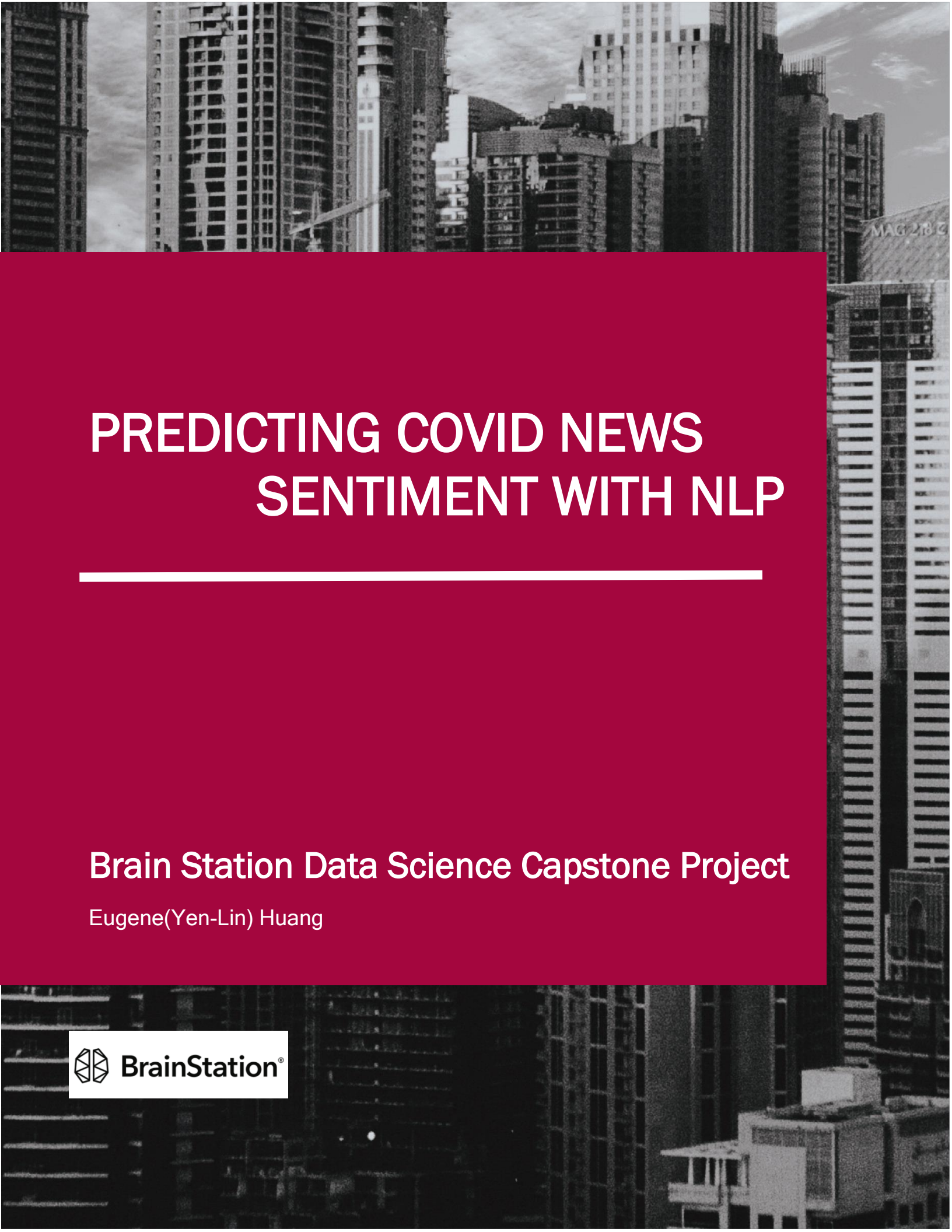

PREDICTING COVID NEWS SENTIMENT WITH NLP

Brain Station Data Science Capstone Project

Eugene(Yen-Lin) Huang

BrainStation®

Introduction

Text data has an increasing impact on today's world. From twitter tweet, news reports, to movie reviews, there is a vast amount of information incorporated in everyday texts. With the development of the Natural Language Processing (NLP) techniques, people can now extract information from texts more efficiently than ever before. In this project, I explore different ways to process the text data with NLP and predict the sentiment of Covid news articles using machine learning models. The main focus for this project is on applying different processing methods to the text data and see how well a model can perform just using the information from texts alone, although I do also run a quick grid search at the very end to see how much the model can be improved by adding other features such as authors and news sources.

The choice to focus on the text itself is two folds. First, the motivation for me to start this project really stems from the interest of extracting information from the text data. Imagine we include all available features out there into our model and find that the most predictive features of the article sentiment are certain authors or news sources, then I would not know how much my text processing had helped predicting the sentiment, which is what I am interested to know about. Second, by focusing on the texts first, we can first find the optimal embedding/vectorization to process the text, then move forward for further modeling.

Throughout this project, there are 4 different vectorization/embedding methods that I use to process the data, including the basic bag-of-words count-vectorizer, TF-IDF vectorization, a word2vec embedding trained by my own neural network, and a more refined LexVec embedding that was trained using all Wikipedia texts. For each vectorization method, I try to fit the model to three different machine learning models with hyperparameter optimization. The models I tried include logistic regression, decision tree, and KNN. In general, the best performing model across all vectorization methods is logistic regression, with which I have achieved about 65% prediction accuracy rate for my best model using the vectorized text alone. Also, perhaps surprisingly, TF-IDF vectorization outperforms both word2vec embeddings that I tried and is the best vectorization method among all alternatives in this project.

This project spans across three different Jupyter notebooks. The first one goes through the data acquisition and cleaning process as well as some preliminary EDA, the second shows my work on training my own word2vec embedding using neural network, and the third demonstrates my modeling process. This report provides a brief summary of my work.

1. Data Cleaning and Exploratory Data Analysis

I requested the original dataset from the AYLIEN website. The dataset contains 1.6 million news articles about Covid-19 around the world. The original file was in JSONL format, from which I have extracted the following features:

`Source`: The source media of the article

`Date`: The publication date of the article

`Headline`: The headline of the article

`Author`: The author of the article

`Main Location`: The location of the publishing media

`Language`: The language the article is written with

`Body`: The full text of the article

`Title`: Title of the article

`Language`: The language the article is written with

`Sentiment_Body`: The categorical sentiment of the article

`Sentiment_Body_Score`: The confidence level of the Sentiment_Body category

`Sentiment_Title`: The categorical sentiment of the article title

`Sentiment_Title_Score`: The confidence level of the Sentiment_Title category

My target variable is **Sentiment_Body,** which can be either positive, neutral, or negative. The goal of my modeling would be predicting this variable.

In my data cleaning process, I first dropped the articles with missing sentiment. I then cleaned the information in the **Author** column and imputed the missing **Main_Location** as much as I can using the **Source** information. If there are still missing values, I replaced then with “unknown”.

In my EDA, I wanted to have a sense of how news article sentiment varies across different authors, countries, or news sources, and whether there is a certain trend across time. I found that there is significant variation in sentiment across different authors and news sources. For example, if we focus on the authors with at least 50 articles and the percentage of their articles in the “positive” category, the top 10 authors would have above 90% of the articles being positive, and the same ratio for the bottom 10 authors would be less than 10% (see Figure 1).

2. Machine Learning Modeling

A huge amount of energy for my modeling was spent on experimenting the best embedding/vectorization method to process the text data. In particular, I have tried 4 different transformation methods as listed below:

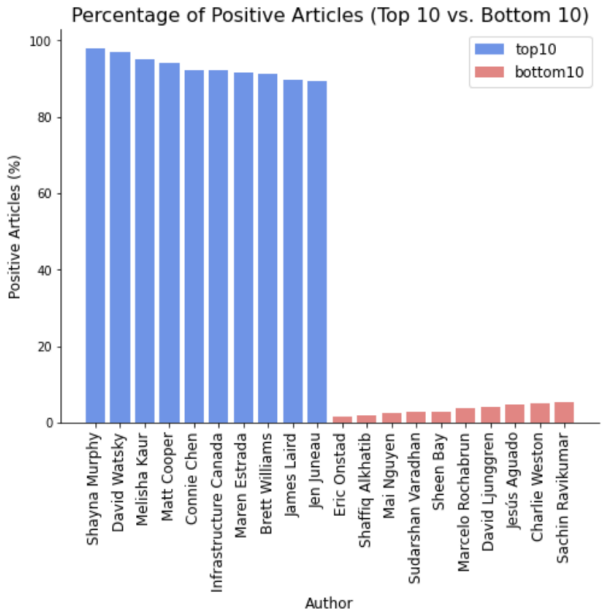

Figure 1

Bag-of-Words: Vectorize the article with simple word count

TF-IDF: Apply different weights to words in an article based on their appearance frequency in the article and across whole corpus.

Wor2Vec(self-trained): A word embedding trained by my own neural network

LexVec(pre-trained): A pre-trained word embedding found on the internet.

For each embedding, I tried to find the machine learning model that would give the best performance. The results are summarized below:

Transformation	Best Model(hyperparameter)	Testing Accuracy
Bag-of-Words	Logistic Regression(C=0.001)	64.75%
TF-IDF	Logistic Regression(C=1)	65.96%
Wor2Vec(self-trained)	Logistic Regression(C=10,000)	56.56%
LexVec(pre-trained)	Logistic Regression(C=10)	59.13%

As shown in the table, the best performing transformation is TF-IDF.

For my final model, I use the TF-TDF to transform the text column and add in a few other features like authors, news source, and news location. The performance is almost identical to the model just using the text column. The final model has a 65.66% testing accuracy. The confusion matrix is shown in the following figure:

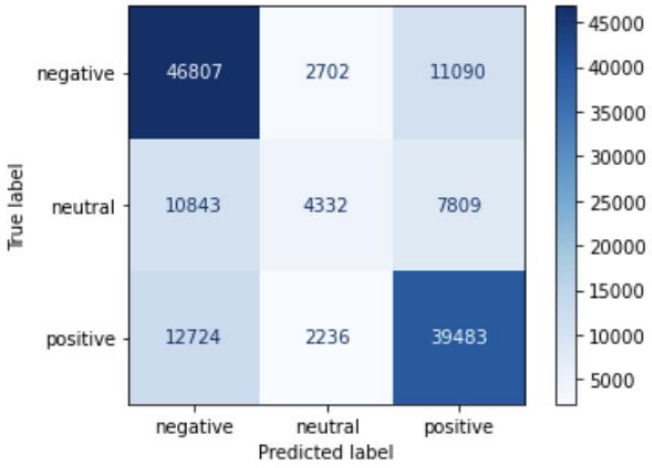

Figure 2

The model does a better job at capturing the positive and negative sentiment but fails to identify the neutral sentiment.

3. Conclusion

In this project, I have tried on different text encoding methods to process the news article texts and predict the article sentiment, including a Word2Vec embedding trained by my own neural network. It is encouraging to see that my text processing is carrying the load for my model prediction. Adding other features does not improve my model performance.

It is encouraging to see that my word processing is carrying the load for my model prediction. For NLP to be useful in the business world, one would like to see the processing on the text itself to produce useful information and help prediction. It would be helpful to know that we are able to predict the sentiment of a written text by the content of the text itself rather than relying on other information such as the authors or sources of the text.

For next step of this project, I would like to explore more advanced NLP models and see how they help my model perform. The Word2Vec and LexVec embedding I used here is more like a first step into the world of NLP. More advanced models are already developed, and I would like to learn more about their application and apply them to this project. Another thing I would like to do is the unsupervised topic modeling for the articles. I would like to explore more about how the machine classifies the unlabeled articles and whether I, as a human, can make sense of resulted clusters.