Fake News Classification and Visualization Tool

Spencer Pozder[1] - pozder.s@husky.neu.edu

I. Introduction

Recently, the emergence of a large amount of sensationalistic, overly-biased, or even factually incorrect "Fake News" has had a noticeable negative impact on politics and the public's perception of the media. It is capable of spreading hate and misinformation to large groups of people, which can impact their voting patterns or policy backing. Fake science news (or just Junk Science articles) can create health problems by spreading misinformation about medical procedures (such as the spreading anti-vaccination movement). Finally, clickbait and other useless, attention grabbing articles make it harder to parse the internet for other information. There are many reasons to find and avoid such pages, but the increased amount and sophistication of such misinformation is making it harder for web users. This paper introduces a automated tool for identifying fake news, as well as providing users with insight into patterns that indicate each page's classification.

II. Method

The method I employ to actually detect and classify fake news articles is a Convolutional Neural Network. In general, the network uses a series of trained filters to find patterns of words that can determine if an article is fake or not. It does this by applying a series of convolutional filters across the text, where each word is an N-dimensional vector representation pre-trained by Stanford University (called the GloVe representations). After each convolutional layer, a maxpooling layer is employed to reduce noise in the model. In general, these patterns of layers are good at finding features within spatial data (like words in a paragraph) that can be used for classification. These features are then piped through a few dense layers with Relu activation that are trained to learn what these features look like in each case. Eventually, the layers output a one-hot representation of the classification: fake or not-fake. This full model uses the first 128 words of each article, and produces a single classification.

However, this classification is not immediately very useful to an internet user. Without a visualization that interacts with the page itself, the user would need to purposefully use the text classifier with each article they visit, and they wouldn't know what it is in the page that makes the classifier determine its class. Instead, the paper introduces a page-editing visualization, enforced by a browser plugin. The idea is that the user would navigate to a page with a questionable news article, select the text in the article, and feed the text from the page into the neural network implemented in the browser plugin. The plugin then changes the page itself to indicate the classification, as well as portray the significance of certain words towards that classification.

[1]College of Engineering, Department of Electrical and Computer Engineering, Northeastern University, 360 Huntington Ave, Boston, MA 02115

III. Data Source

In order to train my Convolutional Neural Network, I used the text corpus "Fake News Corpus"[4]. This corpus contains about 9.4 million articles from 745 different domains. These articles are then tagged with a series of classifications, which are trained against when developing my classifier. In order to create a usable version of each text, I first clean them (lower-casing everything and removing punctuation and stopwords), then determine the index of each words used in the GloVe embedding. These are then converted to a Tensor, and this tensor is fed into the neural network (either padded with zeros or truncated to reach the correct length). This input stream can be seen in Figure 1.

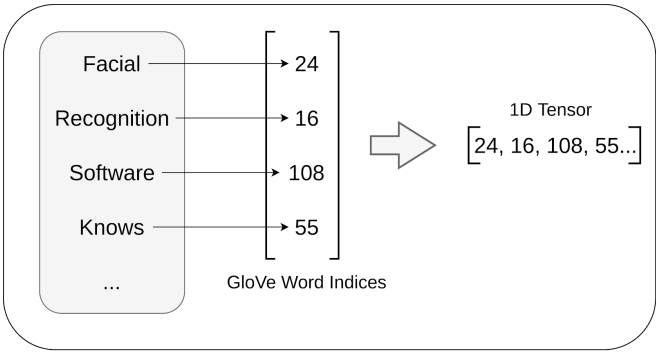

Fig. 1. Article text to tensor input pipeline

IV. Visualization

In order to actually display the important information to the user, I developed a method for quickly measuring each word's importance to the network's classification decision. Instead of just showing the classification on the page, which would appear baseless and not actually useful for avoiding such work in the future, I made a quick method for finding which words have the most attention from the network - similar to the output of neural-network visualization projects like Keras-vis[3]. However, without implementing this entire method, I instead use a sliding-window approach. In general, by removing individual words from the input, and measuring the change in the output classification, I am able to determine the relative importance of each word in the input. From there, I scale the words in the article itself based on this relative importance (most classification-changing words are largest).

In this way, the plugin is able to explain immediately to users what about the article appears fake, as well as drawing their attention to the classification itself with a separate alert. See the page with the visualization applied in Figure 2.

Fig. 2. Part of Fake News page with visualization applied[1]

V. Implementation

The neural network itself is implemented in Tensorflow, using the Keras module. This way, the model can be easily created and trained, as well as easily used in a browser extension utilizing TensorflowJS. There were a few different model architectures attempted: a simple, shallow convolutional network, a deep convolutional network similar to the one presented in[2], as well as a parallel CNN architecture as shown in Figure 3. Each of these models were trained on the full article corpus for a minimum of 4 epochs (around 4 hours per epoch running on a Google Compute instance with a GPU).

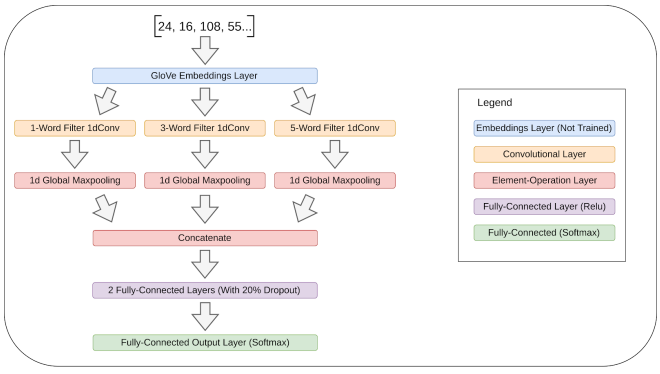

Fig. 3. Parallel Convolutional Neural Network Architecture

The actual browser extension was written in JavaScript for the Chrome browser. This way, it could be easily developed and tested, as there are existing examples of Chrome extensions that utilize TensorflowJS. The Chrome extension itself is written as two separate JavaScript files, one that loads and runs in a background page, and a small one that is added to every page that loads in the browser. This way, the network only has to be loaded into memory once (in the background page), text can be sent to the classifier through the browser's messaging service, and similarly the background page can send the classification and important words to the page containing the article for the JavaScript there to display. The code for this, as well as the code used to train the neural network, is included with this submission.

VI. Analysis

Unfortunately, I was unable to train a network to accurately classify between real and fake news consistently. Each of the three attempted architectures were unable to generalize patterns that indicated news was fake - they all eventually just guessed all news was fake instead. The overall test/validation accuracy for all the models was thus 50%, as each dataset contained approxamtely 50% fake news. This indicates an under-fitting problem, but it is possible that a convolutional network is generally unable to distinguish between the two when trained on this corpus. Given more time, it would be interesting to attempt to build a classifier that uses an RNN instead of a CNN, or to try more CNN architectures or datasets in order to be able to complete this part of the project. However, the browser extension and visualization it creates on each page does appear to be successful. The words are accurately sized based on importance to the classification, and it runs performantly in the browser. The alert is able to grab the user's attention effectively, and the word size changes makes it easy to see why the classification was given, while still allowing the user to read the original article.

VII. Future Work

There is existing software takes a gradient-based approach, measuring the gradient along the last convolutional layer output and following it to the measure each input's importance. However, this method is complicated and not implemented for the JavaScript version of Tensorflow. It would be great to be able to take more time and implement this version of the attention mechanism, which would allow for finding the importance of series of words that are not immediately adjacent. It would also be interesting to introduce automatic parsing to the implementation. In this sense, the classifier could run in the background on every article load and determine a classification for the user/display the visualization, without prompting. This way, users wouldn't have to interact with the plugin to be alert to fake news while browsing, and instead can just use the web as they normally would.

VIII. Conclusions

Overall, the project was successful in creating a visualization tool for news articles that shows whether the article is fake news or not, and why. While the actual classification is not very accurate, the visualization tool itself appears immediately useful and is an exciting example of machine learning's role in combating fake news in the world. The extension introduced is performant enough to run on an everyday-machine, and could even be run on a mobile phone if the extension were written as an app. While there is future work that could be done on the project to improve it, the visualization itself is currently useful and informative, while maintaining page integrity and readability.

REFERENCES

[1] Facial recognition software knows it has seen man before but can't remember his name. https://www.theonion.com/facial-recognition-software-knows-it-has-seen-man-befor-1840033674, Nov 2019.

[2] Francis Fernández Reyes and Suraj Shinde. *Evaluating Deep Neural Networks for Automatic Fake News Detection in Political Domain: 16th Ibero-American Conference on AI, Trujillo, Peru, November 13-16, 2018, Proceedings*, pages 206–216. 11 2018.

[3] Raghavendra Kotikalapudi and contributors. keras-vis. https://github.com/raghakot/keras-vis, 2017.

[4] Maciej Szpakowski. Fakenewscorpus. https://github.com/several27/FakeNewsCorpus, March 2019.