The Onion Connoisseur

Efe Akinci & Michael Zhou

What is The Onion?

not this

The Onion is a satirical newspaper and website.

They write funny articles.

What is the Goal of Our Project?

- Give articles from the Onion (and elsewhere) an Onion Score – a score on how obviously satirical the article is.
- Classify articles into general groups of satire and not satire.

obviously very funny

not so obviously funny

But How?

- We have provided a machine learning model we have created with a dataset of very real (Reuters) and very satirical (Clickhole) articles. This allows the computer to learn how to differentiate satire from real news articles to some degree*.

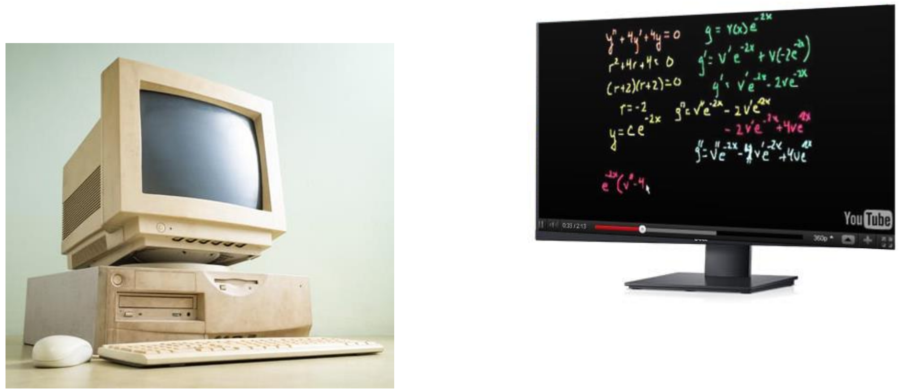


Fig. 1
A computer actively engaged in learning

“I saw an asterisk, why was there an asterisk?”

This section is also known as “problems.”

- Because of the dataset we used, we encountered some problems.
- Because Reuters writes similarly to a news wire, more narrative articles are likely to be determined as satire.
- The machine has no cultural context. A good illustration of this is it calling an article about Jesus Christ rejoining his NBA team real news.
- Additionally, although it is very accurate at identifying Clickhole vs AP News / Reuters (Wire networks) articles, it does not do as good of a job at differentiating The Onion vs CNN, for example.

~~"Why Didn't You Add Other Sources?"~~

Future Directions

This project can be improved by giving the computer a more varied diet of news articles.

Nerdy Stuff – Preprocessing the Data

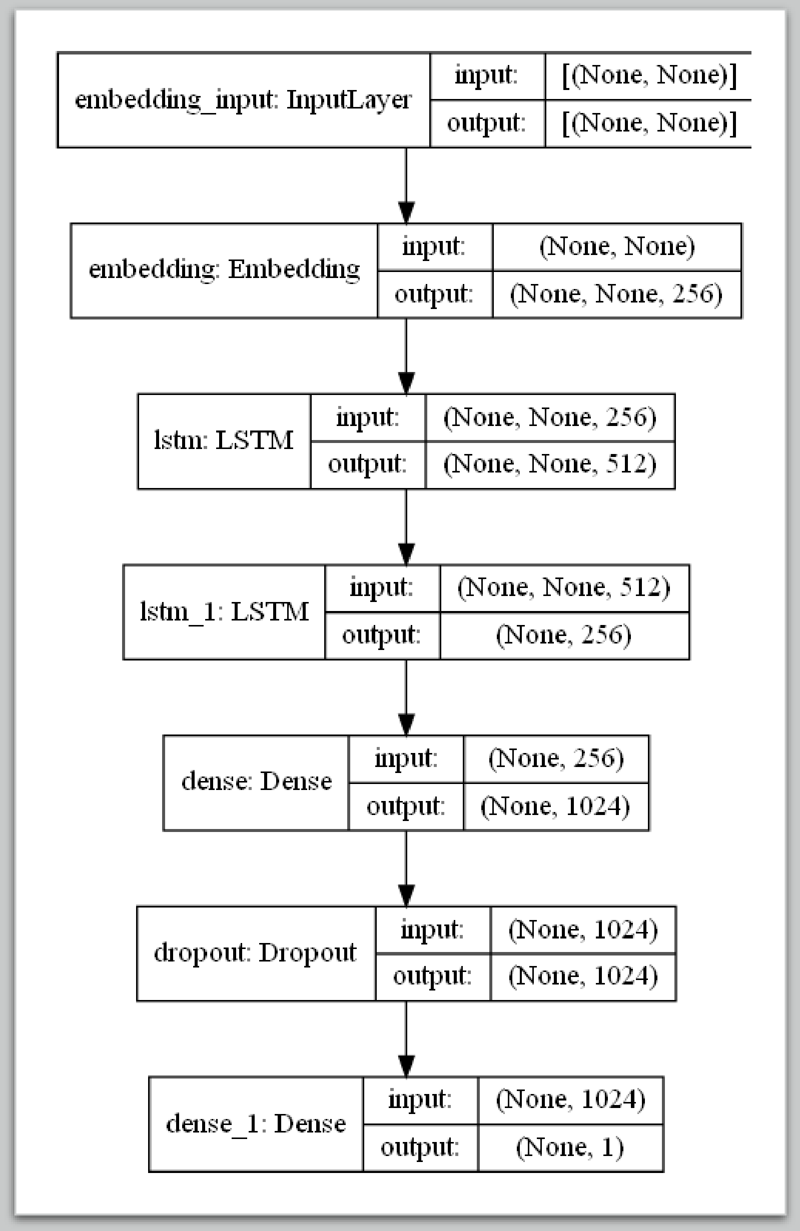
One of the first problems we encountered was preprocessing the data. Out of the box, it turns out that it was **very** easy to tell articles apart, just not in the way we wanted. Every single satire article ended in either “Featured Image by…” or “Photo by…” Apart from that, almost all of the real news started in the same way. There was a lot of other quirks as well. Here is a list of what we ended up doing:

- Stemmed words (this is important, as it groups words such as “onion,” “onions”, and “oniony” all into one group.
- Removed quotations
- Removed the word “Reuters”.
- Removed anything inside parentheses
- Removed links
- Removed contractions
- Removed article start patterns

Nerdy Stuff – The Machine Learning Model

- For the model itself, we used a neural network with an embedding layer followed by 2 LSTM (Long Short-Term Memory) layers, finally linking to a densely connected layer before the output layer.

- You can find an illustration of our model to the right.



embedding_input: InputLayer | input: [(None, None)] | output: [(None, None)]
embedding: Embedding | input: (None, None) | output: (None, None, 256)
lstm: LSTM | input: (None, None, 256) | output: (None, None, 512)
lstm_1: LSTM | input: (None, None, 512) | output: (None, 256)
dense: Dense | input: (None, 256) | output: (None, 1024)
dropout: Dropout | input: (None, 1024) | output: (None, 1024)
dense_1: Dense | input: (None, 1024) | output: (None, 1)

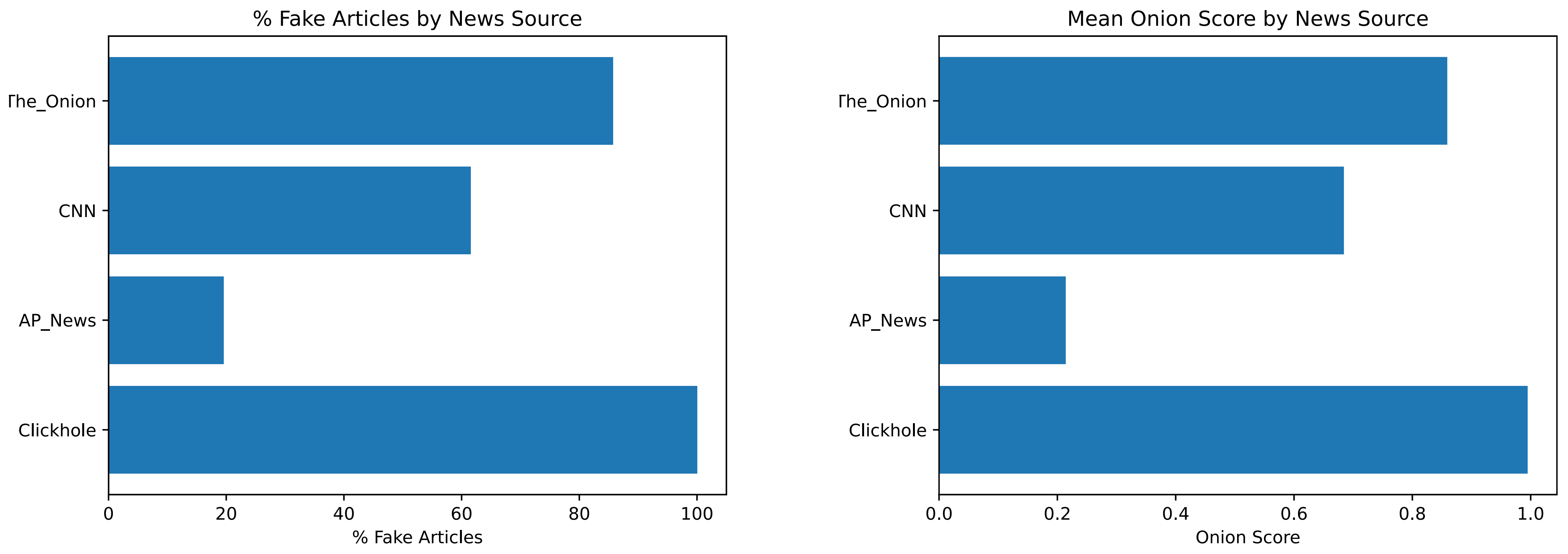
Onion Farming Instructions

1. Install all required packages. This can be found in requirements.txt
2. Download the training dataset from Kaggle. The link to this can be found under the preamble section in README.md.
3. Run utils.py. This will process the data, and output two new files for training.
4. Run ml_train.py. This will train the dataset on the processed data. This will take a **very** long time and be very processing intensive in the case that you do not use a GPU. The weights will be outputted to the "models" folder. Name your favorite h5 file "onion_connoisseur.h5" and put it into the "static" folder. For your convenience, you can download our saved weights to skip the last two steps. A link to this can be found in the README.
5. Run onion_farmer.py. This will download the latest news articles from various news websites. We advise caution when deciding how many news articles to pull. Your IP could very well be blocked if you put too much strain on any server.
6. Run visualizations.py to see the various visualizations our program can provide.
7. OPTIONAL – You can use predict.py to predict whether articles are fake or real by passing in a Pandas dataframe object with the texts you would like to predict under a column called "body."

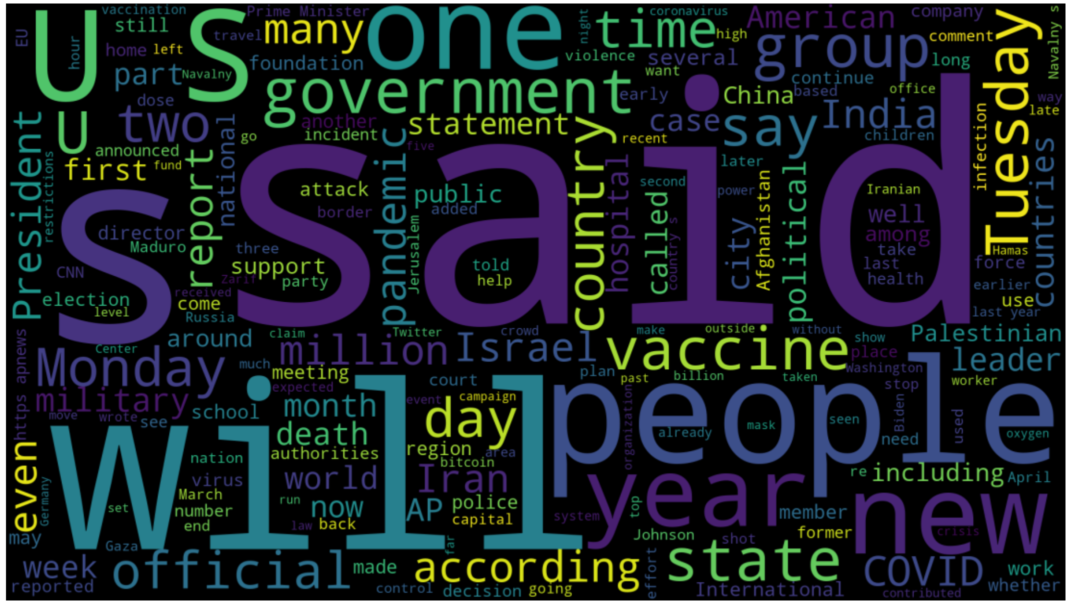
The Onion Cookbook

Program File	Description
ml_train.py	Creates tokenizer, tokenizes tests, and trains machine learning model based on data provided by DataProvider. Stores .h5 files from trained model and exports tokenizer for later use by predict.py.
process_data.py	Source of DataProvider class. Reads data, creates labels, and provides the training arrays for use by ml_train
onion_farmer.py	Downloads articles from the internet based on user input and stores them in the database onion_barn.db. It then uses predict.py to predict whether each article is fake or real and assigns each of them an onion score.
predict.py	Takes in a dataframe with a 'body' column and uses machine learning model to output predictions.
utils.py	Standalone class for preprocessing data for later use by process_data by providing data to the model in ml_train.py
make_wordcloud.py	Uses preprocessed data to generate wordclouds for the different datasets.
visualizations.py	Creates various visualizations based on article data and the predictions generated.

Oniony Figures

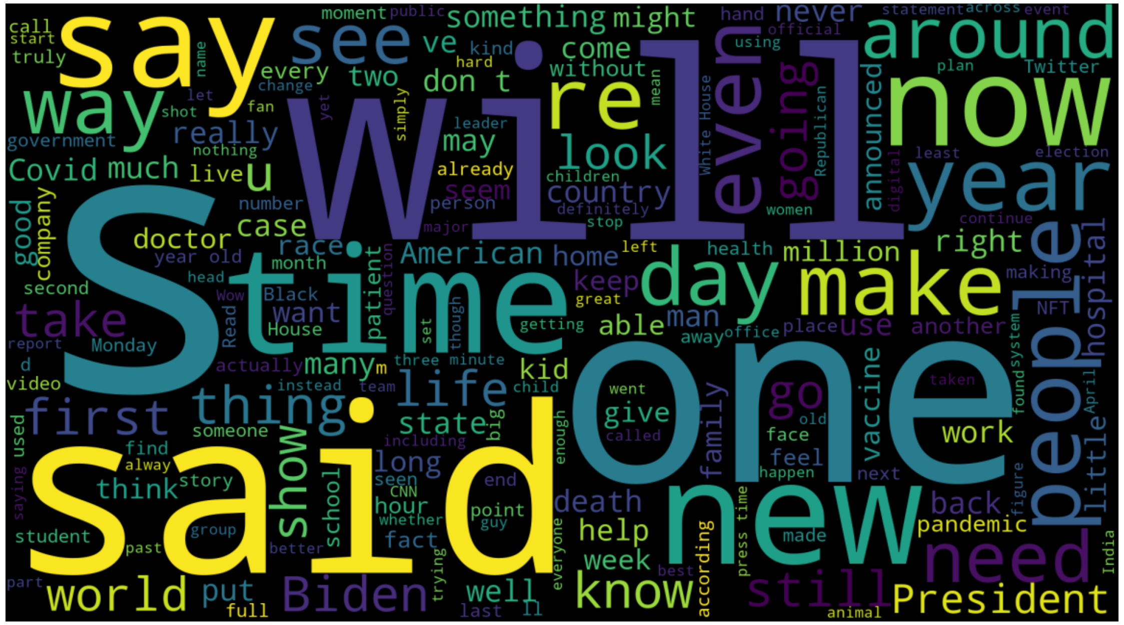


Word Clouds

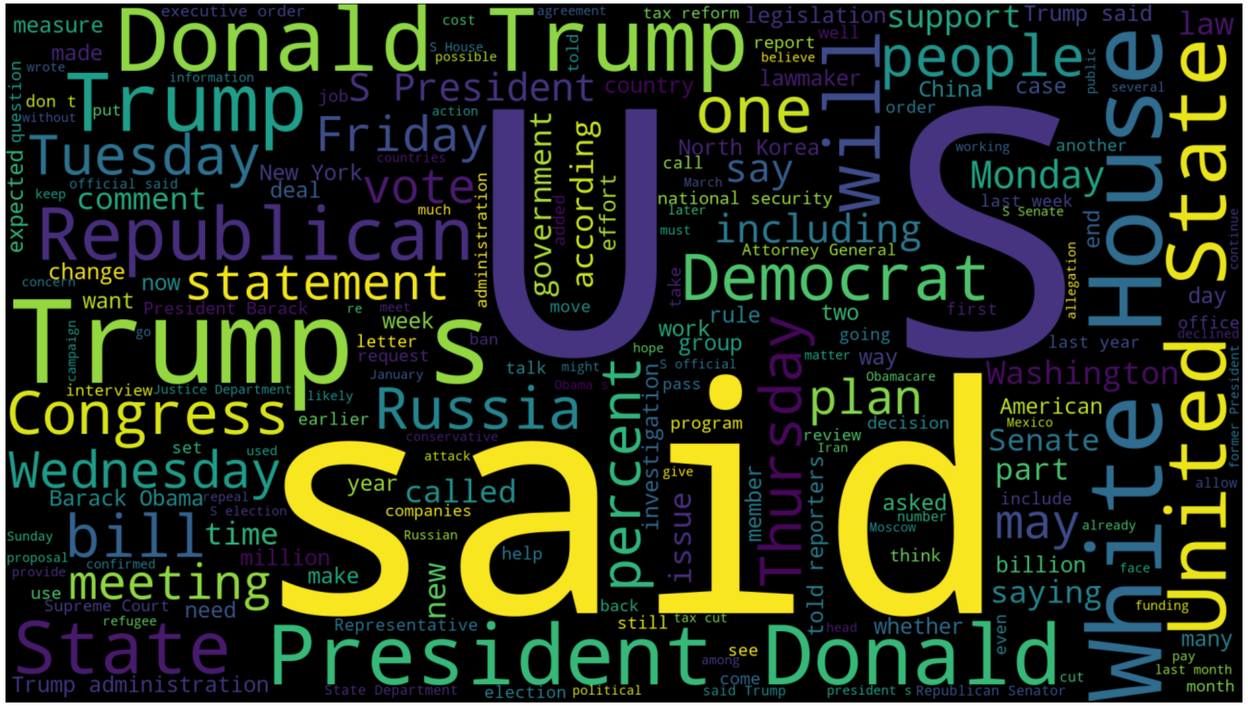


Predicted Real Articles

Predicted Fake Articles

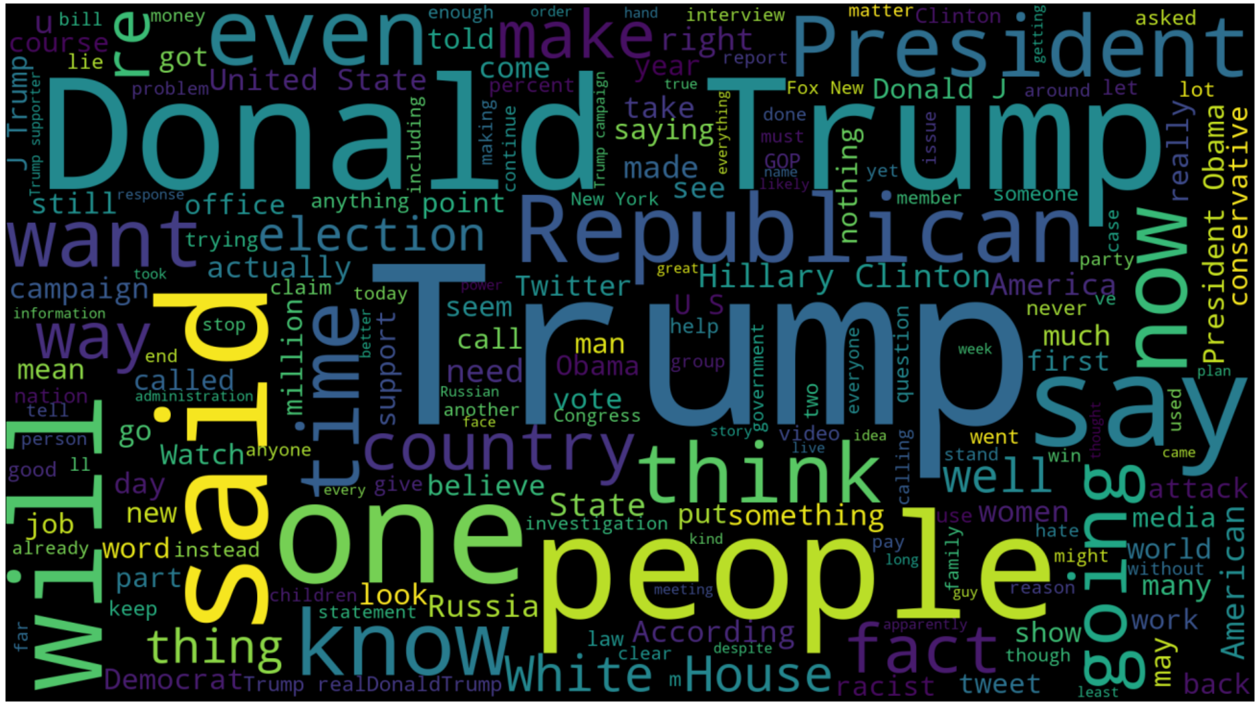


Word Clouds



Real Training Articles

Fake Training Articles



Resources Used

We used this GitHub repository's README to help set up our virtual environment, and to write the venv instructions for our own project's README. (https://github.com/tl-its-umich-edu/course-inventory)

Additionally, https://machinelearningmastery.com/ was a big help in everything machine learning related.

We used this Stack Overflow post (https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas) to learn how to iterate over the rows of a Pandas dataframe object.

Work Division

Efe – ML Model Creation and Training, Data Preprocessing, News Article Scraping, Database Predictions, Word Clouds, Presentation Report.

Michael – API Management + Reddit Scraping, Database Management, Mathematical Calculations, Visualizations, Data Preprocessing

Our GitHub repo can be found at: https://github.com/efea-umich/SI-206-Final