Alternus Vera: TheMeanSquares

Pranav Lodha
Charles W.Davidson College of Engineering
San Jose State University
San Jose, USA
pranav.lodha@sjsu.edu

Subarna Chowdhury Soma
Charles W.Davidson College of Engineering
San Jose State University
San Jose, USA
subarnachowdhury.soma@sjsu.edu

Wasae Qureshi
Charles W.Davidson College of Engineering
San Jose State University
San Jose, USA
wasae.qureshi@sjsu.edu

Jeyasri Subramanian
Charles W.Davidson College of Engineering
San Jose State University
San Jose, USA
jeyasri.subramanian@sjsu.edu

***Abstract*—We will focus on describing our approach in determining if an article is fake news. We can use machine learning to build out features from minimal information to determine if an article is misinforming or misleading the reader.**

I. Introduction

A major problem in society today, caused by the explosive growth of technology is the spread of fake news. The misinformation can lead to risks to all wakes of people's lives, communities, nations, and the world as a whole. With the expanded scope of fake news, it has become increasingly important to use technology, specifically machine learning and artificial intelligence in-order to detect and curb the spread of misinformation.

II. Sentiment Analysis

We started the project by looking at sentiment analysis using NLTK vader. vader created a sentiment analysis package using polarity, which helps to classify text as positive, negative, or neutral. The sentiment that we get tells us the percentage value, this allowed us to learn about the feelings of each document. The final score was the compound score, we kept this value for us to potentially use later in the project. From this we were able to determine a baseline of how the sentiments of fake articles differed from that of true articles. This would also be useful later on with our other factors, as those would use the results from the sentiment analysis to further their predictions. We used a mix of logistic regression and tf-idf as part of our pipeline to build the model. This yielded an accuracy of 94%.

III. LDA Topic Modelling

LDA was used for topic modeling and to cluster topics that were similar together. We created a function that would build a dictionary, and tokinze the documents for LDA. The dictionary was created using gensim, and it would filter items that were visible in less than 15 documents or more than 0.5 documents. After that we kept the first one hundred thousand tokens.

For each of the documents we had we created a dictionary to show how many words and how many times those words appeared in our corpus. We then merged this with our bow corpus, which was a list of tokenized docs of all the words in the text. We used tf-idf to build a tf-idf corpus of that document, which helped us determine the keywords in a fake and real article. Given the bow corpus, we run LDA to get the top 10 topics. Using this we yielded an accuracy of 32%.

IV. Sensationalism

Our sensational word corpus, allowed us to determine the verbs that were in our text. Using the actions we wanted to determine the usage of verbs in fake articles as well as real articles. We did this using our results from our LDA, and a dictionary of 1400 words that were positive or negative. We used cosine similarity to determine the distance between the text and these words.

We also enriched our data using senticnet dataset. which provided us polarity associated with natural language concepts.

Then we compared the sentiment of each document, and calculated the similarity score, saving it as another column.

After this we determined the accuracy of this model was not substantial at only 3%.

V. Political Affiliation

An article's political affiliation can tell us a lot about if its fake or real news. In the United States, two political parties determine most of the outcomes. Focusing on this we wanted to see what affect someone's political affiliation would have.

We looked at the headline of each of the articles to check the keywords in the headline. Using these keywords we checked a dataset to determine which party affiliation the articles were associated with. We ran logistic regression on this and it resulted in an accuracy of 56%.

VI. Clickbait

We used two datasets, one with clickbait headlines and another without clickbait. We would use tf-idf to determine the difference between our datasets and the headlines. Using those results we would use logistic regression and end up with an accuracy of 97%. Leading us to believe that the results must be overfitted.

VII. Spam

Using a spam dictionary and count vectorizer. We would determine if an article was spam. We also have another sensationalism dataset for testing and training. For the classification we used a multinomial naive bayes model, which would give us an accuracy of 56%.

VIII. Author Credibility

Similar to political affiliation the author's credibility plays a large part in determining if an article is fake or real. Using the author count and veracity, we would be determining the status of the author's credibility. We add the veracity and author count to make our determination. Our classifier logistic regression, yielded a result of 73%.

IX. Source Reputation

A source's reputation is a key part of an article, we would obtain a dataset of fake news sites with their reputation. We would then hot encode the values as 1 for fake news, and 0.5 for some fake news. For each site we would then have a fake score. If a site matched the dataset, it would return the value of 0.5 or 1.

X. Content Length

Using the original dataset, and the text, we wanted to gather the body length score. We used logistic regression and text length and the veracity values to determine the body length score. We achieved a 24% accuracy.

XI. Word Frequency

This was a simple vectorizer to determine the occurrence of keywords. We used tf-idf on the text and logistic regression to determine the word frequency score. This yielded a 51% accuracy.

XII. Neural Micro-Patterns-of-Misinformation

Neural Fake News is any piece of fake news that has been generated using a Neural Network based model or to define it more formally. What truly differentiates Fake News from original is the fact that it carefully mimics the "styles" and "patterns" that real news usually follows. Neutral fake news is any piece of fake news that can be generated using Neural network models such as GPT-2 by openAI ,BERT by Google and Grover by AllenNLP. We investigated with micro information in real versus fake news by extracting entities, frequently appearing phrases, anomalies in true and fake news and finally measuring the distance between true article versus fake article, true article versus true article, fake article versus fake article. We have used ISOr pre-labeled true and fake news dataset with near equal volume of real and fake news data.

In Iteration1, we begin with the Encoder-Decoder model to predict the anomaly data from the news article. Even with 50k records of news articles, the results aren't too promising in predicting the anomalies. The model of overfitting and distance similarities doesn't yield results as expected. As per the guidance from Advisor, I began distilling fine-grained feature vectors from news articles. About ISOT Dataset, the dataset has only minimal features such as new article headline, text and type of news. This information is not helpful much for anomaly detection.

In iteration 2, I have included more feature vectors from the news article to enrich the data. Document to Vector embedding includes 10 high frequency vectors to each article. I have applied binary classification to this data volume. Initially the prediction results were good with high accuracy for all models as shown below (Fig 11 a).

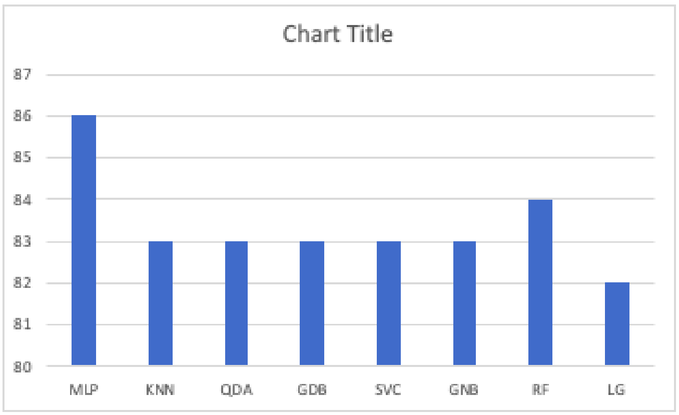


Fig 11 (a) Accuracy graph for binary classification

Second level of distillation including LDA topics modeling and Doc2Vec embedding to LDA topics affected the performance and the accuracy got reduced below 60% as show in the graph below Fig 11 (b)

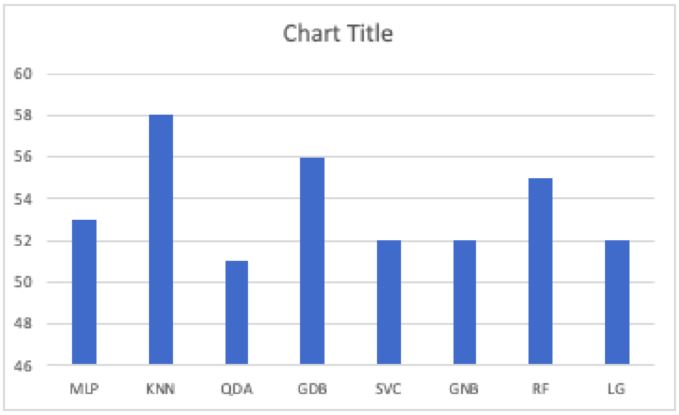


Fig (b) Accuracy for Distillation level 2

As the news dataset doesn't include any feature on spam on sensationalism, I have included sentiment analysis factors for this factor analysis. Compound, Positive, negative, neutral labels from Vader Sentiment analysis is included in the feature vector for pattern prediction. Now the feature vector is enriched with 24 vectors for each article to predict the pattern anomaly. The accuracy of the model is improved above 60% but not as great as Doc2vec embeddings, ref. Fig 11 (c)

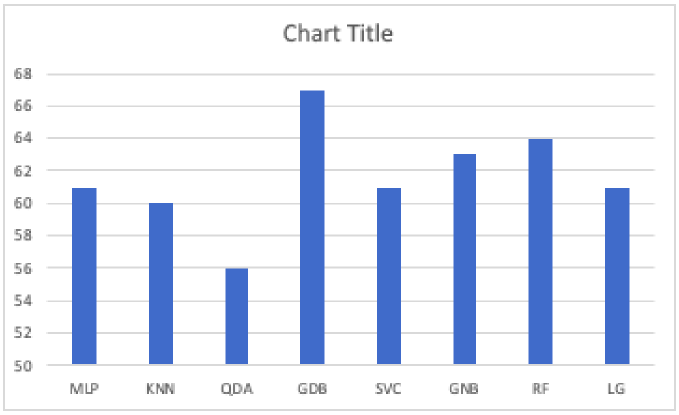


fig (c) Accuracy for LDA Topics, Doc2Vec embeddings and Sentiment analysis.

In iteration 3, As per the direction from the advisor. I have updated the logic to verify multi-class prediction using the Liar Liar dataset with 6 classes. The similar technique of Doc2Vec, Word embedding and sentiment analysis is applied to LL Dataset and multi-label classification doesn't yield good accuracy, ref Fig 11 (d)

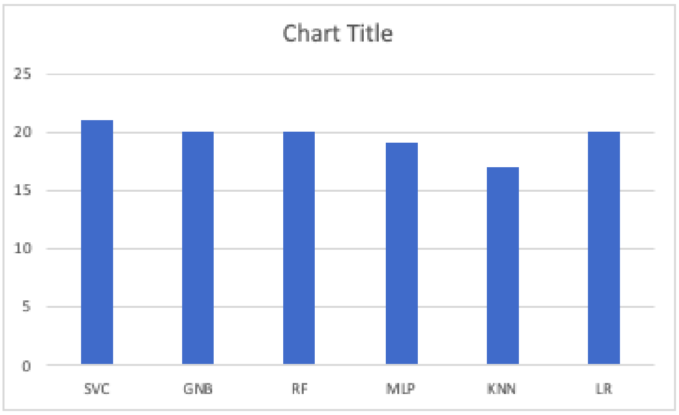


fig 11 (d) Multi-label classification with Liar Liar dataset

Dataset enrichment: We have applied amalgamation technique with KNN classifier and distance similarity, but after multiple iteration, the articles in ISOT dataset does not seem to match with Liar Liar dataset. The maximum distance achieved was 0.54. This execution took more than 12 hours but the results were not promising. Hence at the final iteration, we are continuing with the existing dataset and Liar Liar dataset inference to predict the multi-label classification.

Identifying a dataset with labeled true and fake news is highly challenging. Liar Liar dataset with highly enriched true and fake news information is having only 12k rows of record, that leads to overfitting problem. ISOT dataset with equal count of real and fake data of over 50k records, doesn't have any additional columns of information other than news titles. This model distills the fine grained information just from the news article and is able to predict the fake news given equal volume of true news to compare.

XIII. News Coverage

News coverage means an important factor for fake news detection. Definition wise this factor investigates coverage of the same story in different media outlets. To detect 'News Coverage' we have to measure and compare news coverage of the same story published in different news sources within the same timeframe. For this we have used Kaggle all news data sets. There is no label in this data set to group articles with the same story. So, we have clustered the data into the same group based on their story. To clarify some terms:

- ARTICLE - a single article printed by one news publication
- STORY - the underlying event that an article is in reference to

Preprocessing and data cleanup are done using NLTK to perform different NLP preprocessing on data, for example, removing stopwords, removing special characters, remove punctuations, lowercase, stemming, lemmatization, removing junk and "briefing" articles, removing Daily Summary articles, removing non-ASCII characters etc. After that we have used TF-IDF based clustering. We have generated a TF-IDF matrix with the computed TF-IDF term value constraints. After that processing of the resulting vectors are done to rank their affinity. Finally grouped articles whose affinity is above the given threshold into "stories". After the first set of results I have picked some of the small cluster group data to do a manual story naming and saved it inside a file. Later I have used the file to evaluate the cluster, to visualize the cluster above using T-sne and for scoring news coverage.

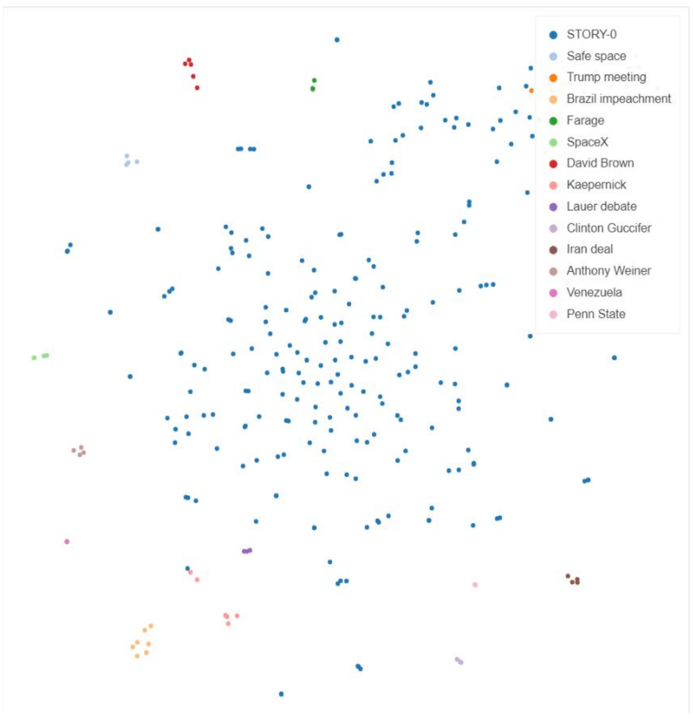


Fig (a) Clustering result of Kaggle All news dataset

To measure the news coverage, we have used sentiment score as one of the measuring criteria. Sentiment is aggregated to compute the coverage of the story also. Next metric is measuring the 'balance' of coverage. It is done by identifying the top point of views (key words). Each point on the sentiment scale represents a distinct point of view. These two terms are then averaged to measure how well balanced the overall coverage of the story is. A perfect score for the aggregate coverage of a story is 1. As this data set is unlabeled, so I have to label this data to implement a classification model. We have a hypothetical parameter to identify 'good/bad' [1/0] coverage based on news coverage score. This value is stored in the 'label_NewsCoverage' column. I have fine-tuned the threshold multiple times. Finally, we have used auto ML to try different classification algorithms to compare accuracy. The Logistic Regression Model has performed well with 75% accuracy. This was the first level of news coverage score computation. Following is the figure of kaggle all news data word cloud:

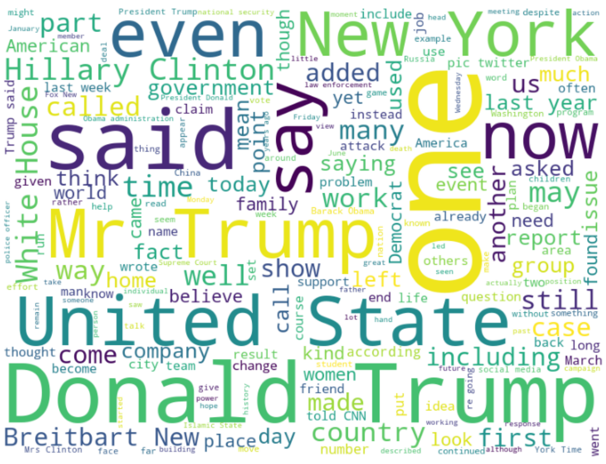


Fig (b) Kaggle all news data word cloud

In the second level of iteration, we talked to our advisor Professor Arsanjani, about how the news score computation can be used for detecting different levels of fake news detection. Following his guidance, we have introduced another dataset called Kaggle fake news dataset. First, we used topic modeling to find the key latent topics or stories from Kaggle fake news data sets. Then we use the dataset "All the news" to see if there are other articles covering a similar story. It is important to note that we are not cross-checking factual claims made in the article, but rather checking to see if a related story and latent topic can be found in other news sources within a time frame. If there is no external coverage for the story/topic, chances are that the article in question is likely a fake news. However, reliable news doesn't solely depend on sufficient news coverage. This week we also performed LDA separately on all the datasets that we are using for computation.

In the third level of iteration, we computed other 10 factors for fake news detection listed down above. Then we have enriched our dataset and broadened news coverage computation by combining fake news and all news data sets. Both the dataset has a publishing date column but only fake news data sets are labeled for classification. We have applied all types of required distillation processes called: tokenization, lemmatization, stop word removal etc. on both the data sets. End result is some latent topic or story parsed each dataset. To make an appropriate scoring coverage, the latent topics between "all news" data set and fake news dataset (`data_fake_news`) are compared using cosine similarity with a time coverage time frame. The coverage time frame is simply a time-range where two articles are considered to be pushed in the same approximate "time frame". Narrowing the window size will result in a low coverage score across all rows. Increasing the window size will result in a high coverage score across all rows. The choice of the window size was arbitrary (30 days).

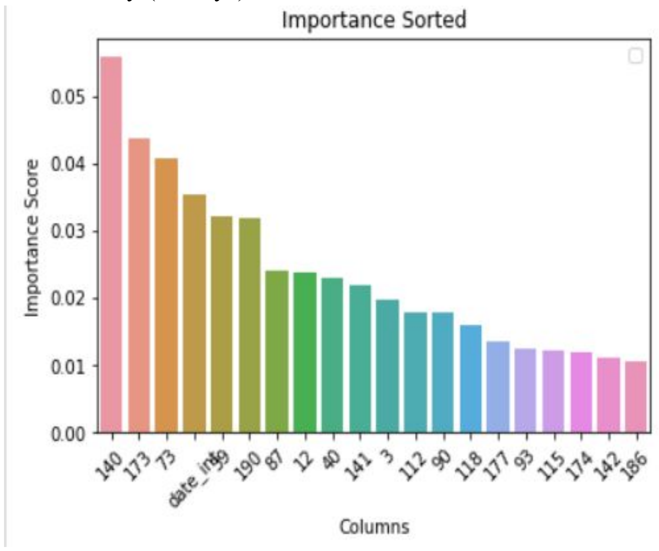
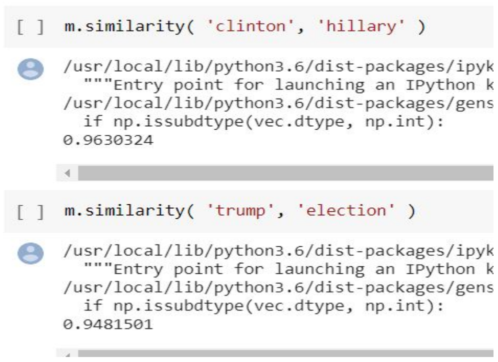


Fig (c) Important features (numeric-word2vec) sorted

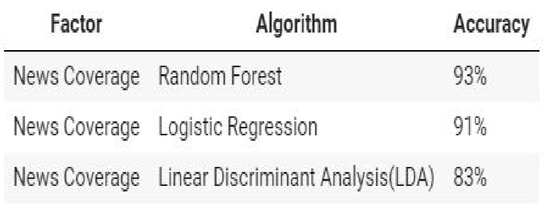
Fig (c) represents the most important feature in determining whether or not the news has high news coverage. This behavior is expected since published dates surely place an important role and have a significant impact in prediction of the news coverage of a story.



```
[ ]  m.similarity( 'clinton', 'hillary' )

     /usr/local/lib/python3.6/dist-packages/ipyk
       """Entry point for launching an IPython k
     /usr/local/lib/python3.6/dist-packages/gens
       if np.issubdtype(vec.dtype, np.int):
     0.9630324

[ ]  m.similarity( 'trump', 'election' )

     /usr/local/lib/python3.6/dist-packages/ipyk
       """Entry point for launching an IPython k
     /usr/local/lib/python3.6/dist-packages/gens
       if np.issubdtype(vec.dtype, np.int):
     0.9481501
```

Fig (d) High similarity between some topics between two datasets

Later for better coverage score the window is increased from 30 days to 60 day. To assign coverage score, the topic "label" for each fake news article is searched in the "all news" dataset. For each label matches using topic similarity, we increment the coverage score by 1. However, if the publication dates of the two news articles (one from a fake news article, and one from the "all news" dataset) do not fall in the same time window (30 days or 60 days), no points are awarded to the coverage score from that particular match. If no news articles match, the coverage score would be assigned a score of 0 accordingly. After that we have performed sentiment analysis and used word2vect to convert topics to vector embeddings. Ultimate goal is to now build a model for news coverage computation. Again, I have performed auto ML to find the best performed algorithm for my model. Random forest algorithms along with Logistic regression and Linear Discriminant Analysis (LDA) have performed well for my model with max 93% accuracy.

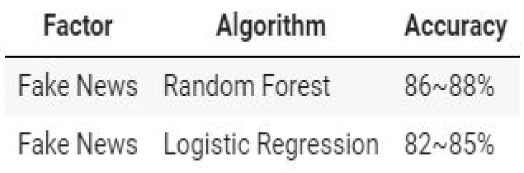


Factor	Algorithm	Accuracy
News Coverage	Random Forest	93%
News Coverage	Logistic Regression	91%
News Coverage	Linear Discriminant Analysis(LDA)	83%

Fig (e) Model performance for news coverage computation

Observing by comparing and analyzing the news coverage score for fake and non-fake news, we have found that 30 out of 100 non-fake news articles had coverage greater than one. On the other hand, only 14 out of 100 fake news had coverage scores greater than one.

In the final iteration of this project, we had multiple goals to achieve based on feedback from our advisor and to make a conclusion of this project. First, we identified that the fake news is highly skewed toward the 'bs' classification label [13,000 news with labels of: 88% 'bs', 3% 'bias' and 8% 'others']. To fix this problem, we performed LDA topic modeling on 'bs' data and non 'bs' data separately. Then compared the topics parsed from 'bs' data with the topics of 'non-bs' data and assigned to the closed or similar class. This process went on up to a threshold and in this way, we resolved skewed 'bs' data partially. With this refined data we have again done all the news coverage computation and executed all the steps explained above. After that we have amalgamated the all-news and fake news data and again performed all the computation and classification model building to verify my previous work. Now the next step is to compute news coverage of the Liar Liar dataset. One major challenge was the absence of a publishing date inside the liar data set. So, no time frame could be set like before to compare the news between liar liar and all news stories or topics parsed from both the datasets. Building such a 'search' model is a very expensive operation in terms of time and resource, so only be done during the training stage. After that we saved the model for testing and validation. Therefore instead of doing the search for every article, we sorted the topics based on weighted probability and news coverage is computed based on that slight required modification. Earlier we tried binary classification to detect fake or non-fake news using our factor. Later when requirements changed to multi class classification, we first tried this multiclass classification on Liar Liar data set using different approaches. Finally using news coverage score we have built a model with following accuracy to detect fake news:



Factor	Algorithm	Accuracy
Fake News	Random Forest	86~88%
Fake News	Logistic Regression	82~85%

Fig (f) Model performance for fake news detection

After all the computation, we saved the models to use for our team Colab to build a polynomial equation for detecting fake news along with other factors.

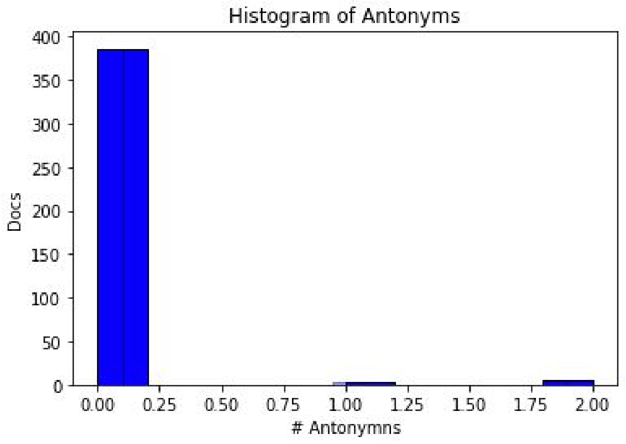
Apart from the above explained methods, we tried some approaches with different data sets for news coverage computation which did not work. As a team we collected financial news data using API from Newyork Times and Reddit News for our first Machine Learning project. At first, we tried to use that news collection, but that dataset has only two news sources. For news coverage I need a data set with multiple sources covering the same story. Later we found Kaggle "All News" dataset which contains news of various domains from 15 news sources. So we proceeded with Kaggle dataset for data enrichment and amalgamation.

XIV. Bias

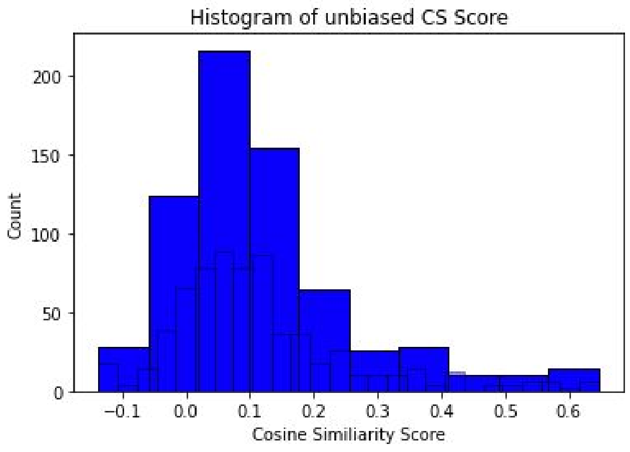
Bias is common among authors in order to get readers to share an opinion, confirming what they already believe or sway readers to changing their opinions. Here we tried to see if we could spot this bias given any text.

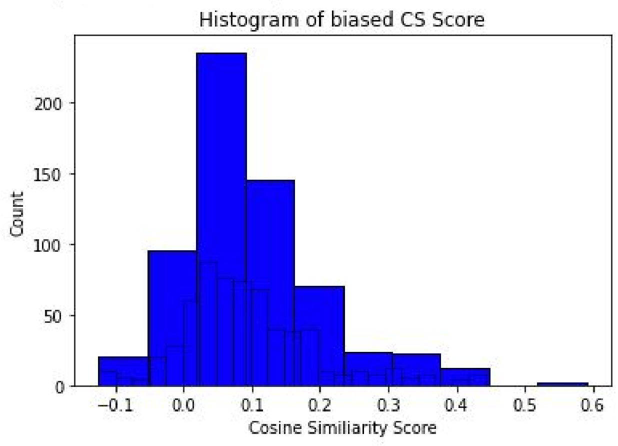
In our initial research, we looked into using the toxicity dataset (Conversation AI) to determine bias. For the first iteration, we used the words from the toxic dataset and changed the label from toxic to bias, not toxic to unbiased. This was a naive assumption because we can't assume toxic words don't occur in unbiased articles. We moved away from this but we will come back to this later.

We talked to our advisor in the second iteration about how we could find variation in the text. The idea here was if there was less variation in the content, the content was more prone to being biased, and vice versa. The first technique we discussed was if we got a single topic of a document, could we check if a word in the topic also had it's antonym in the list of words for that topic. Using this information and by setting some sort of threshold for the number of antonyms found, we could identify if there is more or less variation in the article and if it is susceptible to bias. We used a random news dataset just to see what results we would get using this technique. We plotted a distribution for the number of docs and number antonyms and to our surprise found that the majority of the docs had zero antonyms. We assumed at this point that using antonyms to find variation would not be vital in determining bias.



We went back to the drawing board with our advisor and decided to change our technique. We concluded from the previous iteration that the technique of checking variation, using antonyms, was the issue. In this next iteration we switched our technique to using cosine similarity. We could use this similarity score on the words in the topics to identify bias. In addition we used a set of fixed biased articles (from the fake news data set) and unbiased articles (from the BBC news data set, subject to our opinion about BBC which is prone to bias). This way we could have baseline ground truth for biased or unbiased articles. We created a distribution of the cosine similarity score and the number of docs that score occurred in biased and unbiased articles. We got the following results:





The issue we had was the distribution was identical for both unbiased and biased articles, making it difficult to distinguish between the two.

Due to the limited time, we stopped here and captured our findings. We realized that we need to first make sure that there is a better way to capture the key things the article is talking about in order to apply those techniques to see if the documents have variation. Another thing we could try next time is to use the toxicity dataset as a dictionary. Toxicity does play a role in bias and we could add it as a feature for classification.

XV. Credibility and Reliability

Credibility and Reliability of a news source is a question that needs to be answered in-order to determine if an article is fake news. Generally, the same sources tend to publish fake news.

To determine the credibility and reliability of a source, we looked towards a prelabeled dataset from an open sources site. This prelabeled dataset contained information of 1000 websites with labels such as fake, satire, bias, conspiracy, state, junksci, hate, clickbait, unreliable, political and reliable.These sites would be used to create a baseline for our model.

From that dataset we determined that the reliable sources were too little as we only had 8 out of 1000 reliable sources. To overcome this problem, we decided to use the unreliable sources as our baseline as we had over 300 sites in that category. We decided to use the unreliable sites as the main part of our model.

We then used open pagerank, a site that catalogs the page rank and page rank score of sites on the internet. Using the API we were able to get the page rank and page rank score (score from 1 -10) for each of our 1000 sites. With 1 being unpopular and 10 being popular. For example, google. com has a page rank score of 10, making it a very popular site. We would be making the assumption that the more popular site would be much more credible and reliable.

After we had the page rank score of the 1000 sites, we would find the average page rank score of our eight reliable sources, and the average score of our 300 unreliable sources. The reliable sources had an average of 5.6 out of 10, while the unreliable sources had an average of 4.51.

With that information we would also be checking the spelling of the article. We noticed that articles that were fake, tended to have a lot of mis-spellings. So to improve our credibility and reliability factor we would use a mix of the page rank score and the spelling.

Once we had determined if a site had a page rank score above 5.6, we would check the spelling and if the spelling was correct for 85% of the page, we would classify the site as credible and reliable.

From this we built our model and would classify articles as credible and reliable if they had a page rank score above 5.6, as well as spelling better than 85%. Any site below that would be classified as unreliable and uncredible.

Using these two metrics we were able to determine if a source was credible or reliable. We provide the source, and then use the API to determine the page rank score, we then check for spelling and then use these two metrics to determine if the source is reliable. Using this technique we were able to achieve a performance of 60%. Our thought process was that the performance would be much better, but the difference between the page rank score of the articles was much too little, as well as page rank not being a good enough indicator of credibility and reliability.

XVI. Final Model

We started the project by looking at sentiment analysis using NLTK vader., LDA topic modeling, sensationalism, spell check and spam score from Liar Liar dataset. After deep investigation into the key factors such as Neural micro-patterns of misinformation, News coverage, Credibility and reliability and Bias in news articles, we distilled key factors about fake data in new articles.

From the key factors, we computed a classification model for multi-label such as true,mostly true, half true, barely true, false and pants on fire classifications from Liar Liar dataset. The accuracy score of the above listed classification models helped us to determine a combination model with the following equation.

Classification model = 0.88 * (News coverage factor) + 0.67 * (Micro-pattern of misinformation)+(0.60 * credibility and reliability factor) + (0.5 * Bias factor). This yielded an multi label accuracy of 14.9%.

References

[1] 5. Categorizing and Tagging Words, www.nltk.org/book/ch05.html.

[2] Dwivedi. “Real vs Fake Tweet Detection Using a BERT Transformer Model in Few Lines of Code.” *Medium*, Becoming Human: Artificial Intelligence Magazine, 12 Mar. 2020, becominghuman.ai/real-vs-fake-tweet-detection-using-a-bert-transformer-model-in-few-lines-of-code-ccc33ecb1a2.

[3] “Gensim: Topic Modelling for Humans.” *Radim ÅEhÅ¯ÅEk: Machine Learning Consulting*, radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html#sphx-glr-auto-examples-tutorials-run-doc2vec-lee-py.

[4] Kapadia, Shashank. “Evaluate Topic Models: Latent Dirichlet Allocation (LDA).” *Medium*, Towards Data Science, 19 Aug. 2019, towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0.

[5] Kapadia, Shashank. “Topic Modeling in Python: Latent Dirichlet Allocation (LDA).” *Medium*, Towards Data Science, 5 Sept. 2019, towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0.

[6] Kavita Ganesan. “How to Incorporate Phrases into Word2Vec - a Text Mining Approach.” *Kavita Ganesan*, 20 Mar. 2020, kavita-ganesan.com/how-to-incorporate-phrases-into-word2vec-a-text-mining-approach/#.Xpp1apNKhYh.

[7] Kelechava, Marc. “Using LDA Topic Models as a Classification Model Input.” *Medium*, Towards Data Science, 13 Apr. 2020, towardsdatascience.com/unsupervised-nlp-topic-models-as-a-supervised-learning-input-cf8ee9e5cf28.

[8] Kelechava, Marc. “Using LDA Topic Models as a Classification Model Input.” *Medium*, Towards Data Science, 13 Apr. 2020, towardsdatascience.com/unsupervised-nlp-topic-models-as-a-supervised-learning-input-cf8ee9e5cf28.

[9] Ktattan. “LDA and Document Similarity.” *Kaggle*, Kaggle, 15 Oct. 2017, www.kaggle.com/ktattan/lda-and-document-similarity.

[10] Ktattan. “LDA and Document Similarity.” *Kaggle*, Kaggle, 15 Oct. 2017, www.kaggle.com/ktattan/lda-and-document-similarity.

[11] Li, Susan. “Topic Modeling and Latent Dirichlet Allocation (LDA) in Python.” *Medium*, Towards Data Science, 1 June 2018, towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24.

[12] Li, Susan. “Topic Modeling and Latent Dirichlet Allocation (LDA) in Python.” *Medium*, Towards Data Science, 1 June 2018, towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24.

[13] Mishra, Deepak. “DOC2VEC Gensim Tutorial.” *Medium*, Medium, 30 Aug. 2018, medium.com/@mishra.thedeepak/doc2vec-simple-implementation-example-df2afbbfbad5.

[14] “Natural Language Processing (NLP) Techniques for Extracting Information.” *Search Technologies*, www.searchtechnologies.com/blog/natural-language-processing-techniques.

[15] Rizvi, Mohd Sanad Zaki. “An Exhaustive Guide to Detecting Neural Fake News Using NLP.” *Analytics Vidhya*, 16 Dec. 2019, www.analyticsvidhya.com/blog/2019/12/detect-fight-neural-fake-news-nlp/.

[16] several27. “several27/FakeNewsCorpus.” *GitHub*, 25 Jan. 2020, github.com/several27/FakeNewsCorpus.

[17] Shrestha, Anish. “Fake News Classification Using GLOVE and Long Short Term Memory (LSTM).” *Medium*, Medium, 18 Dec. 2019, medium.com/@sthacruz/fake-news-classification-using-glove-and-long-short-term-memory-lstm-a48f1dd605ab.

[18] Sieg, Adrien. “Text Similarities : Estimate the Degree of Similarity between Two Texts.” *Medium*, Medium, 13 Nov. 2019, medium.com/@adriensieg/text-similarities-da019229c894.

[19] Theiler, Sebastian. “Basics of Using Pre-Trained GloVe Vectors in Python.” *Medium*, Analytics Vidhya, 5 Jan. 2020, medium.com/analytics-vidhya/basics-of-using-pre-trained-glove-vectors-in-python-d38905f356db.