATED WORKS

The issue of detection of fake news has become a topic that has emerged in recent social media studies. The existing fake news detection approach generally falls into two categories: using news content and using social context [3]. For news content-based approaches, linguistic features or visual features are extracted. Linguistic features, such as lexical and syntactic features, capture specific writing styles and sensational headings that usually appear in fake news content [4], while visual features are used to identify fake images that were intentionally created or to capture certain characteristics for images in fake news [5].

Features extracted from user posts represent user social responses, such as easel [6]. Network features are extracted by building certain social networks, such as diffusion networks [7] or co-occurrence networks [8]. The social context model can be categorized as attitude based or propagation-based. The attitude-based model utilizes the user's opinion of the news to deduce the truth of the news [6], while the propagation-based model applies a propagation method to model the unique patterns of information dissemination [9].

III. METHODOLOGY

In this section, we elaborate step by step the process of fake news detection using RNN. We create four scenarios to build four models which are:

- Create model using raw text data and RNN-LSTM method
- Create model using filtered text data and RNN-LSTM method
- Create model using raw text data and RNN-GRU method
- Create model using filtered text data and RNN-GRU method

[1] https://www.kaggle.com/c/fake-news/data

A. *Preprocessing Data*

The preprocessing data is done to get filtered text. First, the text converted into lower-case form. Next, we applied Natural Language Processing to remove stop-words: generally the most common words in a language, and lemmatization. We used NLTK Toolkit [2] to do the preprocessing step.

As seen in Figure 1, after preprocessing, we tokenized and padded filtered text to fit into RNN architecture. We implemented Keras[3] framework to tokenize and padding in this step.

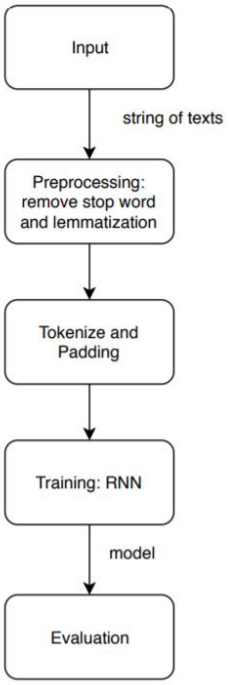

Fig. 1. Diagram of Proposed Framework

Both unfiltered and filtered dataset then split into training and testing data with the composition of testing data is 20% of the data.

B. *Training*

As described above, there are four scenarios that will produce four models. The first two scenario implements LSTM architecture and the last two scenario implements GRU. In this study, we try to compare the result from both method.

Layer (type)	Output Shape	Param #
inputs (InputLayer)	(None, 150)	0
embedding_5 (Embedding)	(None, 150, 50)	11902600
lstm_5 (LSTM)	(None, 64)	29440
FC1 (Dense)	(None, 256)	16640
activation_9 (Activation)	(None, 256)	0
dropout_5 (Dropout)	(None, 256)	0
out_layer (Dense)	(None, 1)	257
activation_10 (Activation)	(None, 1)	0

Total params: 11,948,937
Trainable params: 11,948,937
Non-trainable params: 0

Fig. 2. The First Model Using LSTM and Unfiltered Text as Input

On the first scenario, we used unfiltered text as input. There are 238.052 vocabulary in the unfiltered text. The second scenario used filtered text as input to the RNN architecture. Here we used 235.450 vocabulary. Both scenario used LSTM architecture to train the data. These two scenarios can be seen in figure 2 and 3.

Layer (type)	Output Shape	Param #
inputs (InputLayer)	(None, 150)	0
embedding_6 (Embedding)	(None, 150, 50)	11772500
lstm_6 (LSTM)	(None, 64)	29440
FC1 (Dense)	(None, 256)	16640
activation_11 (Activation)	(None, 256)	0
dropout_6 (Dropout)	(None, 256)	0
out_layer (Dense)	(None, 1)	257
activation_12 (Activation)	(None, 1)	0

Total params: 11,818,837
Trainable params: 11,818,837
Non-trainable params: 0

Fig. 3. The Second Model Using LSTM and Filtered Text as Input

The third and fourth scenario used GRU architecture to train the unfiltered text and filtered text respectively. These two model can be seen in figure 4 and 5.

[2] https://www.nltk.org/
[3] https://keras.io/

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputs (InputLayer)          (None, 150)               0
_________________________________________________________________
embedding_8 (Embedding)      (None, 150, 50)           11902600
_________________________________________________________________
gru_1 (GRU)                  (None, 64)                22080
_________________________________________________________________
FC1 (Dense)                  (None, 256)               16640
_________________________________________________________________
activation_13 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
out_layer (Dense)            (None, 1)                 257
_________________________________________________________________
activation_14 (Activation)   (None, 1)                 0
=================================================================
Total params: 11,941,577
Trainable params: 11,941,577
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4. The Third Model Using GRU and Unfiltered Text as Input

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputs (InputLayer)          (None, 150)               0
_________________________________________________________________
embedding_9 (Embedding)      (None, 150, 50)           11772500
_________________________________________________________________
gru_2 (GRU)                  (None, 64)                22080
_________________________________________________________________
FC1 (Dense)                  (None, 256)               16640
_________________________________________________________________
activation_15 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
out_layer (Dense)            (None, 1)                 257
_________________________________________________________________
activation_16 (Activation)   (None, 1)                 0
=================================================================
Total params: 11,811,477
Trainable params: 11,811,477
Non-trainable params: 0
_________________________________________________________________
```

Fig. 5. The fourt Model Using GRU and Filtered Text as Input

C. Evaluation

For the evaluation, we had eight scenarios to test the four models that were built which are:

- Evaluation for Model-1
  - Using raw test data
  - Using filtered test data
- Evaluation for Model-2
  - Using raw test data
  - Using filtered test data
- Evaluation for Model-3
  - Using raw test data
  - Using filtered test data
- Evaluation for Model-4
  - Using raw test data
  - Using filtered test data

The result of each evaluation will be discussed in section IV.

IV. Result and Analysis

The result of all the models that were built before are shown in Table 1. For the description of four models and the scenario of eight evaluations for each model, are already discussed in section III.

TABLE I. RESULTS

No	Model	Data	Accuracy	Loss
1	Model-1	Raw*	0.92	0.26
2		Filtered	0.96	0.14
3	Model-2	Raw	0.55	1.86
4		Filtered*	0.93	0.24
5	Model-3	Raw*	0.91	0.23
6		Filtered	0.92	0.20
7	Model-4	Raw	0.69	0.82
8		Filtered*	0.90	0.25

*) data used for training and testing

Now let us discuss the result. As explained in section III, Model-1 was trained using raw or unfiltered data. But the result surprisingly showed that using filtered data got higher accuracy than using the raw test data. This went the same when the result of evaluation for Model-3 showed using filtered data for testing can result higher accuracy than using raw data.

Contrary from the two models above, where both used raw data for training, the Model-2 and Model-4 used filtered data for training. But, when we test both model using raw data, the results were really bad as seen in Table 1.

Here, we can conclude that applying preprocessing on the data before training can make higher accuracy than the data that is not preprocessed. Removing stop words and apply lemmatization on the data can help improve the accuracy of the result to classify reliable and un-reliable news accurately.

V. Conclusion

In this study, we have implemented Recurrent Neural Network for classifying fake news. We create four models by implementing LSTM and GRU for learning process and create eight scenarios for the evaluation of each model. The results show that filtering or preprocessing data before learning can help improve accuracy. There are many ways to preprocess data like we did in this study: removing stop-words and apply lemmatization. For further study, we can apply word embedding as one of preprocessing step before do the training.

References

[1] S. Yang, K. Shu, S. Wang, R. Gu, F. Wu, and H. Liu, "Unsupervised Fake News Detection on Social Media: A Generative Approach," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 5644–5651, 2019.

[2] A. Verma, V. Mittal, and S. Dawn, "FIND: Fake Information and News Detections using Deep Learning," *2019 Twelfth Int. Conf. Contemp. Comput.*, pp. 1–7, 2019.

[3] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake News Detection on Social Media: {A} Data Mining Perspective," *CoRR*,

vol. abs/1708.01967, 2017.

[4] M. Potthast, J. Kiesel, K. Reinartz, J. Bevendorff, and B. Stein, “A Stylometric Inquiry into Hyperpartisan and Fake News,” *CoRR*, vol. abs/1702.05638, 2017.

[5] A. Gupta, H. Lamba, P. Kumaraguru, and A. Joshi, “Faking sandy: Characterizing and identifying fake images on twitter during hurricane sandy,” *WWW 2013 Companion - Proc. 22nd Int. Conf. World Wide Web*, pp. 729–736, 2013.

[6] Z. Jin, J. Cao, Y. Jiang, and Y. Zhang, “News Credibility Evaluation on Microblog with a Hierarchical Propagation Model,” in *2014 IEEE International Conference on Data Mining*, 2014, pp. 230–239.

[7] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, “Prominent Features of Rumor Propagation in Online Social Media,” in *2013 IEEE 13th International Conference on Data Mining*, 2013, pp. 1103–1108.

[8] N. Ruchansky, S. Seo, and Y. Liu, “{CSI:} {A} Hybrid Deep Model for Fake News,” *CoRR*, vol. abs/1703.06959, 2017.

[9] Z. Jin, J. Cao, Y. Zhang, and J. Luo, “News verification by exploiting conflicting social viewpoints in microblogs,” *30th AAAI Conf. Artif. Intell. AAAI 2016*, pp. 2972–2978, 2016.