Fake News Detection Final Report

Team Member: Brian Edmonds, Xiaojing Ji, Shiyi Li, Xingyu Liu

Motivation and Objective

Automatically detecting and analyzing fake news on the internet is a hard problem that has yet to be solved. We strive to build a system that demonstrates how machine learning can be used to detect credible vs non-credible news sources. We also experiment with different machine learning models and select features from both the article headline and article body.

Related Work

Several projects have taken aim at Fake News since the 2016 Presidential election. One popular approach, which we adopted in order to benchmark our machine learning data, is to compare a given URL with the OpenSources.co dataset[1]. The OpenSources.co dataset is a list of Credible and Non-Credible news sites maintained by researchers at Merrimack College. A popular Chrome Plugin, B.S. Detector[2], solely uses the OpenSources.co dataset to make a judgement about whether or not a given URL is fake news. Our technique surpasses this approach because it leverages machine learning to statistically analyze given news articles and does not rely on a "blacklist" of news articles from OpenSources.co. As with all blacklists, a fake news site that has not been seen previously by the team at Merrimack college will not be correctly identified as Fake News by the B.S. Detector.

System Overview

This system is deployed as a web application. We chose Python Flask[3] as the primary frontend framework because of its flexibility and extensibility. Flask is a "microframework" for developing small web applications with multiple views or models. Considering the web only contains the main page and a result analysis page, Flask becomes the ideal framework for such simple web interface. Additionally, Flask abstracts away the complexity of the backend, which provides at least six machine learning models.

The entire system applies multiple packages and machine learning libraries for the training and predicting. Major components involved in the machine learning process includes Numpy, Scipy, scikit-learn, Keras, and Theano. The following provides more details about each single component:

[1] http://www.opensources.co/
[2] http://bsdetector.tech/
[3] http://flask.pocoo.org/

- Numpy: a scientific computing package generating N-dimensional array objects. As for this project, several machine learning models use Numpy as the data container; the implementation of our random tree and random forest also depends on this.
- Scipy: an open-source library which contains scientific tools, our random forest use this to generate classification result.
- Scikit-learn: a Python library built on Numpy. This project uses it mainly for data classification.
- Keras: a high-level Neural Network API. In this project, the content keywords analysis applies the sequential model on recurrent layers.
- Theano: the main reason of introducing Theano is that Keras API requires to run on top of certain low-level libraries. Theano is designed for mathematical evaluations on multi-dimensional data.

Other than those machine learning tools, libraries for extracting different attributes from a given news website link are essential for the system. Standard Python library like urllib is applied for the title extraction. A third party open source library named Newspaper aims at collecting content keywords from the given website url.

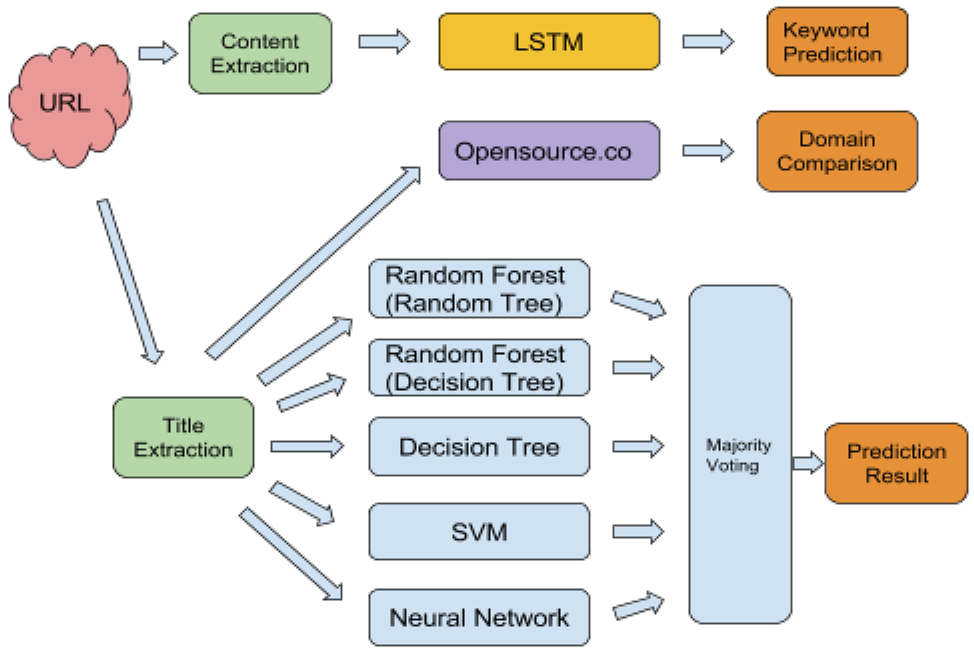
Given the complexity of recognizing fake news, we decide to apply six machine learning algorithms and hope this aggregating approach will provide a more reliable detecting result. Below are the algorithms that we applied and a brief explanation of each.

- SVM: Support Vector Machine (SVM) is a powerful linear classification algorithm. By solving the maximum margin problem, it aims at finding a hyperplane decision boundary in the middle of the two classes. In our model, the soft-margin version is used.
- Decision Tree: Decision tree is a tree model, the query process corresponds to a path from the root to a leaf. In each inner node one feature value will be examined, by comparing with a pre-calculated value it will decide which subtree to go. When it reach a leaf, the result stored in will be the answer. In our project, Gini impurity is used to determine which feature to examine for every node when building the tree.
- Random Forest (Decision Tree Version): The random forest is a collection of different trees which are trained by different sub-samples of the dataset, therefore random forest is less susceptible to overfitting when compared with many other models. This version is constructed based on decision trees which use Gini impurity when training.
- Random Forest (Random Tree Version): This is also a random forest. The most important difference from decision tree version lies in the trees inside. They are not selecting features based on measurements like Entropy and Gini impurity, but by randomly choosing instead. The method is proposed in a paper by Adele Cutler and Guohua Zhao [1].
- Neural Network: The neural network working behind is a feed-forward neural network. It contains 105 neurons(perceptrons), divided into 3 layers. Neurons in the 1st layer will read in the feature, and neurons in the remaining 2 layers will read output of neurons from last layer.

- LSTM: LSTM refers to Long-Short Term Memory, which is a recurrent neural network architecture. It achieved the best known results in natural language text. This model applies binary cross—entropy to the preprocessed content keywords data.

System Architecture

- ❖ Data: Collected ~160 news samples from fake news and ~40 news samples from real news websites, which were part of our training dataset. In addition to manually collection, we also use the fake news dataset from kaggle.
- ❖ Framework: Build system as a web application with flask
- ❖ Our entire system architecture is shown as below:



Headline and Keyword Extraction

Some of our Machine Learning models use the headline of a given news articles as input. We extract the headline from a given URL by first downloading the HTML being served at that URL. Next, we use regular expression to identify the “Title” tags of downloaded HTML and we extract the text between the tags. Almost always, the title tags of a news article will contain the headline of the article followed by a special character and the name of the news organization. For example:
<title>Attack on Champs-Élysées Injects More Uncertainty Into French Vote - The New York Times</title>

Our application extracts only the headline from between the title tags and ignores the name of the news organization. In order to separate the headline from the name of the organization, we split the string value on the rightmost special character. Next, ignore all characters to the right of that special and assume that the remaining text is the article headline.

Keywords are extracted from the article content based on word frequency. We used the newspaper python library[4] in order to extract these keywords which also ignore “stopwords,” the most common words in the English language.

Feature we use in Headline analysis:

1. Number of words
2. Length of headline
3. Number of punctuation marks
4. Number of continuous punctuation sections
5. Number of continuous upper case letter sections

Five classification model:

1. SVM
2. Decision Tree
3. Random Forest (Decision Tree Version)
4. Random Forest (Random Tree Version)
5. Neural Network

Since all the classification models works with different methods but each achieve similar accuracy during our test (10-fold cross-validation with data shuffling), we adopt a majority-voting scheme. Given one news to be classified, all the models will predict and “vote” independently. Next, we tally up the results (fake/real) from each of the models. The final ruling of the system will be based on the majority “vote” from all of the models.

LSTM Content Keywords Model

The unique characteristic of a recurrent neural network is that it doesn’t restart scratches from the beginning. Instead, loops in its chain-like architecture ensure the persistency of the past information, which distinguishes it from traditional neural networks. More precisely, LSTM is capable of learning the long-term dependencies of a given data set. Thus, it often can be successfully applied to many language-related tasks. In the context of our Fake News detection system, we apply LTSM to keywords of an article. To properly use LSTM on content keywords, the model first needs to preprocess the data extracted by Newspaper API. For instance, the preprocessing procedure maps each unique word onto an integer in the defined scope. In this model, each piece of keyword set represents content keywords from a single news web. Sixteen neurons are placed in the sublayers for training.

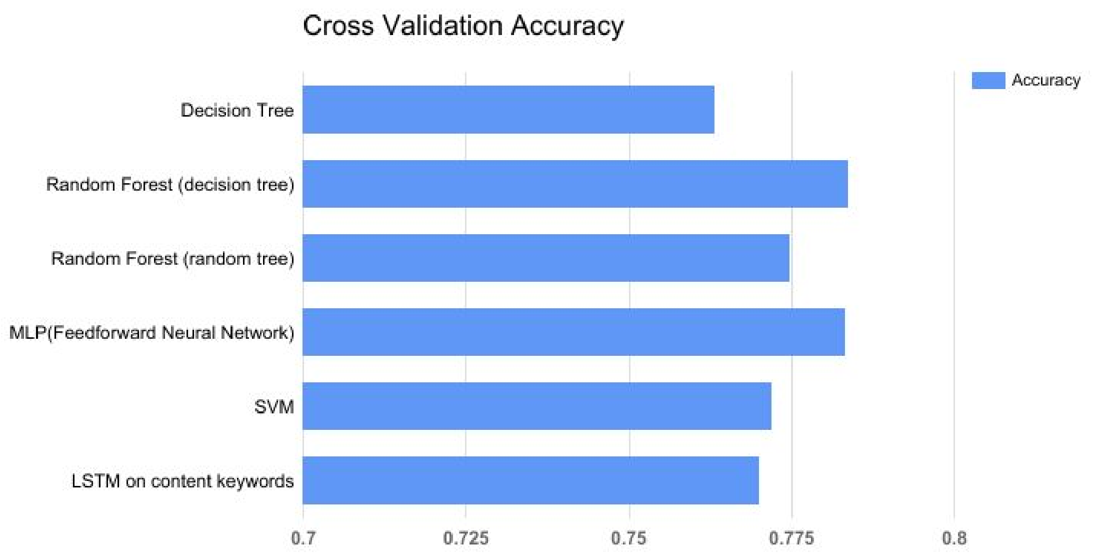
Results

Accuracy

[4] http://newspaper.readthedocs.io/en/latest/

We measure the accuracy of the different classification models by applying 10-fold cross-validation. The dataset is shuffled and divided equally into 10 parts, and, in each sub-test, one section is selected as test set, and remaining nine sections will be training set.

Cross-validation: the accuracy results of cross-validation are displayed in the graph below. Five classification models share the same title dataset for the cross-validation. LSTM uses the keywords dataset generated from different attribute of the news sites.



Execution Time Performance

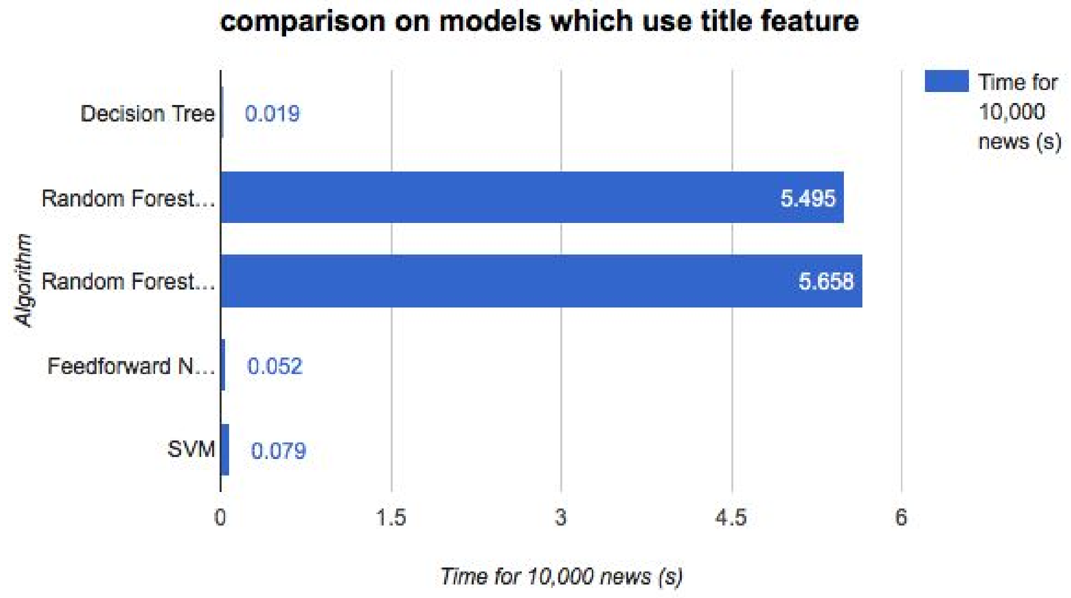
In order to measure the execution time of our models, we conducted multiple tests. In each test we randomly pick 10,000 news articles from our data set(with replacement) to imitate real-world, high volume user access. Next, we record the total time required for to construct features for the machine learning algorithms on backend. We will use the average time among several analog tests to measure the performance.

Time for feature construction: Step 1 one of our test is to construct the feature vector for a given news article. In our project, different features will be used for different models. For SVM, decision tree, neural network and random forest (2 versions), the features of the article headline are extracted and used; whereas, for LSTM, the keyword of the article content is extracted and used. The chart below shows the time required to construct the features for 10,000 news articles. It is important to note that the time to analyze the content keywords of each news relies on the website: in our test the time to extract features from NPR is faster than the amount of time to extract features from CNN.

Method	**Time for 10,000 news (s)**
Title feature extraction (5-dimension)	0.651
Content keyword analysis	267.925

Time for machine learning algorithms: This section evaluates the time needed to run the machine learning algorithms after the feature matrix has been constructed (i.e. the time to construct feature vectors are excluded). The chart below shows the average time for the models to classify 10,000 news. It is important to note that LSTM uses a high dimensional feature vector (keyword feature) to classify, and the remaining 5 models will use the same title feature to classify.

Algorithm	Time for 10,000 news (s)
Decision tree	0.019
Random forest (decision tree version)	5.495
Random forest (random tree version)	5.658
Feedforward neural network	0.052
SVM	0.079
LSTM	456.898



Model Comparison (time and accuracy):

Based on our test results, all of the models work with good efficiency and accuracy. The running time to analyze 10,000 news shows that as long as the website doesn't encounter extremely high volume user access, it can provide feedback in a reasonable amount of time. When generating the feature s for a given news article, it takes more time to extract out the keywords of the news content than it does to extract the features of the article headline. Regarding the machine learning algorithms, LSTM will work on the high-dimensional news content feature, and it will take a longer time than others. For the 5 models which use 5-dimensional title feature for classification, random forests work slowest, the reason is that they must do query on all the decision/random trees inside.

Regarding the classification accuracy, as is shown before, we use the average accuracy in 10-fold cross-validation for measurement. The chart shows that all the models provide similar levels of accuracy and that there isn't a model which can outperform the others significantly. All five classification models give similar accuracy, ranging between 0.76 to 0.8. This is one of the reasons why our system implements the majority vote for the final decision. The LSTM model doesn't provide a significantly better prediction. In a sense, the highest and the lowest won't have a big difference (about 3% only).

What system doesn't do well:

Our system doesn't perform well when fake news' headline tend to be plain, short and emotionless. For example, *"7 Successful Small Business Ideas In Kenya For 2017"*. This is a fake news coming from *"abcnews.com.co"*, which is a known fake news website. If we analyze using just the domain name, the opensources.co dataset correctly identifies the news article as fake news. However, if we apply our machine learning model, all of our machine learning models incorrectly classify the news article as real news. This example of our system not correctly identifying a given article as fake news is almost certainly tied to the data that we use to train our machine learning models.

Deliverables

The main component of the final deliverable is a Flask web application with six machine learning models in the backend. The current application can run on localhost and will be deployed to either Amazon EC2 or Digital Ocean. The source code is available on Github: https://github.com/XJi/Fake-or-Fact.

The deliverable also includes this project report as the complementary because we provide some discussions and a detailed description of the system architecture on this report. This project involves many approaches which haven't been implemented before for solving this specific issue. Considering that detecting fake news online has been recognized as a difficult task and people are still exploring for better solutions, we decide to make our source code public for future reference and analysis. We have stated constraints of the current system and some other potential approaches. We hope people interested in this field can get inspired and continue making contributions on top of our work.

Future Work

During the testing phase, we discovered a few ill-designed websites don’t support our currently implemented title extraction method. In order to avoid potential crashes, we’d like our system to be able to handle exceptions like this.

Additionally, when preprocessing keywords data, the system uses one_hot, a built-in mapping function, to map every unique word to an integer value. This operation removes many useful features from the dataset. For example, it does not preserve the meaning of each word. Future work could attempt to preserve the meaning of each keyword while still maintaining the ability to run the LTSM model.

Our research into the topic of fake news and methods to detect it uncovered some interesting techniques that could be nice additions to our system. Some experts in journalism have suggested that conducting a reverse image search of the main image used for a news article is a good way to identify fake news[2]. The intuition is that an article that recycles an image from elsewhere on the internet is likely not to be credible. Most credible news organizations employ their own photographers and have original photo content. We think that implementing a reverse image search into our detection algorithm could yield good results. Additionally, NPR advised that vigilant readers should check the number of quotations in an article to identify a non-credible news source[3]. The thinking is that credible news sources will employ numerous direct quotations while non-credible news sources will not. The number of quotations in an article could be another good feature for us to add to our machine learning models.

Finally, during our presentation in class, we received some valuable feedback from our peers. They suggested that our fake news detection system could be “outsmarted” by a fact-free article that was formatted and written in a seemingly “professional” way. They pointed out that a site such as The Onion, which is written with a serious tone and presented in traditional news article format, but is clearly satire, would potentially be judged by our system as credible. Future work on our project could strive to correctly identify The Onion and other sites as non-credible even if they have the traditional news article format.

Reference

1. Cutler A, Zhao G. Pert-perfect random tree ensembles[J]. Computing Science and Statistics, 2001, 33: 490-497.
2. Katie Rogers and Jonah Engel Bromwich. "The Hoaxes, Fake News and Misinformation We Saw on Election Day." *The New York Times*. The New York Times, 08 Nov. 2016. Web. 22 Apr. 2017. <https://www.nytimes.com/2016/11/09/us/politics/debunk-fake-news-election-day.html>.
3. Davis, Wynne. "Fake Or Real? How To Self-Check The News And Get The Facts." *NPR*. NPR, 05 Dec. 2016. Web. 22 Apr. 2017. <http://www.npr.org/sections/alltechconsidered/2016/12/05/503581220/fake-or-real-how-to-self-check-the-news-and-get-the-facts>.