Fake news detection using Deep Learning

Applied Deep Learning
Aleksandar Jankovic
Matr.: 01636308

January 20, 2020

Abstract

Fake news have become one of the major concerns because its potential to destabilize governments, which makes them a potential danger to modern society. These fake news are usually released to either smear the opponents or support the candidate on their side. The erroneous information in the fake news is usually written to motivate the voters' irrational emotion and enthusiasm. Such kinds of fake news sometimes can bring about devastating effects, and an important goal in improving the credibility of online social networks is to identify the fake news timely. This report summarizes my work done on the topic of Fake news detection as a part of the course Applied Deep Learning. The CNN model proposed by Rodriguez and Iglesias (2019) [1] is successfully re-implemented, confirming the results reported. Based on this model a new CNN model was designed, trained and tested to tackle the task of detecting the fake news. The new model is a bit more complex, but it also slightly improves the results achieved with the baseline CNN model.

1 Introduction and Problem Statement

With the growth of online newspapers who update news almost instantly, people have found a better and faster way to be informed of the matter of his/her interest. Nowadays social-networking systems, online news portals, and other online media have become the main sources of news through which interesting and breaking news are shared at a rapid pace. However, many news portals serve special interest by feeding with distorted, partially correct, and sometimes imaginary news that is likely to attract the attention of a target group of people. Fake news has become a major concern for being destructive sometimes spreading confusion and deliberate disinformation among the people. These are published usually with the intent to mislead in order to damage a community or person, create chaos, and gain financially or politically. Since people are often unable to spend enough time to cross-check reference and be sure of the credibility of news, automated detection of fake news is indispensable.

This problem has reached its peak of notoriety after the 2016 US electoral campaign when it has been determined that the fake news have been crucial to polarize society and promote the triumph of Donald Trump. Another example is when on June 23th 2018, a series of horrifying images and videos began to circulate on Facebook. One showed a mans skull hacked open that was viewed more than 11,000 times. The Facebook users who posted the images claimed they showed a massacre underway in the Gashish district of Plateau State, Nigeria by Fulani Muslims who were killing Christians from the regions Berom ethnic minority. Consequently, a massacre did happen in Gashish that weekend and somewhere between 86 and 238 Berom people were killed, according to estimates made by the police and by local community leaders. However, some of the most incendiary images and videos were totally irrelevant to the violence in Gashish. The video showing a man's head was cut, did not even happen in Nigeria and it was recorded in Congo, in 2012.

The above mentioned two fake news incidents are enough evidence to take this problem seriously. The big technology companies like Twitter, Facebook and Google have spotted this danger and have already begun to work on systems to detect fake news on their platforms. Several different approaches and solutions for detecting the fake news have been proposed in recent years. Most of the work in the area is focused on the idea of studying and detecting the hoaxes on its main spreading channel: social media, where, for example, the probability of a given post to be false is studied using its own characteristics such as likes, followers, shares, etc., through classical machine learning methods (classification trees, SVM, ...). Applying these kinds of approximations, the best results were obtained with click-bait news being detected with the 93% accuracy.

Lately, the attention in this area has been given to deep learning. Among other results, for the task of detecting fake news Rodriguez and Iglesias (2019) [1] created an LSTM and a CNN based neural architecture. Furthermore, they adapted and tweaked Google's BERT model. The work of Rodriguez and Iglesias is partly an extension of Yang et al. (2018) [3], which also proposed a CNN based architecture. These two work, were the baseline for my project. More precisely, I focused on their CNN based architectures. Both reported similar results, with slightly better results of the model proposed by Rodriguez and Iglesias, 94% accuracy on the test data (versus 93%).

Another point to make here is that one important limitation of prior comparative studies is that these are carried out on a specific type of dataset. Thus, it is difficult to reach a conclusion about the performance of various models. In my case, the CNN models from [1] and [3] are both trained on the same dataset: TI-CNN, which has been built in [3]. It is composed of 20015 news articles labeled as fake or true which are gathered from two sources. On the one hand, the fake ones come from the dataset Getting Real About Fake News [2] while the real ones have been obtained from well-known sources such as The New York Times or The Washington Post. To hyper-tune the parameters, Rodriguez and Iglesias used Fake News Corpus dataset. This one is composed by millions of online articles classified in several categories like bias, clickbait, fake, political or true. I had the same intention, however due to time constrains, I did not get the chance to hyper-tune my network against this dataset.

2 My Solution

After carefully reading and examining the above mentioned papers, I decided that my goals (checkpoints) for this project are the following:

- Since I had almost no experience with building and designing neural networks, I first had to understand and learn how to do it. This included many tutorials, documentations, github repositories etc.
- The first (real) checkpoint for the project was to re-implement the work done in [1].

The main reason to focus more on [1] (instead of on [3]) is that [1] reports slightly better results. On the plus side, the paper is new and, honestly, their approach seemed a bit more interesting to me.

- I wanted to improve their results. I wanted to (based on the network in [1]) design and build my own network that would probably be more complex, but with better accuracy.

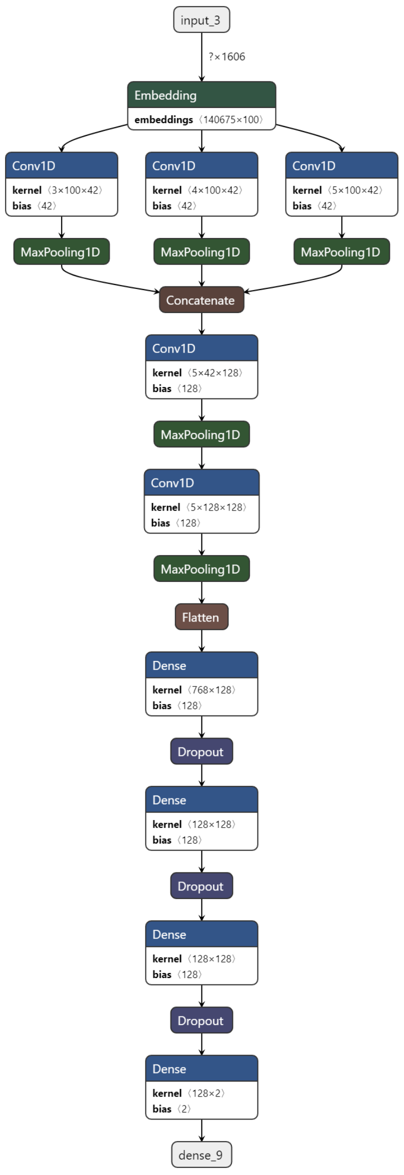

Figure 1: The CNN based architecture created in this project

Additionally, I had to deliver the project in the form of this report, a presentation and a Javascript demo application to show my results.

I successfully accomplished all three of my target goals. After I acquired enough skills, I successfully re-implemented the CNN from [1] well within my estimated time by following the detailed explanations about the architecture, provided in the paper. I also confirmed the results reported. After training the model over 4 epochs, I reached an average accuracy of 93,5% on the test set, with a holdout strategy of 70% of the data for training, 30% for testing and 30% of the training set dedicated to validation during training. The best measured accuracy on that model was at around 95%.

At that point, I had to brainstorm possible changes and improvements. I first tried to tweak some parameters and settings, just to see what will happen. However, this approach did not lead to a significant and consistent accuracy improvement. The results were highly dependent on the randomness of the split and on the size of the splits. The only mention-worthy tweak would be the change in the model optimizer, Adadelta seemed to work a tiny bit better than SGD.

The next attempt was to try to redesign the network to have better accuracy on the test split. Note that the approach in [1] is to learn to distinguish between real and fake news by feeding the network with text (body) and title feature, whereas the CNN in [3] does it with image and title feature. Now, this the first place, where I figured one could make a change. My approach puts the entire weight solely on the text feature. Compared to the architecture in [1] which starts with two branches, a title and a text branch, my network starts with three text branches.

Additionally, after the concatenation of these three branches, I add a Conv1d and a MaxPool Layer, two times. For a complete architecture, please refer to Figure 1. This approach turned out to show slightly better results on the TI-CNN dataset than the baseline model, even with mostly default parameter (and parameters taken from [1]). I again used Adadelta optimizer, which showed better results than the SGD. Also, instead of the "Binary Crossentropy" loss function, I used the "Categorical Crossentropy". My overall conclusion is that this approach improves the results on test data by 1% (achieving 95% on average). One also has to notice the slight increase in complexity between these two models. Last note to make here is that I built, trained and tested all models using Keras in Google Colab environment.

The last step of this project was to deliver i.e. to develop a demo application that runs the trained model in the browser and provides an interface to trigger inference, to prepare a presentation and to write this report. I built a simple Vue.js web application that runs the model in the browser and enables the user to verify the validity of some pre-defined news articles (taken from the dataset). The user can also insert its own articles and verify them. To run the model in the browser I first tried to use the ONNX.js. This did not turn out well. Transforming my Keras model to an ONNX model was easy. However, as it turns out, ONNX.js does not yet support all the operators that ONNX has. In my case, my transformed model used the 'Cast' Operator, which is not supported. This basically meant that there was no way to run my model with ONNX.js. So I switched to Tensorflow.js. Working with Tensorflow.js was a pleasant experience and the model inference worked like a charm.

3 Conclusion and Take-aways

Since I am fairly new to Deep Learning and the Machine Learning field, this project was extremely exciting and fun. I have learned a great deal of new things. I was very skeptical at the begging with the work and time estimate structure, because with almost no previous experience I could not really know what challenges, pitfalls and troubles await me. Interestingly enough, my time estimates for the exercise 2, the report and the presentation were pretty close. I estimated 80 hours for the exercise 2 (including the very needed studying and tutorials). Out of those 80 hours, I actually spent 65. The difference mainly lies in the fact that I needed less time for studying and tutorials to start working than I anticipated (spent 8 hours less than estimated). I estimated 12 hours for this report and the presentation, which is about exactly the time I needed.

Sadly, I totally underestimated the time needed to develop the application. Sure, one reason lies in the fact that the ONNX.js simply did not work with my model. But not only that, I totally failed to foresee all the parts of this task, an example being the need to program the Tokenizer to encode the text input. I estimated a total of 12 hours for the application. It took me about 25-30 hours to finish the work. The most chunks of time went in the research and googling why is my code not working. However, if I were to start the project all over again, there is not really I would want change (except maybe that terrible error with the data split).

References

[1] Álvaro Ibrain Rodríguez and Lara Lloret Iglesias. Fake news detection using deep learning, 2019.

[2] Getting real about fake news. https://www.kaggle.com/mrisdal/fake-news.

[3] Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, and Philip S. Yu. Ti-cnn: Convolutional neural networks for fake news detection, 2018.