Fake News Detection

Wei Luo
wl2671@columbia.edu

Yanmin Ji
yj2466@columbia.edu

Yuanchu Dang
yd2466@columbia.edu

Abstract

The spreading of rumours, or Fake News, keeps growing at an alarming rate on online social network platforms. With the recent booming development in Machine Learning fields, different models have been designed for human-level accuracy in specific tasks. However, most of them cannot be used for real-time predictions because of their large number of parameters.This paper presents a lightweight approach that has similar accuracy yet much faster predicting speed, making it an outstanding candidate for real-time analysis in big data era.

1. Introduction

With the rapid development of Internet and social media, the sheer volume of news and contents starts to explode in an exponential manner, and so do fake ones. Our final project is focused on the topic of fake news detection, an active area of research where many problems remains to be solved or perfected. There are many important subproblems in this field, such as sentiment analysis, predicting the veracity of news and documents, validating sources against one other, etc. For this project, we choose to narrow down to a specific formulation, that is, given a news headline and a body of paragraph, our objective is to determine whether the body discusses, agrees of disagrees with, or is totally unrelated to the headline text. In terms of modeling, we first implement and configure vanilla feed-forward neural networks and achieve solid benchmark accuracy. On top of that, we further experiment with recurrent neural networks, especially long short-term memory units that are known to perform well with sequential data such as texts. Last but certainly not least, we apply BERT - a latest pretrained fine-tuning language model developed by Google AI - to this classification problem and achieve decent training outcomes. Separately, using Twitter's streaming API and our trained classifiers, we build an interface that validates the truthfulness of tweets in real time based on the predefined ground truth body texts.

2. Related Work

Fake news detection has been a hard problem for several years in the field, mainly because there is no established method for this specific task. Traditional rumour detection is based on the hand-crafted features tuned by researchers. These features often included credibility of Twitter users, the popularity of target tweets and other features except content of tweets [6, 8, 9]. Traditional natural language processing approach also involves using decision tree to analyze different feature sets of Twitter rumors [5].

The recent booming development of deep neural network, one of the most important breakthroughs in the past few years, brought new insights. For example, Ma et al. proposed to use Recurrent Neural Network to model the social context information events [7]. Although the performance of their model was impressive, it is impossible to use it for real-time data analysis because it relies on human-organized tree-structured event datasets. Ferrara et al. achieved 97% accuracy in predicting fake images on Twitter. They used both user-based features, such as age of users, followers'size and the follower-followee ratio, and tweet-based features, such as length, retweet count and the presence of emoticons [4].

In the Fake New Challenge competition, several teams approached the problem similarly. For example, team SOLATE IN THE SWEN used an ensemble of decision tree models and MLP models that are very similar to our implementation [1]. Their main issue is that the hand-crafted features of decision tree is both time-consuming at the training stage and hard to vary after the training. In this project, we developed a simple model that can achieve similar accuracy but is much faster than previous models, making it most suitable for online real-time analysis.

3. System Overview

3.1. Dataset

The dataset we use comes from Stage 1 of the Fake News Challenge. In total, the dataset is comprised of 1683 article bodies, 49972 headlines, and ground truth labels corresponding to each headline-body pair. The size of the joined dataframe is over 100MB. We split the entire dataset into training and validation according to a 85% to 15% ratio.

3.2. System

Our system is concatenated disjointly by a modeling component and a front end Twitter application. The modeling box and its various algorithms will be discussed in great details in the upcoming section, and the Twitter application will be explained in the software package description section.

4. Methods

In this section, we would briefly introduce several Machine Learning models we tried for this specific task and how we designed them.

4.1. Models

In this project, we mainly investigate the effects of deep neural network models partially because previous competition participants have shown that deep neural network models can often outperform rule-based or decision tree models [1]. Also, since the goal of the project is to applying the model in the scale of big data, the massive amount of news data should provide enough training data for the model. Here, we mainly introduce three types of neural networks: Multilayer Perceptron (MLP), Recurrent Neural Network (RNN) and Bidirectional Encoder Representations from Transformers (BERT) with MLP.

4.1.1 Multilayer Perceptron (MLP)

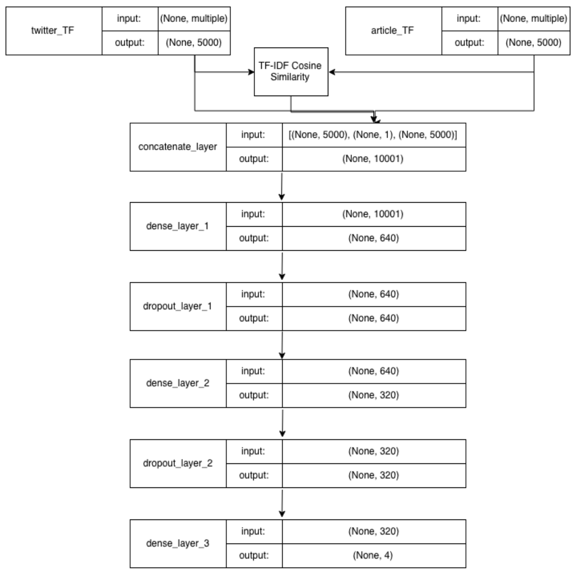


Figure 1. Multilayer Perceptron Structure

MLP is often used in simple tasks such as nonlinear approximation. Although its accuracy is usually lower than complex deep neural networks, it is still popular in certain fields because of its simplicity. MLP normally contains at least three layers: an input layer, a hidden layer and an output layer. Our MLP network structure is implemented as in Fig 1 Since MLP cannot directly learn time series data such as text, we preprocess texts using bag-of-word method. Specifically, we tokenize both tweets and articles together, and convert them into bag-of-word vectors using fixed-size most-commonly-seen tokens in both data. Then, we calculate term frequency vectors for tweets and articles data separately. Term frequencyinverse document frequency (TFIDF) vectors for both data are also calculated. The input layer of MLP takes the concatenation of [TF vector of tweet, cosine similarity of TFIDF vectors, TF vector of article]. The rational behind the concatenation is that while TFIDF calculates content similarity, TF vector can give more weights on similar words. The tensor is then passed through two hidden layers with dropout and a final softmax classifier.

4.1.2 Recurrent Neural Network

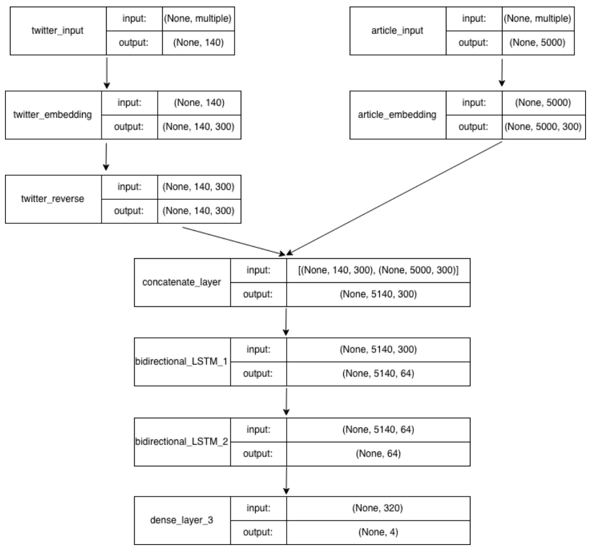


Figure 2. Recurrent Neural Network Structure

Recurrent Neural Network (RNN) is another class of deep neural network where connections between nodes form a directed graph along a sequence. This characteristic leads its popular in applications on time series data, such as text. Unlike feed-forward neural networks, such as MLP mentioned above, RNN can use their internal state to process sequences. Long short-term memory is one type of RNN units that utilizes three gates (an input gate, an output gate and a forget gate) to deal with the vanishing gradient issue of RNNs. Here, we implement two layers of LSTM neural network to detect position-related connections between tweets and articles. The structure of the whole network is shown as in Fig 2.The tweets and articles are tokenized, truncated and zero-padded at the end. The modified tweets and articles are passed through different embedding layers separately. Then, the embedded tweets tensor is flipped along the row direction and concatenated with the embedded article tensor. The rationale behind is the beginning of articles always contains the most important information. By flipping the embedded tweet tensor, tweets and the beginning of articles are placed as close to each other as possible. Therefore, this modification should theoretically boost the performance of our network. Then, the concatenated tensor is passed through two layers of bidirectional LSTM network and a softmax classifier.

4.1.3 BERT

BERT stands for Bidirectional Encoder Representations from Transformers. It is a state-of-the-art language model developed by Google AI Language very recently [3]. BERT was first released in November 2018, and since then has pushed the performance boundaries of various natural language processing tasks. Specifically, BERT increases the GLUE benchmark to 80.4% (7.6% absolute improvement), the MultiNLI accuracy to 86.7 (5.6% absolute improvement) and the SQuAD v1.1 question answering Test F1 to 93.2 (1.5% absolute improvement), outperforming human performance by 2.0%.

On a high level, BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers [3]. The pre-trained BERT representations can be fine-tuned with just very few additional hidden layers to create state-of-the-art models for many generic tasks, such as question answering and language inference, without substantial task-specific architecture modifications. This is in contrast with feature-based pre-trained models that are geared towards very specific language tasks.

BERT is conceptually simple and empirically powerful. Here we display a model that illustrates BERT's bidirectional design. The arrows in the picture indicates that BERT considers both left and right context when making a prediction about the current iteration.

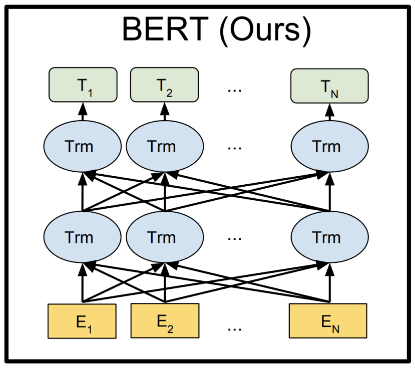


Figure 3. BERT Architecture

One of the major innovation about BERT is its objective function. The authors proposed Mask Language Model (MLM) and Next Sentence Prediction (NSP). Of these two objective functions, MLM aims to predict tokens in a sequence that are randomly masked out, from left and right contexts; and NSP aims to predict if some sentence A naturally follows some sentence B. The second task NSP is particularly relevant to our stance classification from a pair of headline and body. For classification purpose, we can simply use pre-trained BERT embeddings, which consist of three layers of embeddings including a token-level embedding, a sentence-level embedding, and a positional embedding, all concatenated together. This design is shown in the following graph. On top of the aggregate embeddings, we connect the graph to an additional hidden layer and pass the computation results to a vanilla softmax layer with cross-entropy loss.

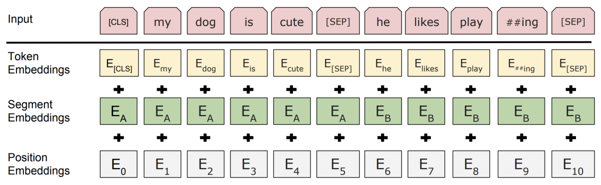


Figure 4. BERT Embedding

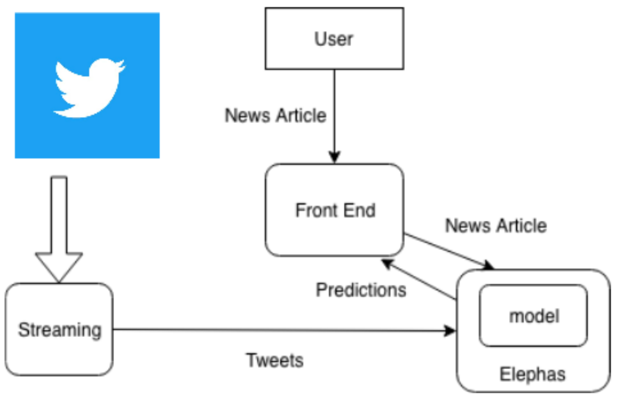
4.2. Tools

For packages and tools, we leverage many of the frontier machine learning modules, including Tensorflow, Keras, Elephas and Gluon. Of these, Tensorflow and Keras are learning packages, used primarily for building neural networks, testing different architectures and hyper-parameters, and training; Elephas is a hybrid module mingling Keras and Spark in a way to bridge the gap between the two so that models can retrieve data stored in a distributed manner; Gluon is a natural language processing package.

5. Software Package Description

Our most up-to-date GitHub repository is being maintained here [2]. Each file in the repository contains self-explanatory comments about its functionality. As our contribution is composed of models and a demo, so is the structure of this repository. In particular, we have folders named feed_forward, bert, data, demo, and trained_models. Inside feed_forward, util.py is responsible for reading and processing data for training and testing. Pred.py contains code for training and predicting using the feed-forward network. The Bert folder contains code for data construction and tokenization (dataset.py, tokenizer.py), constructing and training the model (bert.py), generating predictions (classify.py). The data folder contains training data and ground-truth labels. The model folder contains saved model from feed-forward network. Inside trained_models, there are trained models for feed-forward, GRU, and LSTM, with their corresponding python notebooks. Finally, the demo folder contains code for our twitter-spark demo, which we will discuss in detail here.

For our demo, we use a multi-thread architecture, as depicted in Figure 5.



In our demo, there are two threads at work. First, the background thread uses Tweepy to stream tweets in real-time. We can adjust the specific words or hashtags that we want to track so that we do not get a huge volume of irrelevant text. These tweet texts are constantly being sent over to our trained model in the back end. The second thread is the foreground thread that reads and processes user requests. First, we have a web page that accepts user input for a news body. When the user clicks the button corresponding to submit, the text is sent over to the back end, which then starts its prediction phase using the provided body text and the streamed tweets as headline. For each headline, the model makes a prediction, which may be "agree", "disagree", "discuss", or "unrelated". "Agree" means that the provided body's content agrees with the twitter headline, whereas "disagree" means the body and headline are in direct conflict. "Discuss" means that the model is not sure of the decision and needs further discussion. "Unrelated" means the body and text are not about the same topic. After the model makes each prediction, we update the front-end with a real-time bar chart of the count of each prediction.

As an example, starting with Figure 7.1 in the appendix, suppose the user inputs the text:

> Michael D. Cohen, a former lawyer for President Trump, was sentenced to three years in prison on Wednesday after denouncing Mr. Trump and explaining that I felt it was my duty to cover up his dirty deeds.
>
> Mr. Cohen gave an emotional apology to the court for his involvement in a hush-money scandal that could threaten the Trump presidency a scheme to buy the silence of two women who said they had affairs with Mr. Trump to protect his chances before the 2016 election. Mr. Cohen said his blind loyalty to Mr. Trump led him to ignore my own inner voice and my moral compass. The sentencing in federal court in Manhattan capped a startling fall for Mr. Cohen, 52, who had once hoped to work by Mr. Trumps side in the White House but ended up a central figure in the inquiry into payments to an adult-film star and a former Playboy model before the 2016 election."

After we run the demo for several minutes checking against real-time streaming, the front-end will be able to generate analytics shown in Figure 7.1. Because there are more agrees than disagrees, we can say that news on Twitter about Michael Cohen are more "real news" than "fake news". Also note that there are a lot of unrelated headlines, simply because we are still tracking a wide variety of tweets. Figure 7.1 show sample predictions from our model.

6. Experiment Results

At the current stage, the three-fold validation accuracy of each model we examined is shown in Table 6. It turns out that MLP, the simplest model we examined, has the best performance.

Comparing to the accuracy of RNN model, 0.88, the accuracy of MLP model, 0.93, is 5.6% higher. One possible explanation is that location-related connection between tweets and articles is not strong enough for classifying fake news. In reality, the information contained in tweets is much less than that contained in headlines, which was used as training set. Another possibility is that the size of the dataset might not be large enough for the RNN model to learn. In fact, we did see that the model quickly achieved a high accuracy, but hard to improve further. Therefore, data augmentation using existing model might help for a higher accuracy.

On the other hand, although BERT has proved to have significant boost on multiple natural language processing tasks, the sheer volume of its parameters makes it hard to train. Even with the pre-trained model, the fine-tuning process for the hidden layer and its training processes can be time-consuming. For example, training a single epoch of 4000 batches would take roughly 16 hours. Thus, although we fully implemented the BERT model and deployed it on Google Cloud, we still chose the faster MLP model for the real time Twitter application.

Table 1. Validation Accuracy of Models

Model	Validation Accuracy
MLP	93%
RNN	88%
BERT	70%

7. Conclusion

7.1. Conclusion and Future Work

We propose a lightweight yet accurate machine learning model for detecting fake news based on Twitter contents. We also demonstrate its usage in real-time analysis using the Twitter platform. With the help of Elephas, the model can be easily scalable to accommodate bigger data.

In our future work, we plan to take extra parameters from tweets, such as likes or retweets, as part of input to our network. Although the time-series tree-structured event data is impossible to be generated in real time, it is possible to cache Twitter interactions during a short time period for event tracking. Also, as part of the stance detection limitation, we can only track one article at a time at the current stage. The combination of entity detection and natural language processing could allow the model to detect several events in parallel.

References

[1] Fake news challenge - team solat in the swen. https://github.com/Cisco-Talos/fnc-1. Accessed: 2018-11-20.

[2] Fake news detection. https://github.com/Yuanchu/fakenewschallenge.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, page arXiv:1810.04805, Oct. 2018.

[4] E. Ferrara. Manipulation and abuse on social media by emilio ferrara with ching-man au yeung as coordinator. *ACM SIGWEB Newsletter*, (Spring):4, 2015.

[5] S. Hamidian and M. T. Diab. Rumor detection and classification for twitter data. In *Proceedings of the Fifth International Conference on Social Media Technologies, Communication, and Informatics (SOTICS)*, pages 71–77, 2015.

[6] X. Liu, A. Nourbakhsh, Q. Li, R. Fang, and S. Shah. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1867–1870. ACM, 2015.

[7] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824, 2016.

[8] J. Ma, W. Gao, Z. Wei, Y. Lu, and K.-F. Wong. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1751–1754. ACM, 2015.

[9] F. Yang, Y. Liu, X. Yu, and M. Yang. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM, 2012.

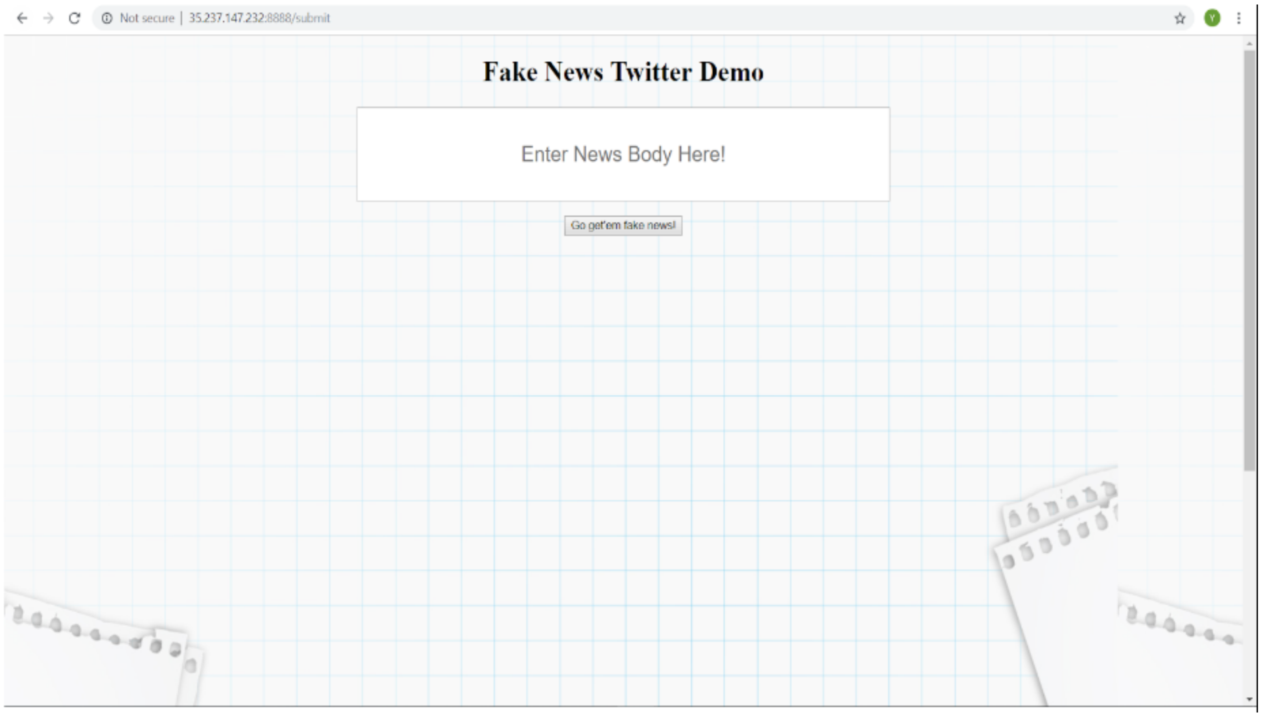


Figure 5.

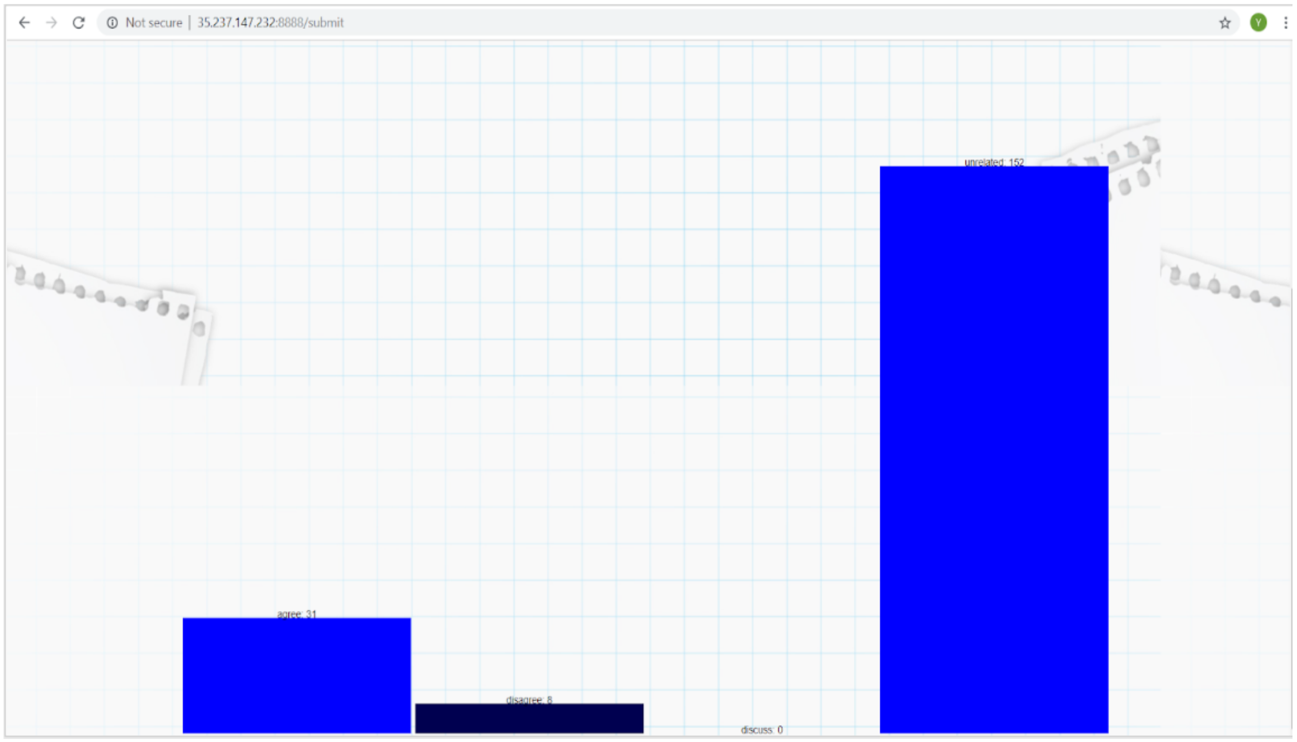
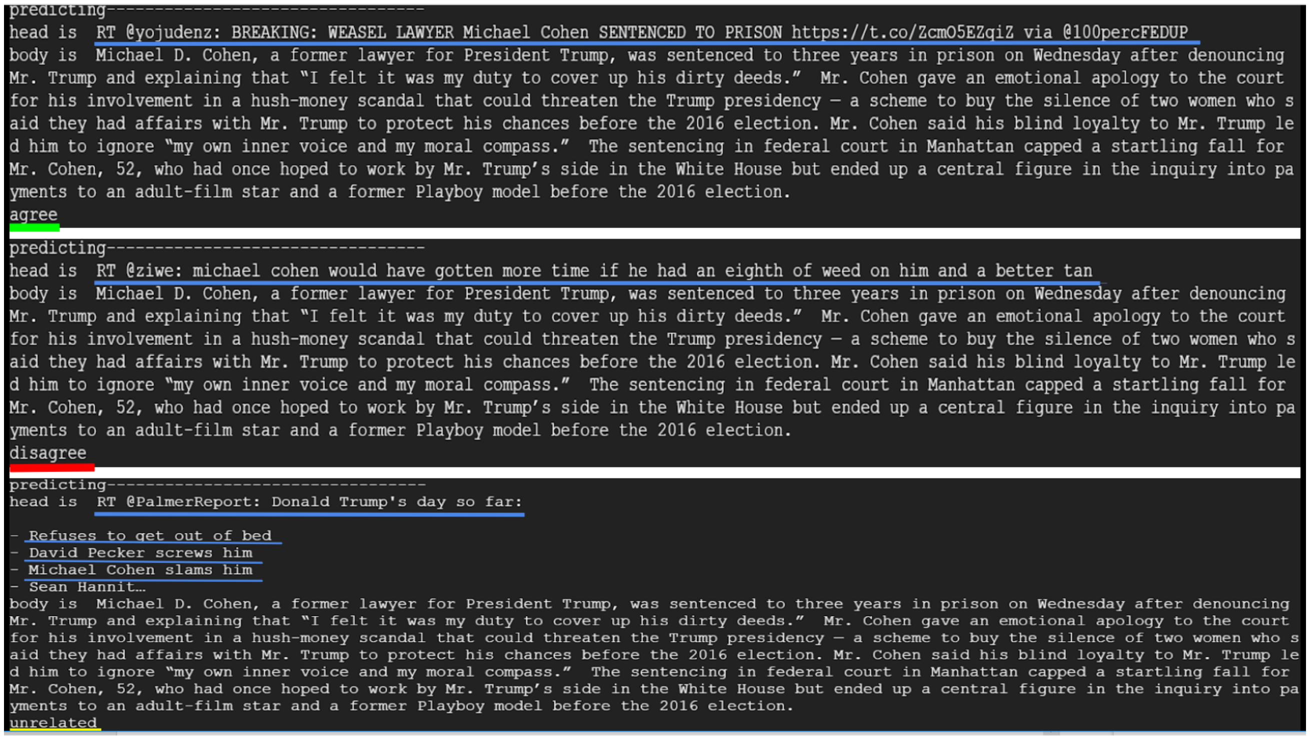


Figure 6.



```
predicting---------------------------------
head is  RT @yojudenz: BREAKING: WEASEL LAWYER Michael Cohen SENTENCED TO PRISON https://t.co/ZcmO5EZqiZ via @100percFEDUP
body is  Michael D. Cohen, a former lawyer for President Trump, was sentenced to three years in prison on Wednesday after denouncing
Mr. Trump and explaining that "I felt it was my duty to cover up his dirty deeds."  Mr. Cohen gave an emotional apology to the court
for his involvement in a hush-money scandal that could threaten the Trump presidency — a scheme to buy the silence of two women who s
aid they had affairs with Mr. Trump to protect his chances before the 2016 election. Mr. Cohen said his blind loyalty to Mr. Trump le
d him to ignore "my own inner voice and my moral compass."  The sentencing in federal court in Manhattan capped a startling fall for
Mr. Cohen, 52, who had once hoped to work by Mr. Trump's side in the White House but ended up a central figure in the inquiry into pa
yments to an adult-film star and a former Playboy model before the 2016 election.
agree
```

```
predicting---------------------------------
head is  RT @ziwe: michael cohen would have gotten more time if he had an eighth of weed on him and a better tan
body is  Michael D. Cohen, a former lawyer for President Trump, was sentenced to three years in prison on Wednesday after denouncing
Mr. Trump and explaining that "I felt it was my duty to cover up his dirty deeds."  Mr. Cohen gave an emotional apology to the court
for his involvement in a hush-money scandal that could threaten the Trump presidency — a scheme to buy the silence of two women who s
aid they had affairs with Mr. Trump to protect his chances before the 2016 election. Mr. Cohen said his blind loyalty to Mr. Trump le
d him to ignore "my own inner voice and my moral compass."  The sentencing in federal court in Manhattan capped a startling fall for
Mr. Cohen, 52, who had once hoped to work by Mr. Trump's side in the White House but ended up a central figure in the inquiry into pa
yments to an adult-film star and a former Playboy model before the 2016 election.
disagree
```

```
predicting---------------------------------
head is  RT @PalmerReport: Donald Trump's day so far:

- Refuses to get out of bed
- David Pecker screws him
- Michael Cohen slams him
- Sean Hannit…
body is  Michael D. Cohen, a former lawyer for President Trump, was sentenced to three years in prison on Wednesday after denouncing
Mr. Trump and explaining that "I felt it was my duty to cover up his dirty deeds."  Mr. Cohen gave an emotional apology to the court
for his involvement in a hush-money scandal that could threaten the Trump presidency — a scheme to buy the silence of two women who s
aid they had affairs with Mr. Trump to protect his chances before the 2016 election. Mr. Cohen said his blind loyalty to Mr. Trump le
d him to ignore "my own inner voice and my moral compass."  The sentencing in federal court in Manhattan capped a startling fall for
Mr. Cohen, 52, who had once hoped to work by Mr. Trump's side in the White House but ended up a central figure in the inquiry into pa
yments to an adult-film star and a former Playboy model before the 2016 election.
unrelated
```

Figure 7.