ECE 684 Natural Language Processing Final Project
Evan Glas

Predicting 2020 County-Level Election Outcomes via Covid-related Excerpts from Local News

Evan Glas

Abstract

In recent years, United States news outlets have received increasing criticism for political bias. This report seeks to test the hypothesis that political polarization affected reporting of the COVID-19 pandemic in local news outlets. A dataset of samples of excerpts of textual media was collected and labeled according to each locality's voting results in the 2020 presidential election. Several models were trained and tested on this dataset, including a Naïve Bayes model, a Logistic Regression, a lightweight neural network, and an LSTM. The Naïve Bayes model was then used to generate an artificial dataset over which each model was evaluated once more. The models demonstrate reasonable performance on out-of-sample data, suggesting a possible link between US counties' political lean and representation of the pandemic in local media.

Introduction

Motivation

Potentially a reflection of the country's rising political polarization, news outlets have been known to cater to their viewer's political lean to drive media consumption. Left unchecked, this tendency presents a threat to the integrity of the United States' collective media environment. United States democracy depends on an informed public to drive policy decision, thus the media's capacity to present unbiased news is vital to the country's governance. Political bias in the United States media was perhaps especially evident during the COVID-19 pandemic [1]. At a time when the public was most in need of reliable information to navigate unprecedented times, media outlets often presented contradictory narratives of the pandemic's severity, up-to-date scientific understanding, and government response. The widespread nature of political bias within news coverage may have facilitated individuals to restrict themselves to media outlets that reinforced their preexisting views. Taken in combination with social media, it is likely that traditional media outlets played a significant role in the nation's heated divide and fractured response to the pandemic.

This project seeks to determine whether such a bias may exist in local news sources consistent with county-level voting data. If a classifier can predict a county's voting results according to text documents derived from local news sources, this may indicate an underlying difference in reporting dependent on the political lean of the news sources' respective readership. This may be problematic as, in an ideal world, each media outlet would have reported on the pandemic from an "unbiased" perspective indistinguishable, in distribution, from other news sources. If there exist systemic differences in local news outlets' reporting, this may have worsened the geographic political divide on the pandemic issue on a local level. Such a result would potentially call into question the readiness of US media to properly report on future crisis-level issues, notwithstanding everyday political content.

Document Classification Models

There exist numerous natural language processing methods to classify text documents. Put briefly, the task of document classification seeks to assign a label or class to of some length of text. Depending on the application, different methods may be preferred given differences in accuracy, speed, interpretability, or other qualities. As described below, there exist two broad classes of models that may be applied to document classification: generative and discriminative models.

Generative Probabilistic

Generative probabilistic models seek to learn the joint distribution of documents and labels. Given a corpus of text, a generative model will attempt to learn the underlying distribution from which the text was generated. As it is impossible to exactly quantify the distribution from which humans generate textual data, generative models must apply certain numerical assumptions to approximate a desired distribution. Since generative models learn an approximation of $\mathbb{P}(\text{Document}, \text{Label})$, generative models hold the ability to generate documents with high probability under the learned distribution. This is a feature that may be valuable especially under conditional generation in which the model seeks to generate text given external information such as a prompt, context, or question. Generative language models have gained widespread publicity in recent years such as OpenAI's GPT class of models or Google's Bard.

Naïve Bayes

A Naïve Bayes model computes document probabilities under the bag-of-words assumption. That is, the model follows the "naïve" assumption that tokens are conditionally independent given the document label. A Naïve Bayes model thus represents a document as an unordered collection of tokens as the assumption of conditional independence removes the importance of word position. Naïve Bayes models are highly interpretable and computationally light-weight, however, the bag-of-words assumption generally fails to capture nuance in natural language. For instance, under the bag-of-words assumption, the documents "I am" and "am I" would be taken to be semantically equivalent, despite their evidently different meanings. Mathematically, a Naïve Bayes models is constructed as follows where C_j is class j and X^n is a document of n tokens:

$$P\left(C_j|X^n\right) = \frac{P\left(X^n|C_j\right)P\left(C_j\right)}{P(X^n)} \tag{1}$$

The above equation may be simplified under the bag-of-words assumption which states the following to be true:

$$P(X^n|C_j) = P(X_1, X_2, \dots, X_n|C_j) = \prod_{i=1}^{n} P(X_i|C_j) \tag{2}$$

$$\Rightarrow j = \arg\max_j P(C_i|X^n) = \frac{\left(P(C_j)\prod_{i=1}^{n} P(X_i|C_j)\right)}{P(X^n)} \propto P(C_j)\prod_{i=1}^{n} P(X_i|C_j) \tag{3}$$

Accordingly, the predicted class is that which maximizes the document probability under the assumption of conditional independence given the class label.

A Naïve Bayes model is a special case of a model class known as n-gram models. n-gram models assume that words are tokens are conditionally independent given the class label and previous $n-1$ tokens in the document. Thus, a Naïve Bayes model is an n-gram model with $n = 1$, also known as a unigram model.

Naïve Bayes models can be used to generate artificial text by drawing words according to the class-conditional distribution of tokens. If one desires to generate the document with maximal probability under the bag-of-words assumption, the model will simply repeat the most likely word given the desired class. Perhaps more relevant, the model can also generate documents stochastically by drawing successive tokens according to the learned class-conditional distribution.

Discriminative

As opposed to generative models, discriminative models seek solely to model the conditional distribution $P(C_j|X^n)$ rather than the joint distribution $P(C_j|X^n)$. Discriminative models can thus be used to find the probability of a class given a document. However, they do not attempt to learn the generating distribution and can thus not be applied to synthesize artificial data. There exists a broad range of discriminative document classification models again with varying characteristics. This project explores the use of logistic regression and two neural-network-based models.

Logistic Regression

A logistic regression is a linear model that maps a document in the form of semantic vectors to class “probabilities”. As is the case for many natural language models, a logistic regression may only deal with input text once the tokens have been converted into quantitative objects or vectors. There exist a variety of ways to embed tokens into vectors with varying degrees of complexity. In this project, I apply two forms of embeddings, one-hot and term frequency-inverse document frequency (TF-IDF) embeddings. One-hot encodings simply convert tokens to an integer index/one-hot vector representation of that index. TF-IDF, on the other hand, considers both a token’s frequency within a document in the context of its frequency in the entire text corpus. TF-IDF may then provide a better way to distinguish the words most relevant to each individual document, which may prove beneficial as input to discriminative models. Given a set of text embeddings, logistic regression may be trained via gradient descent, traditionally using the binary cross-entropy loss function. Logistic regressions are guaranteed to converge to a global minimum as the loss function is strictly convex. Further, logistic regressions tend to train rapidly compared to more sophisticated discriminative models.

Neural Networks

I apply two distinct neural network-based models in my project. Neural networks are a class of functions that can model complex function spaces via a compilation of simple computations. As is the case for logistic regression, neural networks may be trained via some form of gradient descent in which an optimizer iteratively attempts to minimize some given loss function. Neural networks can take on arbitrarily large sizes ranging from dozens of parameters to billions. Given limited computational resources, this project merely considers two relatively small networks. The first network consists solely of an embedding layer and a fully connected layer. The second network appends a Long Short Term

Memory (LSTM) layer. An LSTM is a class of recurrent neural network that may be applied to serial data. An LSTM can be particularly useful when the context of prior tokens may heavily influence the meaning of the current token, as is the case for natural language.

Data

Dataset

This project uses the NELA-Local Dataset, a collection of over 1.4 million online local news articles published between April 4th, 2020 and December 31st 2021 [2]. The dataset contains samples from media outlets in 255 counties and 46 states. For each news outlet, the dataset includes its geographic location and a description among other features. The dataset also includes a mapping of media outlet locations to local demographic data. Relevant to this project, the dataset includes county-level voting histories in the form of log-odds of voting for the Republic candidate in the 2020, 2016, and 2012 elections [3]. This data was then used to construct labels as detailed in the following section.

Data Processing

As the raw dataset contained over 4 billion characters, it was immediately necessary to limit analysis to some subset to feasibly process the text on a laptop. First, samples were dropped if they did not contain any of the words in the following set, $C = \{\text{covid, pandemic, coronavirus}\}$. This way, the new dataset only retained documents with a possible reference to the COVID-19 pandemic. The dataset was again reduced to a series 300-character windows around each occurrence of a word in C. This step was not only beneficial in drastically limiting the dataset size but promoting the inclusion of documents more focused on the pandemic itself. As the untouched articles could be thousands of characters in length (mean $\approx$ 3000 characters) yet contain as little as a single reference to a word in C, it is likely that most of the dataset was not directly related to COVID-19 topics.

Next, the dataset was processed to remove geopolitical entities, people, or organizations that could present confounding variables in the document labels. For example, it was common for an article in the dataset to begin with the name of the city in which the reporter was based. The model could potentially learn the political lean of unseen documents purely by learning the political lean of individual cities rather than the context of the document itself. These elements were removed by tokenizing the documents with the Scapy nlp module [4]. This module provided part-of-speech tagging which, under brief analysis, appeared to effectively label the document tokens. This step in the data processing pipeline took well over an hour given the computationally expensive process of tagging millions of words across thousands of documents. Non-alphanumeric characters were also removed from the dataset and all remaining letters were made lowercase.

Next, document labels were generated using each media outlets' county's voting results from the 2020 presidential election. If a county was in the top quartile of counties in the dataset with a highest proportion of Trump voters, it was given a label of "True". If a county was in the bottom quartile of counties in the dataset with a highest proportion of Trump voters, it was given a label of "False". Without adjustment, the windowed documents leaned heavily towards "False". The dataset was reduced once more until 50000 true documents and 50000 false documents remained.

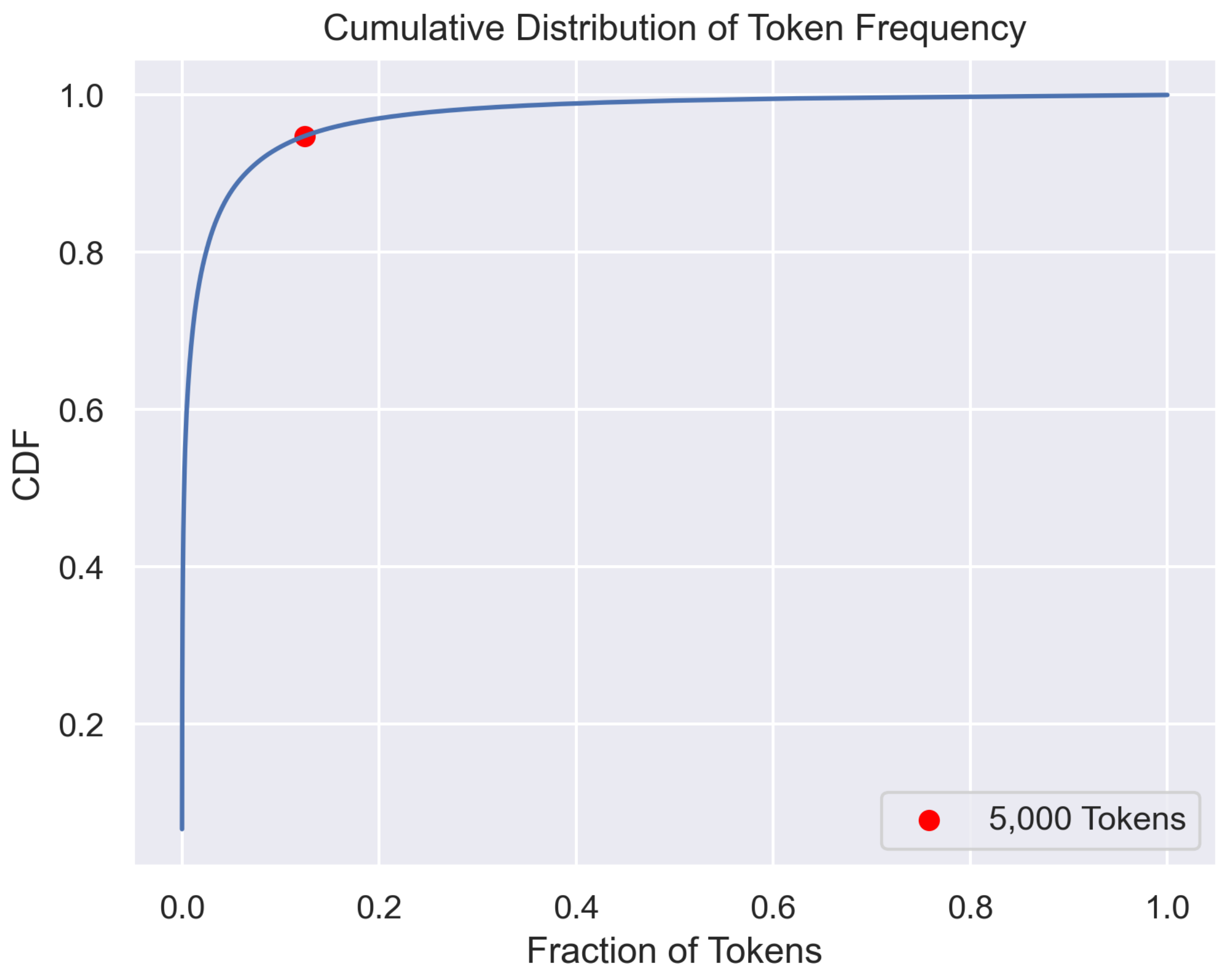
The dataset was divided using an 80-20 train-test split. For use in the neural network-based models, the training set was divided into an 80-20 train validation split.

Methods

Naïve Bayes

The Naïve Bayes models were generated using the scikit-learn MultinomialNB module [5]. The documents were vectorized using the scikit-learn CountVectorizer (for one-hot tokenization) and

TfidfVectorizer modules [5]. Each vectorizer was given a 5,000 token limit to prevent overfitting. As shown in the below figure, the 5,000 top unique words represented about 95% of all words in the dataset.



This model was then used to generate an artificial dataset of 1000 “True” samples and 1000 “False” samples according to the learned condition distribution across the most frequent 10000 tokens. All models were evaluated over this artificial dataset as well as the real test data.

Logistic Regression

A logistic regression was trained on both the word-count and TF-IDF embeddings using the scikit-learn LogisticRegression module. The regression was run for a maximum of 1000 training iterations to limit overfitting. This model contained 5001 total parameters (a weight for each feature as well as a constant term).

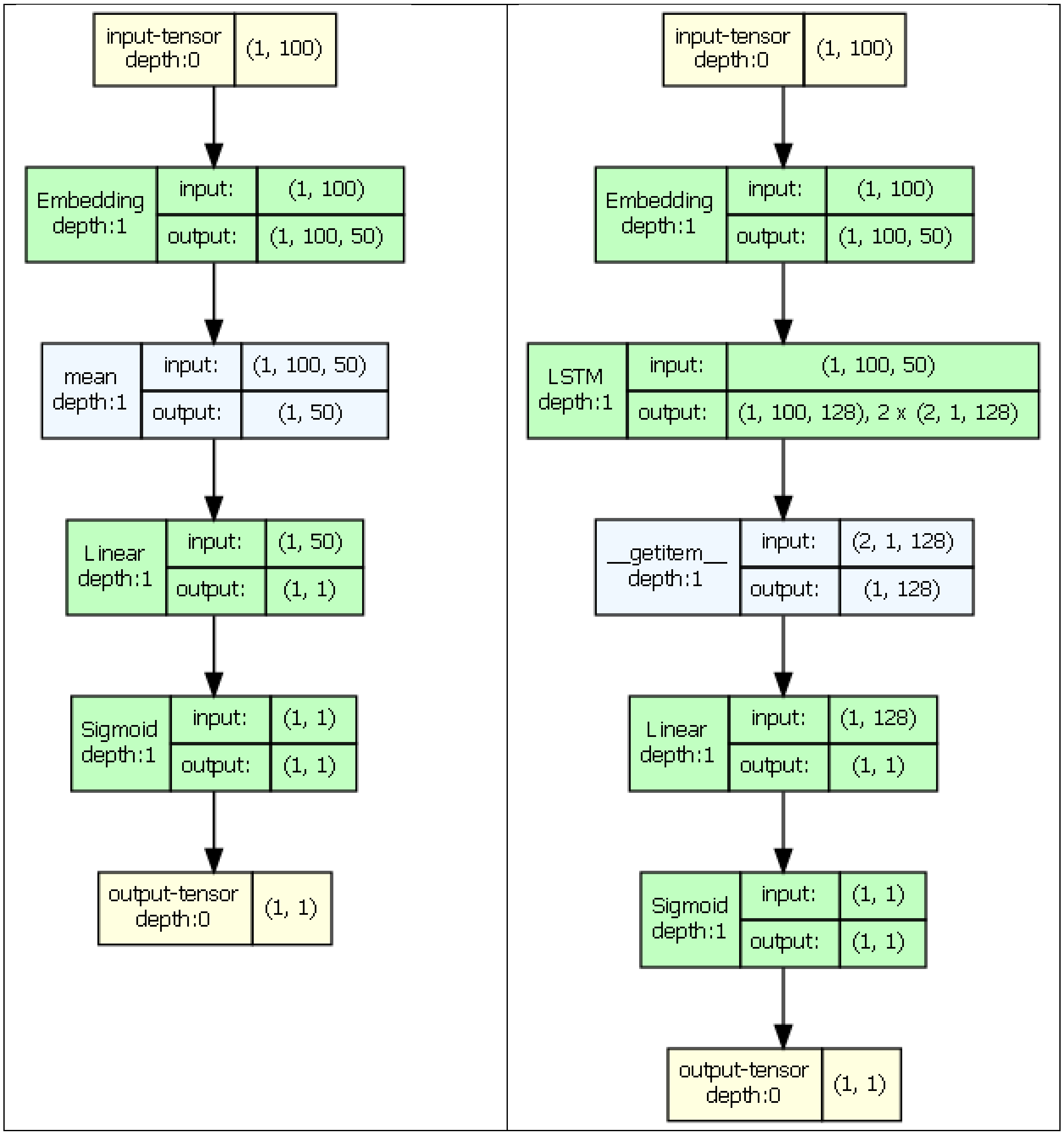
Lightweight Neural Network

A lightweight neural network was constructed using a single embedding layer and a fully connected layer. Input vectors were generated by averaging each individual token vector across a document while also applying necessary padding to standardize the input shape across each batch. The embedding layer projected the tokens onto 50-element vectors. Next, the fully connected layer mapped the embedding to a single output logit. The model contained 250101 total parameters.

LSTM

An LSTM model was constructed by appending two LSTM layers with 128-dimensional hidden layers to the above model. Both lightweight neural network and LSTM models used the binary cross-entropy loss function and Adam optimizer. Both models were trained over 10 epochs. The train and validation losses were recorded for each epoch, and the model with the lowest validation score was ultimately saved to apply to the test set. The LSTM model contained 474435 total parameters. The below figures illustrate the architecture of both the lightweight neural network and the LSTM models:

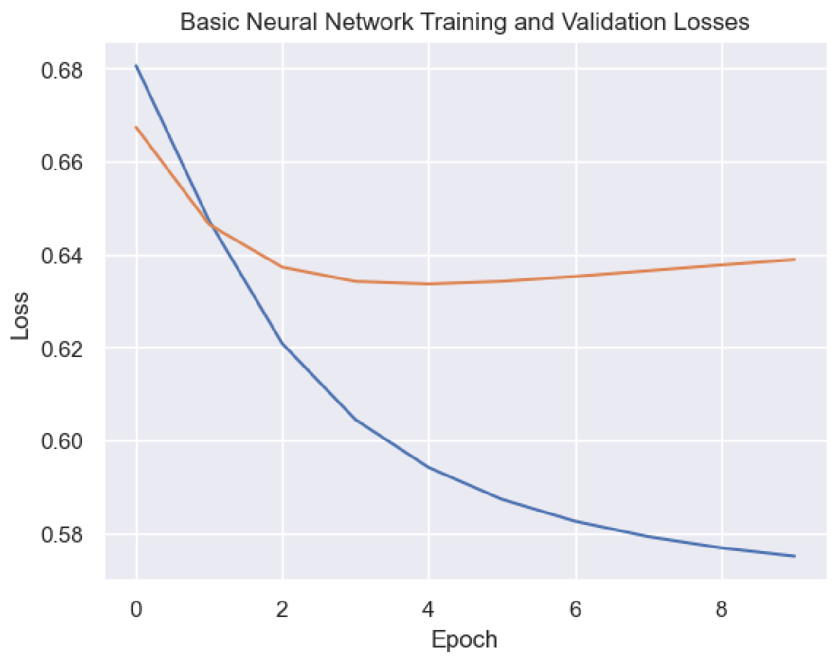
Lightweight Neural Network	LSTM

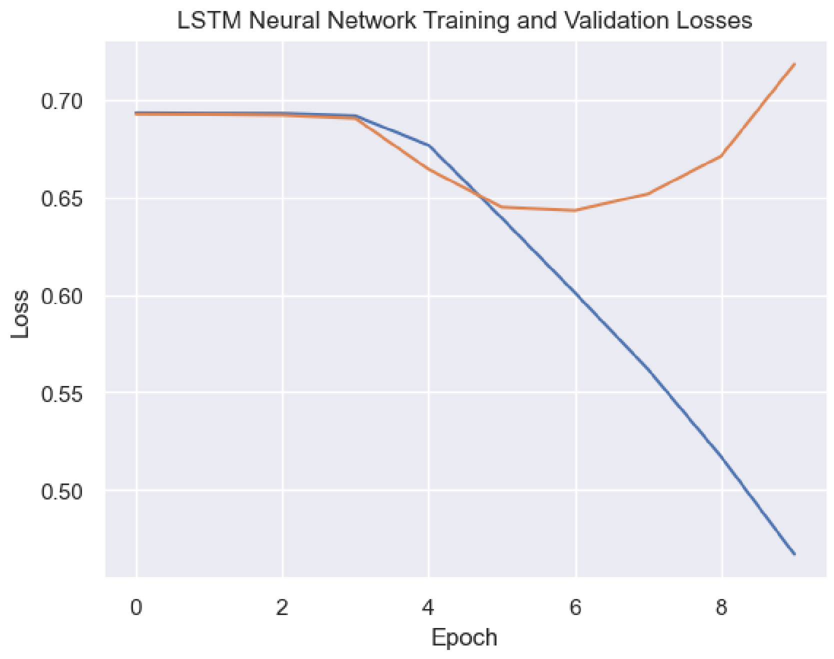

input-tensor depth:0 (1, 100)
Embedding depth:1 input: (1, 100) output: (1, 100, 50)
mean depth:1 input: (1, 100, 50) output: (1, 50)
Linear depth:1 input: (1, 50) output: (1, 1)
Sigmoid depth:1 input: (1, 1) output: (1, 1)
output-tensor depth:0 (1, 1)
input-tensor depth:0 (1, 100)
Embedding depth:1 input: (1, 100) output: (1, 100, 50)
LSTM depth:1 input: (1, 100, 50) output: (1, 100, 128), 2 x (2, 1, 128)
__getitem__ depth:1 input: (2, 1, 128) output: (1, 128)
Linear depth:1 input: (1, 128) output: (1, 1)
Sigmoid depth:1 input: (1, 1) output: (1, 1)
output-tensor depth:0 (1, 1)

Results

Neural Network Training

The below figures depict the training process of the lightweight neural network and LSTM models, respectively. As shown, each model begins to overfit the training data after 4-6 epochs as the training loss continues to decline while the validation loss stagnates or even rises. The validation loss demonstrates a substantial increase in the LSTM model after epoch 6.





Train/Test Accuracies by Model

	Naïve Bayes (Counts)	Naïve Bayes (TF-IDF)	Logistic Regression (Counts)	Logistic Regression (TF-IDF)	Lightweight Neural Network	LSTM

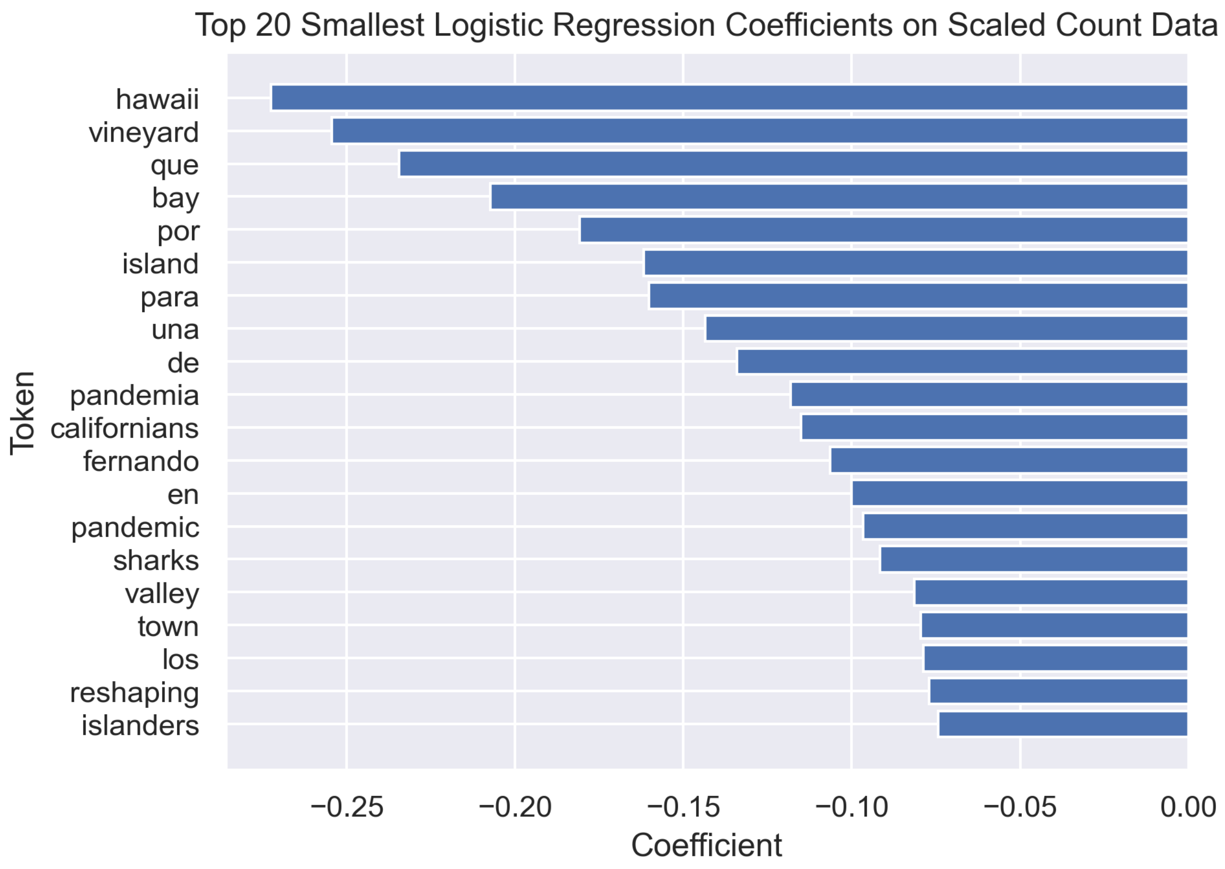
Train Accuracy	0.6528	0.6613	0.6943	0.6916	0.69	0.71
Test Accuracy	0.635	0.6377	0.6466	0.6494	0.64	0.63

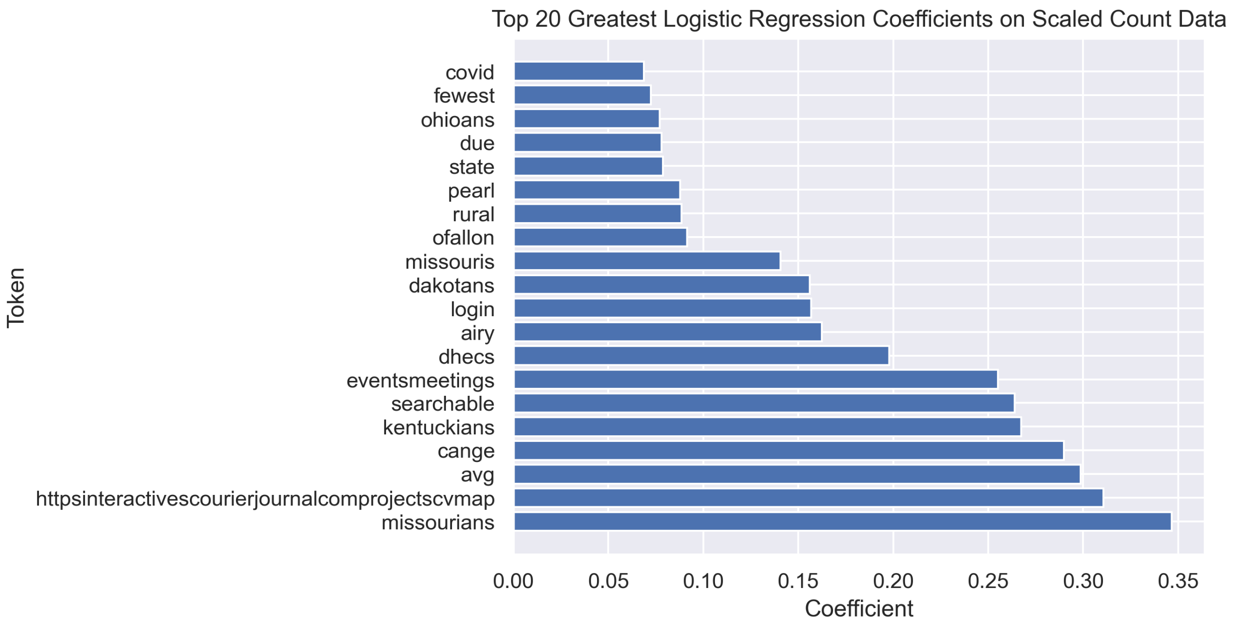
Artificial Data Accuracies (From Naïve Bayes Model) by Model

	Naïve Bayes (Counts)	Logistic Regression (Counts)	Lightweight Neural Network	LSTM
Accuracy	0.8075	0.7425	0.72	0.64

Logistic Regression Coefficients

The below figures provide some insight towards the tokens most closely associated by the logistic regression model with either the “True” or “False” labels. To yield greater comparability across the regression coefficients, the data was scaled before training the model. However, since the data matrix was a 10,000 by 80,000 sparse matrix, mean-centering the data was infeasible on my laptop as the resulting array size would have been extremely large.


Top 20 Smallest Logistic Regression Coefficients on Scaled Count Data
hawaii
vineyard
que
bay
por
island
para
una
de
pandemia
californians
fernando
en
pandemic
sharks
valley
town
los
reshaping
islanders
Token
−0.25
−0.20
−0.15
−0.10
−0.05
0.00
Coefficient


Top 20 Greatest Logistic Regression Coefficients on Scaled Count Data
covid
fewest
ohioans
due
state
pearl
rural
ofallon
missouris
dakotans
login
airy
dhecs
eventsmeetings
searchable
kentuckians
cange
avg
httpsinteractivescourierjournalcomprojectscvmap
missourians
Token
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Coefficient

Model Training Times

	Naïve Bayes (Counts)	Naïve Bayes (TF-IDF)	Logistic Regression (Counts)	Logistic Regression (TF-IDF)	Lightweight Neural Network	LSTM
Time (ms)	37.2	36.6	8810	1590	>60000	>60000

Discussion

Interpretation of Results

The above results demonstrate similar performance on the test set across all models within the range (63%, 65%). However, the models differ markedly in their training set performance, likely indicating varying degrees of overfitting. The Naïve Bayes models demonstrate the lowest training accuracies of around 66%, while the remaining models demonstrate a training accuracy of about 70%, indicating a greater degree of overfitting given their similar test accuracies.

The models are perhaps best differentiated, performance-wise, by their training times. The table above depicts training times orders of magnitude different across each model class. The Naïve Bayes model trains the most quickly by far at about 37 ms. The logistic regression models each train in about 9 seconds and 1.5 seconds, respectively, while the neural networks each take over a minute to train through 10 epochs.

Given each model's similar performance on the test data, it would be sensible to designate the Naïve Bayes models as the "best" under this application. However, it is important to note that, given time constraints, very little hyperparameter tuning was applied to the logistic regression or neural network models. Given the latter type of model's dependence on numerous hyperparameters including hidden dimension size, number of layers, optimizer, loss function, etc., it is not necessarily fair to take their above performance to be the "best possible" under their respective architectures.

The coefficient magnitudes in the logistic regression model depict slightly disappointing results in the sense that the coefficient of highest magnitude were often "noisy" (e.g,. website url's) or state demonyms. These feature would essentially allow the models to "cheat" by predicting local voting results according to geographic location rather than bias present in the documents themselves. For instance, the model associates the tokens “missourians” and “dakotans” with a higher proportion of Trump voters, and the tokens “hawaii” and “californians” with a lower proportion of Trump voters.

Future Work, Areas of Potential Improvement

There are many areas potential improvement and future work. Foremost, each of the models could be cross validated under additional hyperparameters to achieve better performance. It could have also been possible to test entirely different classes of models, including decision forests or transformers. Given fewer computational restraints, the models could have also been applied to larger datasets which could thus reduce bias in the predictions. Further, the data processing steps could have been taken further to remove less-valuable content from the data. For instance, it was considered to apply stemming/lemmatization to each of the documents, however, this may have required excessive computational resources given time constraints. Finally, as demonstrated in the logistic regression coefficients, there were multiple instances of geographic entities/demonyms present in the documents. It may have been possible to apply a larger part-of-speech model to the data in order to more accurately remove such undesirable features from the data.

Conclusion

The results of this project demonstrate a moderate predictability of localities’ voting results according to excerpts of local news including references to the COVID-19 pandemic. It is worth noting that, given the points addressed in the above discussion, these results do not present concrete evidence of media bias, but rather suggest possible value in continued investigation of this topic.

References

[1] P. S. Hart, S. Chinn, and S. Soroka, "Politicization and polarization in COVID-19 news coverage," *Sci. Commun.,* vol. 42, no. 5, pp. 679-697, 2020/10 2020, doi: 10.1177/1075547020950735.
[2] B. Horne, M. Gruppi, K. Joseph, J. Green, J. Wihbey, and S. Adali. *NELA-Local*, Harvard Dataverse, doi: doi:10.7910/DVN/GFE66K.

[3] B. D. Horne, M. Gruppi, K. Joseph, J. Green, J. P. Wihbey, and S. Adalı, "NELA-Local: A Dataset of U.S. Local News Articles for the Study of County-Level News Ecosystems," *Proceedings of the International AAAI Conference on Web and Social Media,* vol. 16, no. 1, pp. 1275-1284, 05/31 2022, doi: 10.1609/icwsm.v16i1.19379.

[4] M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolu tional neural networks and incremental parsing.

[5] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011 2011.