6242 Group Project Final Report

Classificiation and Visualization of Fake News

Mansi Arora
MS Analytics, Georgia Tech
marora33@gatech.edu

Rishi Bhatia
MS Analytics, Gerogia Tech
rbhatia37@gatech.edu

Michael Cho
MS Analytics, Georgia Tech
mcho89@gatech.edu

Jeh Lokhande
MS Analytics, Georgia Tech
jlokhande3@gatech.edu

Taylor Million
MS Computer Science, Georgia Tech
tmillion3@gatech.edu

Nabila Usmani
MS Analytics, Georgia Tech
nusmani6@gatech.edu

1 INTRODUCTION / PROBLEM DEFINITION

Fake News is an increasingly pervasive trend that facilitates the misinformation to internet consumers. This project aims to develop a model to detect these fake articles, and visualize their topic spread. Our project group innovated in the following manner:

(1) Develop a fake news model to classify articles, using an exhaustive set of feature extraction methods that combines techniques from the current state of art techniques.
(2) Provide a visual insight into the classification of news articles and the decision process behind it. This visualization includes:
   (a) An intuition behind the classification of the article by highlighting components that contributed to the classification. For example, if an article uses unusual punctuation such as '???' then it will be highlighted in the article.
   (b) A node map of the article keywords. Upon hovering on a node, a google trend chart of the keyword will be displayed, providing the user with another way of visualizing the rate at which the keyword topics are spreading.

2 BACKGROUND / MOTIVATION

Fake news, a decade-spanning problem, was historically detected with human fact-checking resources (such as Politifact, Snopes and Full Fact). Human fact-checking is expensive, cumbersome and erroneous. The incentivization of ads and nature of virality has created a business model based on click-bait and fake news dissemination making human-based fact checking obsolete. Competitions, initiatives and startups such as Fake News Challenge [10], Digital News Initiative [8], Make News Credible Again (MNCA) [16] and Factmata [9] have recently set out to tackle the problem using natural language processing (NLP) and artificial intelligence (AI). The current limitations of the research on the subject are the following:

- Limited scope of the fake news classification models
- Limited resources available on the science of fake news
- Lack of insight provided on the reason for the classification

3 SURVEY

This project spans many topics, from linguistic features to AI classification models. Though fake news detection research is expanding, few works use a combinatorial model and even less use article-related topic visualization as this project does. Relevant work in related areas like NLP and modeling is described here.

3.1 Foundational Works

To tackle the fake news classification problem, two datasets (Emergent by [11] and Liar by [22]) were developed, evaluated and are now the foundation of research. This project leverages Kaggle's fake news dataset and Signal One Media's news dataset (1 million rows), restricted to only the top few trusted news providers.

3.2 Natural Language Processing

Many works around fake news detection focus on the comparison of NLP components of the classification. Linguistically motivated features such as Bag-of-Words, Part-of-Speech(POS) tagging, Linguistic Inquiry Word Count(LIWC), Term Frequency-Inverse Document Frequency (tf-idf), Brown Clustering and N-grams are discussed by [1, 4, 6, 12, 14, 17, 18], respectively. This work will assist with feature selection along with other meta-feature selection. Modeling improves upon Perez-Rosas', Khurana's and Shu's exclusively linguistic research and use of global semantic meaning features will improve Davis', Gunasekaran's and Chui's Support-Vector-Machines (SVM).

3.3 Algorithms for Model development

Much fake news research is around modeling techniques for classification. These models include boolean label crowdsourcing [21], KNN [19], Naive Bayes [5], SVMs [4], RNNs [13], CNNs [2], Topic RNNs [7], LDA Topic Modeling [3], multilingual context [20] and time-sensitive probabilistic models [15]. The discussion of these models, their implementation and performance evaluations provide a great reference for choosing model algorithms. This project's cohesive model will be an improvement by using global semantic meaning features.

4 PROPOSED METHOD

The project's approach is novel because it classifies news articles, visualizes the google trends of the article topic and justifies article classification.

4.1 Intuition

The model was trained using Kaggle's fake news and Signal Media One's news dataset, along with other news articles scraped using a python script. Unlike most fake-news detection frameworks, our approach focuses on visualising multiple aspects of fake news such as the descriptive features of the articles (Number of capitalized words, First person word count), related topics of the article and

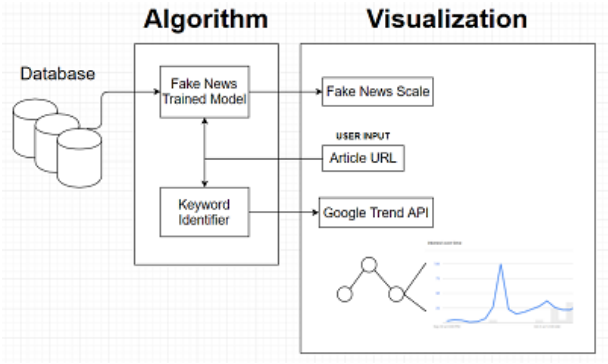

Figure 1: Visualization of Project Pipeline

their google trends, along with the contents of the article with interesting features highlighted.

4.2 Description of Approaches

Data collection

The data was collected from various sources to create a comprehensive collection of real and fake news articles. Fake news articles were taken from Kaggle's dataset consisting of 12,999 posts with their text and metadata scraped from 244 websitestagged as "BS" by the BS Detector, which is a Google Chrome plugin. Signal Media One's dataset was used to obtain real news articles by only retaining articles from reliable sources such as Reuters, CNN, CNBC. Apart from these two data sources, a python script was used to scrape newspaper articles that can be used to update the dataset whenever required.

Data cleaning

The data scraped and collected has non ASCII characters (graphical characters such as ™, ©) which were removed. All articles with a character count of <1000 were removed from the dataset. Regex was then used to remove:

(1) New line characters such as CR LF (\t \n \r)
(2) URL's in the text body that don't provide any information

After extraction of features from the text body, all non-space, non-alphanumeric characters were removed and text was converted to lowercase for input to the word2vec/GloVe

Feature Creation

Feature creation can be broadly split into the meta-features (hand-crafted) extracted from the body of the text and the feature vectors from the output of the word2vec/GloVe model. The below features were extracted from the text body:

(1) Word count - Count of words in the article
(2) Unique word count - Unique words in the article
(3) Special character count - Characters that are graphical characters or punctuations
(4) Average word length - Average length of words in the article
(5) Capitalized word count - Capitalized words in the article
(6) First Person word count - Words that are first person words in the article. Word vectors are the vectors that are created after the output from the word2vec model (trained on google's word corpus). We also compared performance with the global vectors model (GloVe) by Stanford, but decided to finalise the word2vec approach.

Model Building

An SVM model was used as a baseline. Then, a Sequential Deep Learning model was fitted and tuned for hyperparameters such as number of layers. We used cross validation to avoid overfitting. Later, when the same problem was modelled using an XGBoost model. Similar accuracy were obtained for out-of-sample tests. Since the XGBoost model is a simpler model than the Sequential Deep Learning model, we chose it as our final model.

Topic Modeling

Topics or abstract themes were extracted based on the underlying articles and words in a corpus of text. These topics were used to visualize trends, search interest areas, and top Google queries related to the topic.

Visualization

The visualization contains 2 main components:

(1) An initial page for user input
(2) A dashboard for result visualization
  (a) A "Fakeness Score".
  (b) Result: details of three features used to determine the fakeness of the article.
  (c) Article contents showing plain text format. Upon hovering over a feature in the result details, the sections associated with the feature will be highlighted in the article contents.
  (d) Related topics, which shows a node map of the keywords in the article. Upon hovering over a node, a visualization of the google trend information for that keyword will appear on a pane to the right.

Figure 2: Sample Visualization (Initial Page)

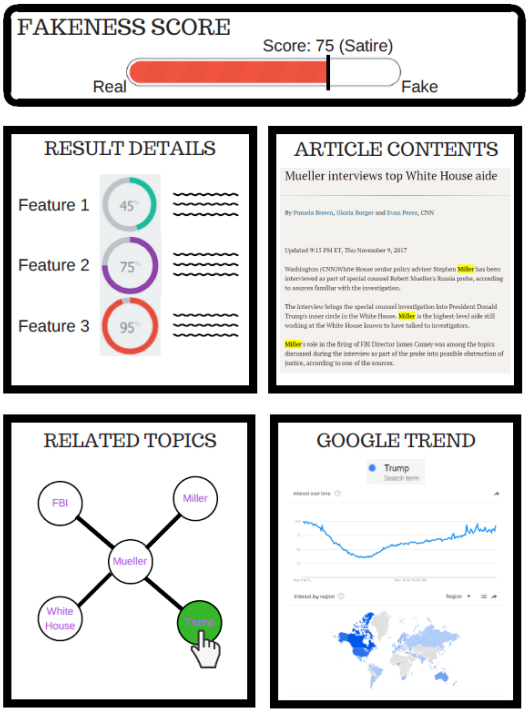

Figure 3: Sample Visualization (Results Page)

Integration of the Model with the Visualization

The model was integrated with the visualization through an interface that can take a URL input, scrape the article's content, apply the models, and visualize the results in a dashboard format. On the backend, Flask is used as a webserver to route webpage traffic. After the URL is entered, the HTML calls the python function to scrape the content of the provided URL. The newspaper python library is for article scraping. Upon completion of the scraping, our final classification model was applied to the article text. Next, the results are displayed using a keen.io dashboard template. The Keen.io API uses MongoDB as an interlink to funnel data to visualizations. This allows for flexibility if a need for data storage arises. Currently, the visualization is textual output of the results and in the future, plotly will be used to create the graphical representation of the results. Finally, Dokku can be used to easily deploy the Flask app to scale the project from a local implementation to production.

5 EXPERIMENTS AND EVALUATION

(1) Can articles be scraped from news websites along with their metadata?
- The newspaper python package can be used to extract article text, author information, and movie/image urls if any

(2) Can article topics be extracted from the scraped text?
- Yes, this was tested using the properties of the newspaper object.

(3) What cleaning does the data require?
- All non-ASCII characters, new line characters, stopwords, URLs, etc. were removed

(4) What are the various features that can be used for developing the model?
- Features such as number of capitalised words, word2vec, and others mentioned above are extracted

(5) What is the baseline for model performance?
- An SVM model was used as a baseline, as many previous research studies had proven results using this technique. The model reported an accuracy of 64.3 percent on the out-of-sample dataset.

(6) What Other Modeling Methods were Tried?
- A Sequential deep learning model was tried, with various modifications such as number of layers, cross-validation sets and number of epochs. With 2 hidden layers, a 30 percent cross validation set, and 50 epochs, the model reported an accuracy of 93.2 percent on the out-of-sample dataset. The figure below shows how the model accuracy varies for training and validation sets over epochs.

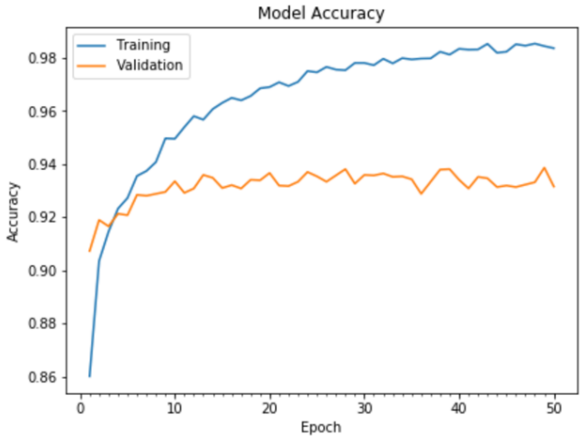

Figure 4: Model Accuracy of Sequential Deep Learning Model as Function of Epochs

- However, when the model was exposed to totally different news articles, we witnessed a significant drop in the accuracy.
- An XGBoost model was also tried, and performed significantly better than the previously tried methods. This was our final model, with an out-of-sample accuracy of 94.08 percent

(7) What are the various visual elements that can be added to the final dashboard?
- The final dashboard will consist of the newspaper article with highlighted features, along with its fakeness score. The topics associated with the article will be displayed along with the Google API trend charts.

(8) Can code be triggered on html events?
- Yes, Flask can be used to route webpage traffic and call functions to scrape data and apply models.

(9) Can the Google trend API be integrated with the visualisation framework in place?
- Yes. Google trend API was integrated with the visualization framework using the pytrends library in Python. To make best use of the Google trend API, the top five most used proper/regular nouns were extracted from the input fake news article text. From there, each two word combination of the top ten nouns was entered into the Google Trends API via pytrends to determine which searches had the most article related searches in the "related queries" section. Two word combinations were used, as they were the most effective in generating relevant results Google Trends. Using only one of the frequent nouns generated results that were too broad and unrelated to the article, while using to many words were too narrow and yielded no results. Then, using plotly, the results were outputted as a node map, where the root nodes were the two word combinations and the branch nodes were the related queries from the Google Trends API.

Figure 5: Home Page

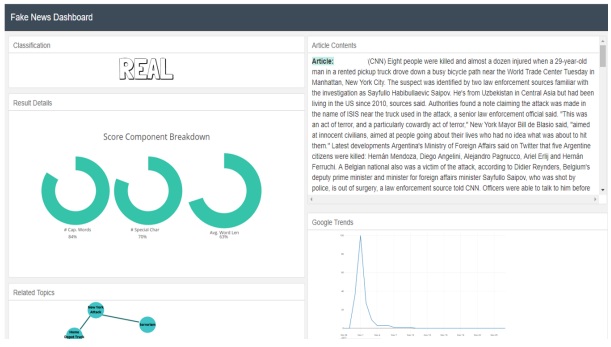

Figure 6: Results Page

(10) Is it possible to record user agreement/disagreement with the classification?
- A user study was conducted as part of the experiments where 10 articles that were a mix of real and fake news were identified and users were asked to rate their opinion of whether they were real/fake. The articles were chosen such that most of them did not have a famous source/news agency and hence were difficult to classify. Out of the 6 news articles that were actually fake, the users were able to correctly identify only 4 articles. This reinforces the need and utility for an automated method like the one developed as part of this project.

6 DISCUSSION

From our experiment on the different datasets and different models, it was observed that the domain of news plays a large role in the training of the model. News from tech trains the model very differently as compared to news from sports or from finance. This is mainly because of the style of writing in these domains and the way fake articles are compiled. Our models hence are very specific to the category of general or regular news.

Furthermore, deep learning models require a large amount of data to train, and to be classified as fake or real with certainty. This takes a large amount of time since data sources (either classified as fake or real) on the internet still need to be double checked for whether they are really fake or real. This is one of the reasons why XGboost gives a more robust classification than Neural Nets. However, the deep neural net model could potentially give a better performance on expanding the training data set.

In terms of visualization, the features extracted and google trends data provided gave the user a reasonable intuition as to how our model was classifying news as fake or real. Future work could improve the visualization by providing more explanation as to why specific features are extracted and increasing the size of nodes where a specific fake news search term is spreading more rapidly.

Finally, there are a lot more semantic features that can be extracted from the articles using NLP and pattern packages such as sentence structure, timeline of the news (some article topics that were fake earlier might not be fake now) and url data.

7 DISTRIBUTION OF TEAM MEMBER EFFORT

All team members have contributed similar amount of effort.

8 CONCLUSION

Fake news classification has many more nuances that can be addressed. Our model so far has achieved an accuracy of 94.08 percent from the dataset accumulated. However, the model could be made more robust by increasing the diversity of news sources, having additional data to train on, and having more semantic features extracted. On the visualization side, more explanation on features extracted and more emphasis on specific nodes for the google trends data could improve user intuition.

9 APPENDIX

How to use the fake news visualization demo:

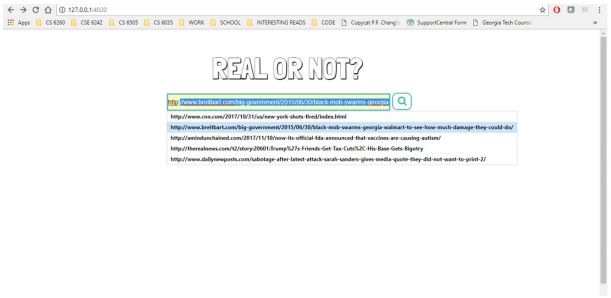

Figure 7: After running run.py, open a browser and enter "127.0.0.1:4020". Then, in the search box, enter an article URL.

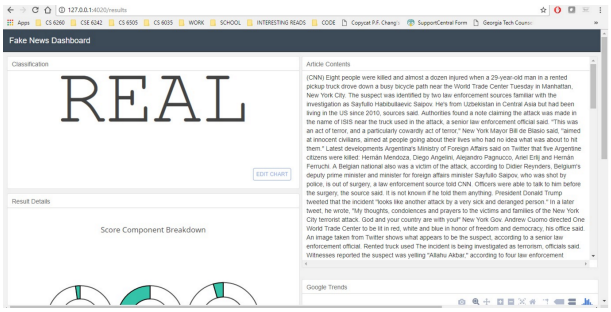

Figure 8: Top third of results page. Contains whether news article classified as real or fake (Left) and the Article in plain text (Right)

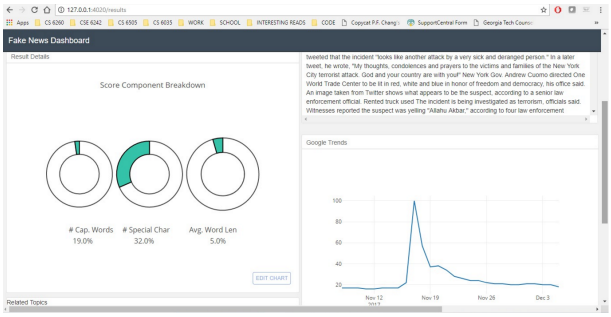

Figure 9: Middle third of results page. Contains a visual of the features extracted from the Article

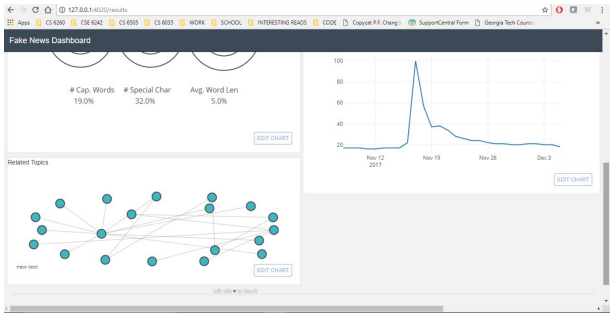

Figure 10: Bottom third of results page. Contains node map (Left)

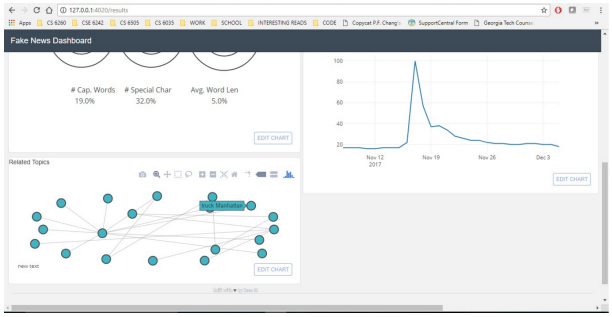

Figure 11: Bottom third of results page. When hovering over node map (Left), Google Trends data appears (Right)

REFERENCES

[1] Ahmet Aker, Leon Derczynski, and Kalina Bontcheva. 2017. Simple Open Stance Classification for Rumour Analysis. (2017). http://www.derczynski.com/sheffield/papers/simple-open-stance.pdf

[2] Samir Bajaj. 2017. "The Pope Has a New Baby!" Fake News Detection Using Deep Learning. (2017). https://web.stanford.edu/class/cs224n/reports/2710385.pdf

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. (2003). http://machinelearning.wustl.edu/mlpapers/paper_files/BleiNJ03.pdf

[4] Justin Chiu, Ajda Gokcen, Wenyi Wang, and Xiaohua Yan. 2013. Classification of Fake and Real Articles Based on Support Vector Machines. (2013). http://www.cs.cmu.edu/~xiaohuay/papers/report_11761.pdf

[5] Niall J. Conroy, Victoria L. Rubin, and Yimin Chen. 2015. Automatic Deception Detection: Methods for Finding Fake News. (2015). https://www.researchgate.net/publication/281818865_Automatic_Deception_Detection_Methods_for_Finding_Fake_News

[6] Eric Davis, Jason Adams, and Shay Cohen. 2007. Classifying Articles as Fake or Real. (2007). https://pdfs.semanticscholar.org/1899/d5ff9d3a0f3ae93195a8ead138d8a95e5d45.pdf

[7] Adji B. Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2017. TOPIC RNN:A Recurrent Neural Network With Long-range Semantic Dependency. (2017). http://www.columbia.edu/~jwp2128/Papers/DiengWangetal2017.pdf

[8] Digital News Initiative 2017. Digital News Initiative. (2017). Retrieved October 2017 from https://digitalnewsinitiative.com/

[9] Factmata 2017. Factmata. (2017). Retrieved October 2017 from http://factmata.com/

[10] Fake News Challenge 2017. Fake News Challenge. (2017). Retrieved October 2017 from http://www.fakenewschallenge.org/

[11] William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. (2016). http://aclweb.org/anthology/N/N16/N16-1138.pdf

[12] KUMARAN GUNASEKARAN, GOWTHAM GANESAN, and SUDARSHAN SRINIVASA. 2016. Fake News Detection in Social Media. (2016). https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a085.pdf

[13] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-Term Memory. (1997). http://www.bioinf.jku.at/publications/older/2604.pdf

[14] Urja Khurana. 2017. The Linguistic Features of Fake News Headlines and Statements. (2017). https://esc.fnwi.uva.nl/thesis/centraal/files/f1840275767.pdf
[15] Zhiwei Li, Bin Wang, Mingjing Li, and Wei-Ying Ma. 2005. A Probabilistic Model for Retrospective News Event Detection. (2005). http://delivery.acm.org/10.1145/1080000/1076055/p106-
[16] MNCA 2017. Make News Credible Again. (2017). Retrieved October 2017 from https://makenewscredibleagain.github.io/
[17] Veronica Perez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2017. Automatic Detection of Fake News. (2017). https://arxiv.org/pdf/1708.07104.pdf
[18] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake News Detection on Social Media: A Data Mining Perspective. (2017). https://arxiv.org/pdf/1708.01967.pdf
[19] Vivek Singh, Rupanjal Dasgupta, Darshan Sonagra, Karthik Raman, and Isha Ghosh. 2017. Automated Fake News Detection Using Linguistic Analysis and Machine Learning. (2017). brims.org/2017/proceedings/papers/challenge_papers/AutomatedFakeNewsDetection.pdf
[20] Wim De Smet and Marie-Francine Moens. 2009. Cross-Language Linking of News Stories on the Web using Interlingual Topic. (2009). http://class.inrialpes.fr/pub/desmet-swsm09.pdf
[21] Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some Like it Hoax : Automated Fake News Detection in Social Networks. (2017). https://arxiv.org/pdf/1704.07506.pdf
[22] William Yang Wang. 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. (2017). https://arxiv.org/pdf/1705.00648.pdf