Fall 2019

Detecting and mapping real-time Influenza-like illness using Twitter stream data

Elisha D. Brunette
Eastern Washington University

Detecting and Mapping Real-Time Influenza-Like Illness using Twitter Stream Data

---

A Thesis

Presented To

Eastern Washington University

Cheney, Washington

In Partial Fulfillment of the Requirements

For the Degree

Master of Science in Computer Science

By

Elisha D. Brunette

Fall 2019

THESIS OF ELISHA BRUNETTE APPROVED BY

_____DATE_____

DR. DAN LI, GRADUATE STUDY COMMITTEE

_____DATE_____

DR. STUART STEINER, GRADUATE STUDY COMMITTEE

Abstract

Influenza has been identified by the World Health Organization as a global issue that could be more effectively served through an accelerated and widely-accessible public health surveillance tracking process. The ability to map and predict influenza outbreaks in a real-time heat map would be invaluable to health care systems to prepare for influenza outbreaks. In this study, the Twitter stream data is filtered to identify potential influenza-like illness (ILI) cases. Then the tracking of real-time influenza cases is further explored and analyzed through various machine-learning models. Among seven learning models developed to identify ILI tweets, the ELMo deep neural network model outperforms others regarding model accuracy and F-score. A heat map is generated to visualize real-time outbreaks of ILI in the U.S.A.

Acknowledgements

I would like to thank Dr. Li for giving me guidance throughout this project, answering my questions, and providing ideas for improving the strength of this project. I would also like to thank Dr. Stuart Steiner and Professor Rose Krause for reviewing my thesis, acting as my second and third committee members, and for the edits and responses they have provided.

Contents

List of Figures

Glossary

Area under the curve (AUC):

A definite integral is used between two points to identify the area under an ROC curve. In order to find the area under the curve use the function y = f(x) between x = a and x = b, and then integrate y = f(x) between the limits of a and b.

Backpropagation (BP):

The process that a neural network uses to update the weights corresponding to the input features during the training phase.

Bidirectional Encoder Representation from Transformers (BERT):

A deep neural network model that uses attention mechanisms to gather relevant context for a random word in a sentence, and then encode it into a vector which represents that word and is used in creating a classification for that word.

Corpus:

A large collection of text centered around a specific topic or idea.

Deep Neural Networks (DNN):

This is a variant of an artificial neural network with added layers that exist between the input and output, using a mathematical interpretation of the data in between the input and output. The DNN establishes if the relationship is linear or non-linear in nature.

Embedded from Language Models (ELMo):

This model is a long short-term memory, bidirectional model which creates real world contextual situations.

EXtreme Gradient Boosting (XGBoost):

This is a machine learning model which is a greedy algorithm utilizing an ensemble of decision trees, enhanced for speed and performance while performing data analysis.

F1 Score:

When using binary classification, this is a measure used to assessed the model's accuracy which considers both precision and recall to compute the model's score.

Forward feeding:

The process that a neural network uses to train. This process includes multiplying a weight

to the input features, using a net input function to sum those weights, and then using an activation function to produce an output classification.

GridSearchCV:

A function from the SKLearn library in Python [27]. This function exhaustively searches through parameters for models, and then tests each model using k-fold cross validation. It then returns the model which produces the highest result metric desired.

Influenza like illness (ILI):

A non-specific respiratory illness which is not the influenza, however mimics or has similar symptoms to the influenza virus (e.g. cough, fever, headache, nausea).

Kernel functions:

A function used to take non-linear data and transform the data into a higher or different dimensional space.

K-Fold Cross Validation:

A re-sampling procedure which evaluates machine learning models that are trained on a limited data sample.

Naïve Bayes:

A machine learning model based on Bayes Theorem. It is a naïve model, because it assumes the features are independent of each other.

Natural Learning Process (NLP):

This is a sub-field of multiple disciplines including, but not limited to data science. NLP addresses and clarifies the human and computer interaction, which is identified or termed the natural language. This is specifically to aide with the analysis and processing of extensive amounts of natural language data to provide clarity regarding speech recognition, natural language understanding, and natural language generation.

Random Forest:

This is a model using an ensemble method, consisting of multiple decision trees for data classification.

Receiver Operating Characteristics (ROC):

The ROC curve indicates sensitive/specific pairing corresponding to a particular decision threshold. In order to distinguish how well a parameter between two diagnostic groups is

measured, the area under the ROC curve is identified.

SKLearn Library:

A Python Library that implements various models, methods, and pre-processing techniques.

Support Vector Machine (SVM):

This model implements supervised learning using labeled training data for outputting a discriminative classifier which is defined by a separating hyperplane.

Trained Models:

Model algorithms that have completed the respective training process.

1 Introduction

Social media has become a popular way of interacting with the world. Societies all over the world are shifting to internet-based means of communication, and this has created an immense amount of information available for use in scientific research. "Over 2.5 quintillion bytes of data are created every single day, and it's only going to grow from there. By 2020, it's estimated that 1.7MB of data will be created every second for every person on earth" [13]. Based on a study conducted in 2018, an adult spent on average three hours per day interacting with social media [15]. This has been an exponential growth from 2014, where the average usage was only 32 minutes. Social media has become the number one way to interact with other humans, and has become a popular way to gather, analyze, and disseminate information.

The volume of people on social media platforms is enormous, and the amount of information created every minute from those members is even more staggering. Figure 1 shows the details of data generated minute-by-minute from various social media platforms [13]. Due to public availability of this information, the data produced by these sources is widely used for scientific research. Twitter has provided a free Application Program Interface (API) for developers, thus making it a popular hub for gathering information. There are 78% of Twitter users who utilize social media for gathering news in their fields of interest, and 83% of 193 United Nation member countries have an official Twitter account [3]. Twitter has become a popular social media site for sharing everything from global news and politics to personal health issues inclusive of signs and symptoms of the flu.

Social media has become so expansive it now has the potential to be a powerful resource for public health surveillance on a global level [7]. The spread of infectious diseases is a global issue that needs to be attended to. Scalability of detecting outbreaks using human interaction is not plausible, thus passing this responsibility off to machines is far more applicable. The World Health Organization states 3 to 5 million cases of severe illnesses are reported annually, and an estimated 290,000 to 650,000 deaths are due to respiratory illnesses worldwide [21]. The need for detecting the flu is a global problem, thus this thesis aims to find a national solution by using public health surveillance of epidemiological

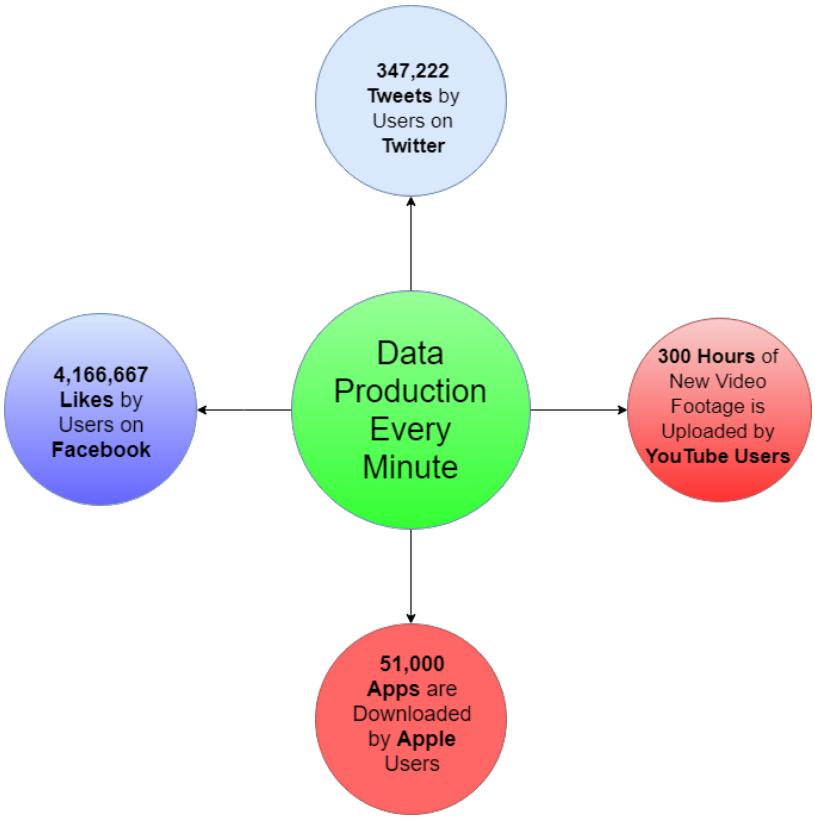

Figure 1: The amount of data produced every minute by social media platforms.

information tracking.

As indicated from pandemics such as the H1N1 virus in 2009, there is a growing need to improve the gathering and dissemination of live health care tracking for enabling appropriate and timely provision of services to the public [4]. Google Flu Trends (GFT) was one of the first and the most sophisticated examples of attempting to provide this service to the general public [16]. GFT provided the first known real-time flu outbreak tracker, which was accomplished through monitoring Google Searches available to the public. Google found a 0.90 correlation compared to the Center of Disease Control's (CDC) predictions, however GFT's predictions were coming out one to two weeks before the CDC had released outbreak information in 2008. Google ultimately had to discontinue GFT in 2015, due to multiple coincidence correlations in their data [19]. The rise of GFT is the first applicable example of how to use big data analysis for epidemiological health surveillance to track and predict the flu.

While Google Flu Trends was tracking the flu through searches, the H1N1 virus was

also being tracked through Twitter using machine learning models [26]. When the H1N1 pandemic happened in 2010, Twitter rose to the front for social disease tracking because of the contextual nature of the tweets. The Twitter API provides metadata associated with each user and tweet, which enables predictive models to find a connection between flu outbreaks and tweets received. The nature and accessibility of this real-time social media platform enables live tracking, which makes it invaluable as a global tool for visualizing the outbreaks of the flu.

Various techniques have been developed to predict the spread of influenza and other infectious diseases predominately using traditional supervised learning algorithms. One of the first research papers pertaining to this topic obtained a 0.78 correlation compared to the CDC reports [8]. This research paper was one of the first to use Twitter tweets and logistic regression to predict the spread of influenza. Others expanded on this work using other models such as Support Vector Machine and Neural Networks. A survey paper discussing the prediction of disease outbreaks found that the best results came from Support Vector Machine producing the highest Pearson Correlation [2]. Other models that presented high correlations are mechanistic disease models called Metpopulation, and a topic model called ATAM+ [23]. These models all received a Pearson correlation of 0.95 or above compared to the 2008 CDC data and will be discussed in more depth later in this thesis.

The aim of this thesis is to use machine learning models to classify real-time tweets and stream them into a live heatmap indicating the spread of infectious diseases across the United States. This study explores traditional machine learning models, as well as recently developed Natural Language Processing (NLP) models known as Embedded from Language Models (ELMo) and Bidirectional Encoder Representation from Transformers (BERT) to detect and predict influenza tweets. NLP is used to put human sentences into a form which machines can understand. Bag of words, n-grams, and feature extraction are common techniques used for preprocessing sentences prior to feeding the data into models. The BERT model has specific data masking that has to occur before the model can be trained. Further discussion on the process that BERT uses for training is explored in Section 4. Each model was fine tuned on different parameters, and used bag-of-words for data preprocessing for all models except the BERT model. Discussion around the Twitter API and creating a

heatmap from this data onto a map of the United States is also provided as a product of this thesis.

2 Literature Review

There is an extensive amount of existing work using efficient machine learning techniques to map influenza outbreaks using Twitter. Various methods have been implemented using a combination of data analytics, lexicon-based sentiment analysis, and machine learning processes to refine and interpret raw data. There have been ongoing various uses for these different processes since 2009 in regard to tracking and predicting epidemiological data on a national scope.

Data analytics is a way of interpreting data and has been applied to create heatmaps that track flu and cancer outbreaks in real-time using Twitter. Lee [20] used Twitter data to create a real-time influenza and cancer timeline, heatmap, and word analysis in 2013. This project produced a heatmap of the United States that counted the number of times the terms "flu" and "cancer" were mentioned in each state. This heatmap was scoped around tracking influenza, cancer, and cancer treatments. Pie charts were used to illustrate each category of cancer and provided a break down of specific cancer treatments. In addition, this project illustrated a timeline of tweets mentioning the flu. This particular project did not have any steps in place to remove the tweets that are not related to the flu, which is noteworthy because Lee's project did not include various machine learning techniques and metrics as an outcome.

Predicting the flu has traditionally had two different research methods for solutions. The first common strategy for finding a solution was lexicon-based preprocessing. Lexicon-based preprocessing has a variety of different approaches including bag-of-words, part of speech tagging, stemming, emoticon detection, and hashtag analysis [22]. These methods analyze the sentence in different ways. Part of speech tagging is an algorithm that categorizes words in a sentence into specific word function categories (e.g. is the word a verb, noun, or adjective). N-gram analysis is then utilized to further refine the incoming data and classify these word function categories either by hand or by an automatic algorithm into

word pairings (e.g. noun-verb-noun, noun-verb-adjective) [6]. N-gram analysis is one of the most common feature generation strategies, where n is the number of specific words to generate a feature [8], [12], [18]. An additional software option for feature generation is bag-of-words. This particular feature generation uses a count of specific words generated from the corpus data that the model is being trained on. Every time the specific word is mentioned in the test set, the count for that word is incremented by one. The bag-of-words method is one of the first feature generation methods implemented in Natural Language Processing (NLP). Bag-of-words is simple to implement and produces a representation of the text data for input into the machine learning models.

Machine learning is a common approach in text mining and historically has included sentiment analysis preprocessing for feature generation. The classification approach from machine learning is used in this thesis. Every machine learning problem using the classification approach starts out with a classified dataset. A classified dataset is a dataset that has the data classified into the different categories that are applicable for solving the problem. For the purposes of this thesis, those classifications answer if the the user actively has the influenza virus. Commonly, this classification is represented with 0 and 1, where 0 represents that the user does not have influenza and 1 represents that the user actively has influenza. This classified dataset is then split into training and testing sets with the split traditionally around 80% training data, and 20% testing data. A controversy around how much data should be set aside for the training set and the testing set is an ongoing discussion. This is determined based on the amount of data that is present in the classified dataset. The bigger the dataset, the higher the ratio of training data that can be used during the training phase. The test dataset needs to be held aside when training, because the model can then be evaluated using various metrics discussed in Section 5. Figure 2 shows the process in general, and what a classified dataset would look like.

Culotta [8] was one of the first to use Twitter tweets for making influenza predictions on a national scale. N-gram analysis and linear regression were used to predict the flu in 2010. Keywords and keyword groupings were formulated into features based on two different methods. One method utilized groupings that were hand chosen, and those that were gathered from several documents discussing the flu. The top 5000 most frequent words

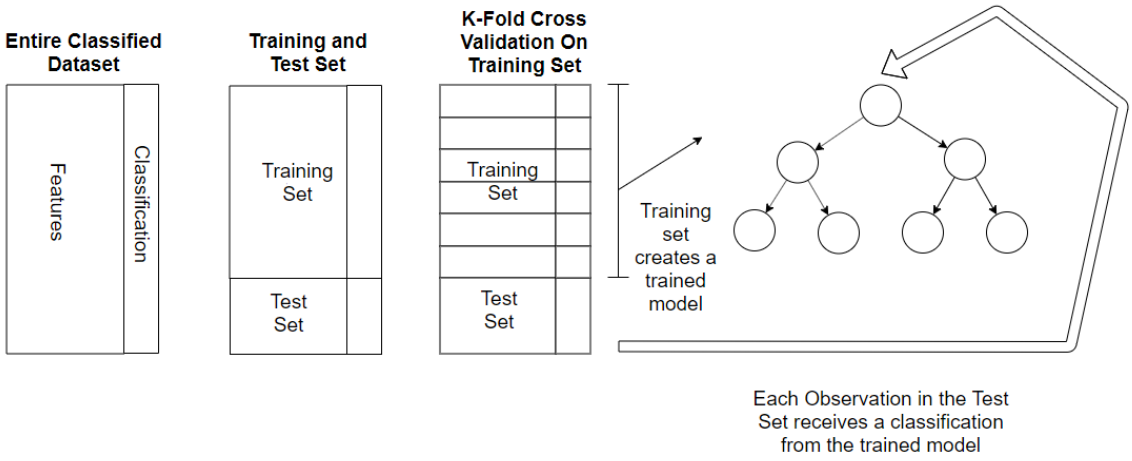

Figure 2: The classification framework for machine learning.

from these documents were used as features for the model input. This paper displayed that the manually selected words outperformed the document based keywords. The metric used to evaluate this article was the Pearson Coefficient metric by which compared the results with the report published by the Center for Disease Control (CDC). The hand-picked feature set received scores of 0.73 and the document-based keyword set received a Pearson Coefficient score of 0.58.

Culotta's [8] research in 2010 showed it was possible to use Twitter to map influenza, and inspired others to expand upon this research. Doan [12] recreated Culotta's work and expanded the research by using knowledge-based sentence splitting and semantic-based filtering. This research compared knowledge-based approaches and empirical approaches. The results from this study received correlation coefficients greater than 0.93. Google Flu Trends received a correlation coefficient, of 0.9912 the highest compared to archived logs from the CDC for the 2009 flu season. The results were attributed to Culotta's 2010 model which had the most census data in its training set. This paper revealed that a hybrid of models and sentiment analysis are more statistically significant compared to the traditional sentence-based analysis.

Support Vector Machine (SVM) is another machine learning model used to track epidemiological data for reporting infectious disease globally. SVM was implemented in 2009 to determine if a tweet was related to the H1N1 virus outbreak [26]. This epidemic showed

a rise in flu and influenza-like illnesses (ILI) in social media along with reports from the CDC for the 2009 flu season. In order to prepare the Twitter stream data for the model training process, the Twitter stream was filtered on common flu keywords. Retweets, links, and hashtags were also removed as a cleaning step before submitting the data for model training. The results from this method made predictions ahead of the CDC by two weeks, and showed a direct relationship to the H1N1 outbreak of 2009 [26]. This method was not reproduced and extended because the method required an unreasonably extensive amount of overhead and the models were trained on data that was only gathered for the 2009 year.

Machine learning models have been used on a more global scale from companies such as Medweb. Medweb is a medical research company that generated their own pseudo tweets related to ILI and proceeded to facilitate a machine learning study. The three different branches that collaborated on the research project were from China, Japan, and the US. These three groups from within the company submitted models for this study. This collection of pseudo tweets was generated from volunteers, instead of directly collecting and classifying their data from Twitter. The study had 80 volunteers generate 32 pseudo-tweets consisting of a maximum of 100 characters. This resulted in a total count of 2560 tweets. The data was translated into Chinese, Japanese, and English, and proceeded to implement machine learning models on the generated data. These tweets were classified into positive and negative for the flu, and each team tried different variations of sentiment analysis combined with different machine learning approaches [28]. The best model for English was ensembles of Heirchariclе Attention Network (HAN). Deep Character Level Convolutional Neural Network with loss functions performed the best based on F1 Score, with accuracy across all three languages. This data was not compared with the CDC and thus did not have a Peason Coefficent for comparison to other research.

A survey paper was released in 2018 that reviewed the topic of predicting ILI and the flu which summarized results from 22 different modeling approaches [2]. Models included in this review were neural networks, Ailment Topic Aspect Model (ATAM), graph data mining, and mechanic disease models. ATAM is a method that makes predictions based on illness-symptom pairings and the model was improved by another author to make ATAM+. Each paper introduced in the review had a different preprocessing method associated with

it. The SVM model had the most amount of papers cited in the review with four citations total, and the SVM model receiving the highest Pearson Correlation of 0.9897. Graph data mining results produced the lowest Pearson Coefficient value of 0.545. Graph models use sentences to generate graphs and predict if the graph is related to the classification of the tweets. Topic models are models that are specifically created and applied for this topic, such as ATAM and ATAM+. These models were not far behind the SVM model with Pearson scores between 0.95 to 0.98.

The SVM model that outperformed the rest of the models presented in the 2018 survey paper involved a significant preprocessing step using n-gram analysis and was conducted in 2013 [18]. The first step in preprocessing was to identify if the text from each tweet was related to the flu, then further classify those tweets into self/other and awareness/infection for feature generation of each tweet. Each category had n-gram analysis to create pairings based on two to three word groupings. Examples of these pairings include: noun-verb-object, subject-verb-object, first noun-verb, last noun-verb noun-flu. These n-gram pairings were analyzed and classified into related/unrelated, concerned awareness/infection, and self/other pairings. These features that were created from the tweets were then fed into the SVM model for results. This paper received a Pearson coefficient of 0.9879 compared to the CDC's archived 2009 data.

Cai [5] made another attempt at using n-gram analysis combined with three different models and showed a correlation to the CDC trends for 2016. Cai manually determined if a tweet was related to the flu or not. This process was done for the corpus of text that was collected from Twitter and totalled 1000 records. N-gram analysis was used to create the feature space of each text, and the models used for this process was Naive Bayes, Support Vector Classification, and logistic regression. The results achieved had a 0.96 correlation compared to the official CDC data from 2016.

Machine learning has continued to evolve past the traditional models mentioned above. Introducing transitive learning into neural networks and deep neural networks have become the new state of the art techniques for NLP. Transitive learning starts with a base model that is trained on a large amount of corpus data. This model takes months to become fully trained, and then this model can be further trained on domain-specific data when imple-

mented into a research project. The corpus data consists most commonly of Wikipedia and various encyclopedias. Domain-specific census data gives the model domain-specific terminology for training, and increases the overall accuracy of the model when applied to that field of study. Applied learning techniques have become popular with deep neural networks such as ELMo [25], and BERT [10], explained in Section 4.7. For the purposes of this thesis these modern models are utilized in combination with Decision Tree, Random Forest, Naive Bayes, XGBoost, and SVM to analyze the data gathered from Twitter. Data preprocessing was conducted using bag-of-words, and then the above named machine learning models were used to filter out the tweets that are not necessary in the final heatmap.

3 Data Collecting and Pre-Processing

3.1 Stream Introduction

The data was predominately collected using a software package called Tweepy. The Twitter API allowed this software to collect data from a tweet stream provided by Twitter. Tweepy allows for simplified interactions with the Twitter API by providing a wrapper for the core services. Interaction with the Twitter API was ongoing throughout the duration of this project. The preprocessing of Twitter stream data included the following steps:

1. Gather geo-coordinates to be plotted on a map;
2. Filter the stream for flu keywords;
3. Remove unused information from the stream;
4. Upload the flu stream data to a cloud database for offsite storage.

The first step was a requirement from the beginning of the project, and the other steps were added into the project once the amount of data gathered from the stream exceeded one terabyte. The stream produced 10 GB of data every day. After these preprocessing steps, the data intake was reduced to 100 KB every day. This made the stream more efficient and greatly reduced the data footprint of the project.

3.2 Tweepy API

The Tweepy API provides three options for filtering the information from Twitter.

1. The Twitter API allows filtering of the input stream based on specified keywords, locations, and languages in order to ensure the information being assimilated to the client contains at least one of the fields specified in the filter.

2. The client can filter the stream data without Twitter's back-end doing any of the preprocessing or filtering. This allows for increased access to data that will be less restricted in the filtering process.

3. The last is to combine the above two methods together to allow for greater control of information, processing, and increase the accuracy of gathering pertinent data for the project.

The first method comes with some complications, because it relies on the Twitter back-end to filter the data. Based on our experiments in this study, this method has proven to be inaccurate because 18% of the tweets are unrelated to the search terms, or the language filter does not account for bi-lingual users. The location filter is the most reliable filter because geolocation can be extracted from the tweets for 98% of the given cases. This filter has disadvantages as well, because filtering on geolocation removes language and keyword filters.

In the second method, the information is unstructured and random. The client is responsible for filtering and searching through the information provided by Twitter. This makes the information random and unorganized because it removes all of the filters for the stream data. Specific keywords and phrases are overwhelmed by the volume of information that the Twitter API provides. In general, this method is reliable for finding what is trending on Twitter.

The last method allows for the most control of the data by the programmer to filter from a smaller stream of data and gather only the information required from Twitter. This last method was implemented for this project because of the increased accuracy and efficiency of gathering data. The data available through Twitter is enormous and this method allows

for the most efficient filtering of the data. This study aimed to answer the question "Does the user actively have the flu?" To answer this specific question using Twitter, the optimal solution found is to filter the stream on keywords related to the problem and remove fields that are not necessary in the tweet. The details for data filtering are discussed in the following sections.

3.3 Data Collection

3.3.1 Phase 1

Data collection and processing included three phases with many iterations in each phase. In the first phase, there was no keyword filtering present on the stream. The information was stored to an external disk drive, and extra fields were removed using Java. Extended time was spent gathering one terabyte of data which was not filtered sufficiently for use in this study. Manual analysis for over one thousand tweets resulted in zero positive occurrences for tweets indicating the flu. Data collected during this phase indicated multiple irrelevant topics including the latest movies, some political issues, and trending retweets.

During the processing step of phase one, statistical information was gathered. Of the information collected and processed, 29.95% of the tweets had geolocation coordinates information. Deleted tweets made up a large portion of the data collected during this phase. These tweets were irrelevant in this study as they contained no pertinent information. The process of data collection and filtering increased the time spent in this phase due to the volume of irrelevant data collected. After this phase, a second step was created to reduce the amount of extraneous data during processing. This step required filtering the stream for flu keywords.

3.3.2 Phase 2

Due to the low occurrence of relevant tweets, stream filtering for flu keywords was initiated. A keyword filter was applied to the stream, and the created client software filtered out the useless fields within each tweet as new data was gathered. By filtering the stream, the amount of information being stored onto the external disk drive was reduced

by over 99%, improving practicality of storing the data. This filtering removed the tweets that did not have a geolocation associated with them. There are five different fields in a tweet that could determine if coordinates are present for geographical mapping.

3.3.3 Phase 3

In this phase, the incoming data collected from Twitter was filtered using the same steps as Phase 2 and then uploaded to a relational database hosted on Amazon's Web Services (AWS) for cloud storage. The benefit this phase provided to the project was three-fold. By storing the data on a cloud the information became more accessible, manageable, and easier to query. Once the tweets were located on AWS they were hand-classified. 9000 tweets were classified and prepared for the modeling phase.

The data collection process lasted over a year, and resulted in one and a half terabytes of information. In the early stages of the project, the stream did not have keyword filtering. Keywords were collected from the CDC during Phase 2 that represented influenza symptoms and were used to filter the stream [11]. However despite attempts to improve the relevance of data gathered, specific keywords from this list, such as 'headache' and 'nauseous' increased the frequency of irrelevant tweets gathered as these symptoms can be associated with tweets separate from personal flu reports. True flu tweet classifications were reduced from 10% to 2% when compared to filtering on 'flu' and 'influenza' to the list of keywords below. A full list of keywords used were 'fever', 'runny nose', 'stuffy nose', 'headaches', 'fatigue', 'vomiting', 'diarrhea', 'cough', 'sore throat', 'flu', and 'influenza'.

3.4 Stream Database Structure

The database structure was not a complex data structure, because the queries that were required for this project were simple select and update statements. The relationship of the data was independent, and the data did not need to be accessed independently. The structure of the database is listed in Table 1.

The raw stream data came with a plethora of information that was filtered down to the fields listed in Table 1. The coordinates associated with a tweet were available in five different fields within the metadata, with 92% of the tweets processed requiring the

AWS Hosted Cloud Database Schema	
Column	Type
Created_At	varchar(30)
tweet_id	bigint(64)unsigned
user_id	bigint(64)unsigned
text	longtext
location	varchar(100)
name_of_place	varchar(100)
language	varchar(2)
hashtag	mediumtext
tweet_coordinates	varchar(42)
flu	int(2)unsigned

Table 1: The Schema used for the AWS hosted Database.

centroid to be calculated for the geolocation of the tweet. The centroid of a bounding box is computed using the average longitude and average latitude. The average longitude and latitude became the point coordinate for the bounding box for each tweet. The bounding box point estimates were used for the heatmap portion of the project. The classification of each observation was stored in the 'flu' column for retrieval when starting the modeling phase. The classifications column had two possible values (0,1). The value '0' is associated with the user not actively having the flu, and '1' indicates that the user actively has the flu. These classifications are used to train the model to predict for each future observation.

3.5 Data Pre-processing

This section outlines the steps taken to process raw text data into a form that a machine learning model can interpret and analyze. Each model outlined in Section 4 follows the processes and parameters to fine-tune the model for each dataset. Each model had the same input data, and all input data was hand classified. In total there were 10,000 tweets that were hand classified and train-test sets were created from these tweets. When classifying each tweet, the main objective was to answer the question: "Does the user actively have the flu?" Due to the narrow scope of this question, the dataset resulted in only 2% of the total English tweets having positive classifications. Eighty percent of the data was used for training the models, which was accomplished utilizing SciKit Learns train_test_split method [24]. This concluded how the data was classified in preparation for using bag-of-words.

After the datasets were split into reproducible subsets, the bag-of-words method was used for processing the text data into a multiplicity of features for modeling. The bag-of-words method was used as a preprocessing step for all of the models implemented in this study, except ELMo and BERT. The bag-of-words method assigned a feature to each word, and if that word was present in the tweet, then the feature value associated with that word was incremented by one. This created a sparse dataset for modeling, because most observations did not have many words. Despite the limited feature set produced, the data was successfully utilized for modeling after this step was complete.

ELMo and BERT required a different data preparation process through each of their individual libraries. Bag-of-words was not used, as it is not a compatible text processing step for these algorithms. There are multiple functions available for processing data through ELMo, and the ones chosen for this thesis are embedding and compression functions available through the Keras Tensorflow library. These functions work similarly to the bag-of-words method, but compress the dataset to create a dataset that is not as sparse as the bag-of-words method. ELMo requires the sentence to be split up, instead of created into features for predicting the next word in the sentence. Data input for BERT requires that one of the words is masked by a function in the BERT library. Then the model predicts that word using the other words in the sentence. Both models are outlined in greater detail in Section 4.7.

After training was complete, the models were used to make predictions for the test set. Prediction result metrics were recorded and described in Section 5. The first dataset used the training and test data from Cai's [5] research, but the tweets had to be reclassified to answer the specific question mentioned above. A breakdown of the classification distributions are as follows for each dataset:

- Dataset 1: 1000 tweets from Cai's research [5] were used for this dataset, and 16% of observations were positive classifications.
- Dataset 2: 2000 tweets were gathered through the Twitter stream. 2% of observations were positive classifications.
- Dataset 3: 5000 tweets were gathered through the Twitter stream. 2% of observations

were positive classifications.

- Dataset 4: 2000 tweets were gathered through the Twitter stream. 50% of observations were positive classifications. This method used over sampling to generate 1000 positively classified observations.

- Dataset 5: 167 tweets were gathered by taking all of the positively classified tweets from Dataset 3, and under sampling was performed on the negative classifications from Dataset 3 to create a sum total of 334 tweets for this dataset. This dataset contained 50% positive classifications.

4 Predictive Models

There is a plethora of models available for use in deciphering data-related questions to identify specific classifications from acquired features. The models implemented and optimized for the purposes of this project include Decision Tree, Random Forest, Support Vector Machine, Naïve Bayes, XGBoost, Embedded from Language Models (ELMo), and Bi-Directional Encoder Representations from Transforms (BERT). At the time of writing this paper, ELMo and BERT are the most recent models developed for Natural Language Processing. BERT was first introduced in "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" published in October 2018 [10], and ELMo was introduced in "Deep contextualized word representations" published in March 2018 [25]. These models use deep neural networks which will be discussed later in Section 4.6. This study aims to explore the selected models in order to classify real-time tweets for influenza cases.

K-Fold cross-validation was also implemented in this project to evaluate the performance of the models. Cross-validation is the process of training the data by splitting the data into k equal subsets, and holding one of those subsets aside from the validation phase as shown in Figure 3. The model is trained on the $k-1$ subsets, and tested on the validation set. This process then iterates through all k subsets allowing for each subset to be the validation set and averages the results from each of the k iterations.

K-Fold cross validation helps prevent the model from overfitting the data. Overfitting is

Figure 3: An example of how cross validation splits up the dataset, and iterates through every k subset.

the process where the model becomes extremely proficient at identifying correct classifications for specific feature observations in the training set. The model does not perform well on the test set, because the model excels at classifying the observations in the training set only. Cross validation significantly reduces this possibility, because it allows for the optimal use of the entire dataset by allowing for every section to be the test set.

4.1 Decision Tree

Decision tree is a classification model that splits the dataset on different feature values at each node. When traversing a decision tree, each observation starts at the root node, and traverses through the tree to a leaf node. Each leaf node in the tree determines the classification of the observation. Each node in the tree is connected to two or more subtrees depending on the implementation. When training the dataset, Gini and Entropy splitting functions are used to determine which features split the dataset into smaller datasets. These calculations are performed for each feature.

The decision tree in Figure 4 looks at all of the features from the dataset, and uses either

the Gini or Entropy calculations to determine the splitting feature. The Gini equation is:

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

where

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

and

$$Gini(D) = 1 - \sum_{i-1}^{C} (p_i)^2$$

where $|D|$ is the number of observations in the dataset, $|D_1|$ is the count for the first classification for the observation, $|D_2|$ is the count for the second classification for the observations, C is the number of classes that the observation can be categorized into, and p_i is the probability of that class in the observation set. The Entropy equation is:

$$Entropy = \sum_{i=1}^{C} -p_i * log_2(p_i)$$

where C and p_i are the same as in the previous equation. Determining which feature to split on each layer the entropy of each feature is determined by:

$$Entropy_A(D) = \Sigma_{j=1}^{v} \frac{|D_j|}{|D|} Entropy(D_j)$$

where $\frac{|D_j|}{|D|}$ is the weight of the jth partition. These equations are combined to find the total gain from Entropy:

$$Gain(A) = Entropy(D) - Entropy_A(D)$$

where $Gain(A)$ determines how much would be gained by branching on feature A. The feature is determined by the lowest value of $Gain(A)$

The decision tree model has a tendency to overfit because of how specific the tree can be trained. This problem is generally solved by the Random Forest model.

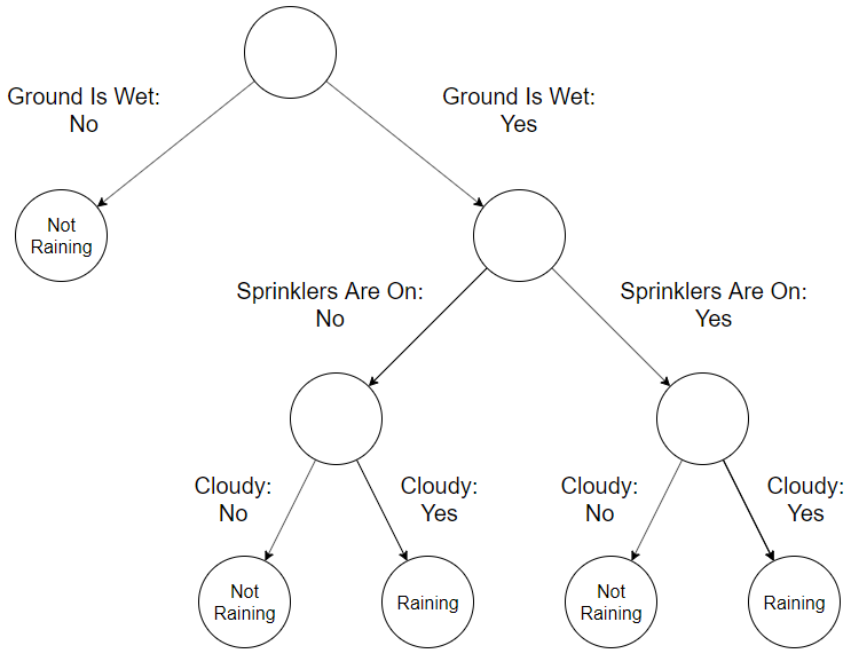

Figure 4: An example decision tree.

4.2 Random Forest

Random Forest is considered to be an ensemble method, which is created using a specified number of Decision Trees. These Decision Trees are tallied to create a group decision for each observation, instead of relying on just one tree for the final decision. Each tree in the forest provides a prediction for the test observation, consequentially the grouping of results is combined and counted to determine the best overall result. Each tree in the random forest trains on a sub-group of the available features. The model is strengthened by each tree being trained on a different subset of features and by the number of trees in the forest, because the model doesn't depend one specific feature for classification. These factors allow for less overfitting in the overall model.

4.3 Naïve Bayes

Naïve Bayes is a statistical learning model. This model is based on Bayes theorem [17] and the equation that this model is based on is:

$$P(C_i|x_1, ..., x_n) = \frac{P(C_i)P(x_1, ..., x_n|C_i)}{P(x_1, ..., x_n)}$$

C_i is the classification values, and x_i is a feature from the dataset. This model is naïve because it assumes that each feature is independent of each other. Because of this assumption, the equation above can be simplified to:

$$P(x_1, ..., x_n|C_i) = P(x_i|C_i) \cdot P(x_2|C_i) \cdot ... \cdot P(x_n|C_i) = \Pi_{k_i}^{n} P(x_k|C_i)$$

Naïve Bayes is implemented in this study to provide statistical perspective of the data for comparison of outcomes.

The first type of Naïve Bayes that is implemented in this project is the Complement Naïve Bayes and uses the complement calculation to compute the model weights [24]:

$$\theta_c i = \frac{\alpha_i + \sum_{j_i y_j \neq c} d_{ij}}{\alpha + \sum_{j_i y_j \neq c} \sum_k d_{kj}}$$

$$w_{ci} = log\theta_{ci}$$

$$c_{ci} = \frac{W_{ci}}{\sum_j |w_{cj}|}$$

The last model for Naïve Bayes that is implemented assumes a Gaussian distribution to the data. The Guassian distribution computes the probablility using the equation:

$$P(x_i|y_i) = \frac{1}{\sqrt{2\pi\sigma_y^2}\ \exp(-\frac{x_i - \mu_y}{2\sigma_y^2})}$$

4.4 Support Vector Machine

Support Vector Machine is a model used for classifying both linear and nonlinear data. This algorithm creates a higher dimensional dataset by using nonlinear transformation.

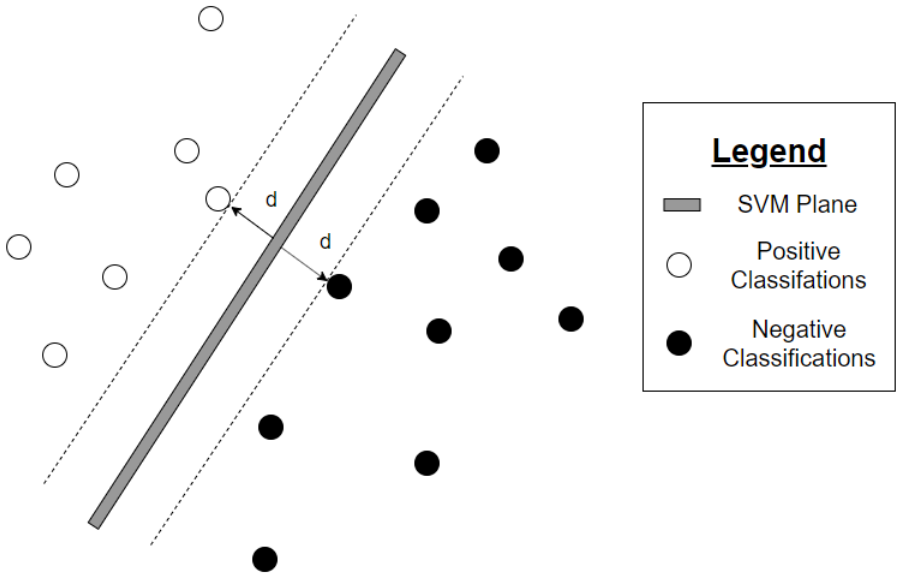

Figure 5: The distance that is being optimized for SVM.

The higher dimensional dataset is split by a hyperplane into the various classifications. The SVM minimizes the distance from support vectors, shown as margins in Figure 5, to the hyperplane.

Kernel functions are used to transform data into higher dimensions. These generated dimensions are used to create the most optimal plane through the data. The kernel is determined by the shape of the data. An example is shown in Figure 5. SVM is used frequently in multi-dimensional space and for binary classifications. The program looks for different planes through the data, depending on the kernel used, to improve the accuracy of the model.

4.5 XGBoost

XGBoost is a greedy algorithm using an ensemble of decision trees. This model does have some similarity to random forest as it utilizes an ensemble method, but it does not traverse all of the available features when generating each individual tree in the forest. A greedy algorithm finds the highest feature value to split the node on, and it can miss the global maximum numeric value depending on the value of each of the sub-tree's root nodes. Figure 7 shows an example from a decision tree where the algorithm is attempting

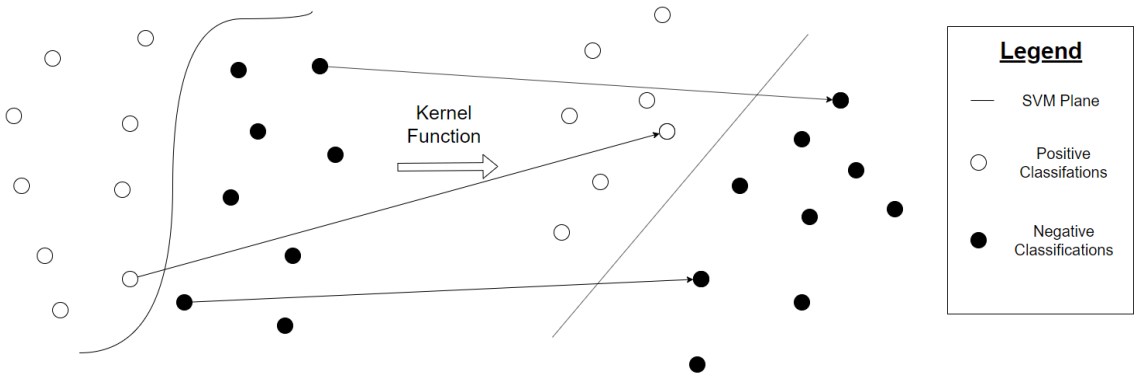

Figure 6: A picture depicting how the kernels change the input space to a feature space.

to find the greatest sum of each node through the decision tree. An optional algorithm traverses all of the root nodes, and uses the root nodes containing the highest numeric sum for determining its most optimal path. The greedy algorithm will traverse the root nodes and only sum the highest numeric value left and right at each split for determining its most optimal path. As visually indicated in Figure 7, the numeric value 6 is greater than 5, so the greedy algorithm will not traverse the root nodes of the left sub-tree past the numeric value of 5 and will miss the numeric value of 99. This algorithm is quick to train and easily scaled to large data sets. This algorithm tries to optimize the function below:

$$\mathcal{L}_{split} = \frac{1}{2}[\frac{(\Sigma_{i\in I_L}\ g_i)^2}{\Sigma_{i\in I_L}\ h_i + \lambda} + \frac{(\Sigma_{i\in I_R}\ g_i)^2}{\Sigma_{i\in I_R}\ h_i + \lambda} - \frac{(\Sigma_{i\in I}\ g_i)^2}{\Sigma_{i\in I}\ h_i + \lambda}] - \lambda$$

where

$$g_i = \partial_{\hat{y}(t-1)} l(y_i, \hat{y}^{(t-1)})$$

$$h_i = \partial_{\hat{y(t-1)}^2} l(y_i, \hat{y}^{(t-1)})$$

I_L is the left subtree instance, I_R is the right subtree instance, I is the current node instance, λ is a constant, $\hat{y}_i^(t)$ is the i-th prediction of the t-th iteration, and y_i is a classification of the the observation. Each node in the tree has a weight to be optimized by the equation:

$$\omega_j^* - \frac{\Sigma_{i\in I_j} g_i}{\Sigma_{i\in I_j} h_i + \lambda}$$

By using these equations at each split in the tree, the algorithm is able to determine

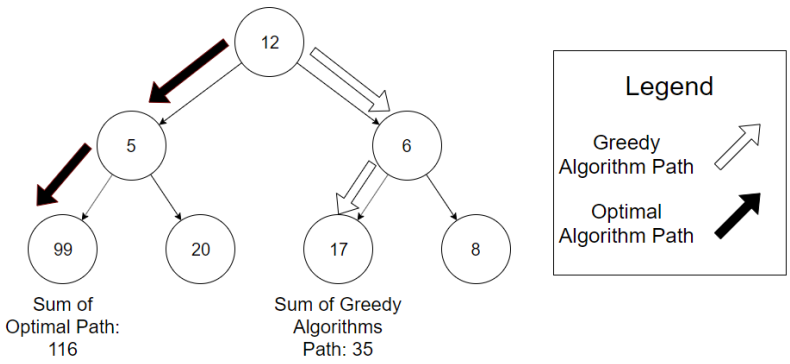

Figure 7: An example of a greedy algorithm.

which direction to traverse the nodes which reduces the amount of time spent traversing the tree, and also reduces the amount of hardware space required for storing information. As stated in [14], XGBoost is a powerful algorithm that is fast and is currently outperforming other algorithms in recent Kaggle competitions. The evidence supporting the efficiency and consistent performance of this model was a positive factor in determining whether to use it for this study.

4.6 Neural Networks

ELMo and BERT are the last two models to discuss. Both of these models use deep neural networks, which are an expansion of neural networks. ELMo and BERT have different data preprocessing steps. The preprocessing of ELMo is the bag-of-words method, and BERT has a specific word masking setup described in Section 4.7. After the data preprocessing the data becomes an input to a deep neural network. Neural networks have a hidden layer that consists of a collection of nodes as seen in Figure 8. Each node in the hidden layer takes all of the raw data, assigns a weight to the data input, applies the transformation function, and these results are combined into the final result. The results are compared with the actual classification label, and then backpropagation is used to adjust the weights of the data input. This process continues until the neural network is able to meet a certain threshold for loss. Figure 9 shows an example of how a neural network's forward feeding and backpropagation works.

This process is iterative and determines which input features are the most important to

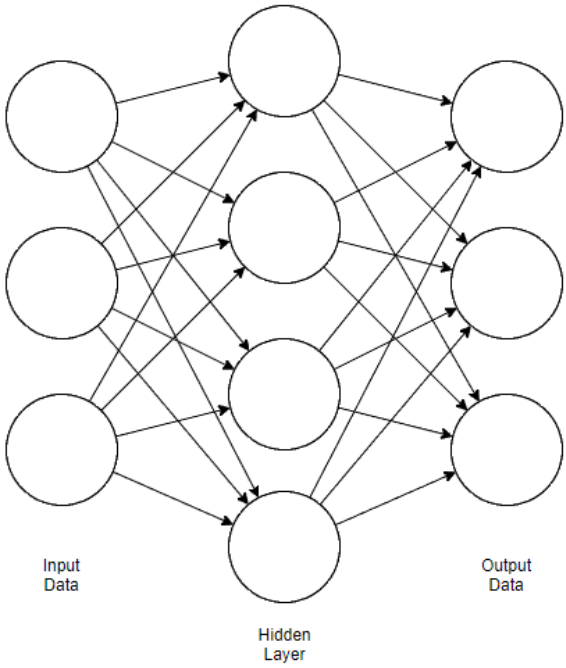

Figure 8: A conceptual overview of a neural network.

receive the highest desired performance metric. The metric optimized for determining how well the model performs is called loss and is calculated after forward feeding. The change in weight determines the value of loss, and thus a lower value for loss is desired when training a neural network.

Deep neural networks are an extension of neural networks with multiple hidden layers that transform the input data before the output layer. Each hidden layer does the same process as described above. This method is typically used for two step modeling. The first group of hidden layers is responsible for clustering the data and the second group of hidden layers is used to determine the classification of the output from the first group of hidden layers. This type of learning model is commonly used for image recognition.

4.7 ELMo and BERT

The models described in this section attempt to map incoming tweets to a classification to assimilate and organize data from acquisition. This section outlines the algorithms behind those models and these algorithms. ELMo and BERT have a lot of similarities regarding pre-training and training for the model. The pre-training for both of these models is done with

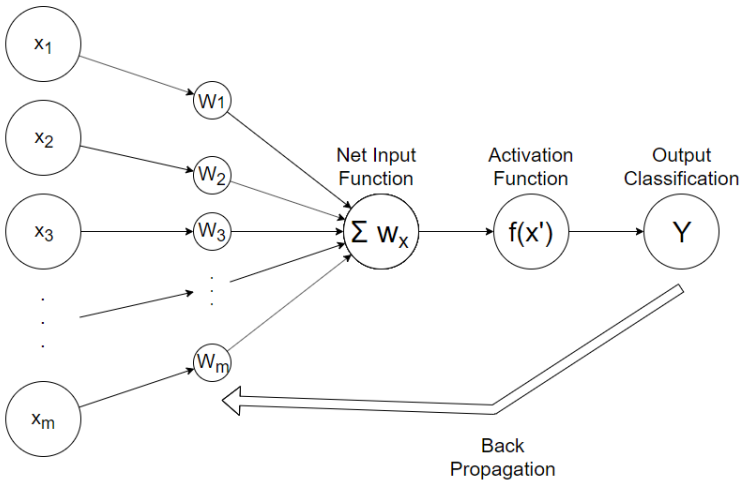

Figure 9: A conceptual view of forward feeding and backpropagation in a neural network.

huge corpus data from Wikipedia and other large collections of textbooks or encyclopedias that can be collected from the internet. The models are then trained again using domain-specific specific training data. By training the model in this manner, it allows the model to understand how a sentence is structured, and how it is used in further model training. Both models encode the sentence as an input for the model. BERT's training phase is dependant on the word encoding step, because it is trained to predict the word that is encoded.

ELMo reads through the sentence both forwards and backwards, then predicts the next word in the sentence for each direction. This makes the model diverse and helps it learn the new terminology of the specific field. This also helps the model learn context for different circumstances. BERT's training phase puts a mask on one random word in the sentence and tries to guess that word by analyzing the remaining words in the sentence. Figure 10 shows how BERT is trained to understand the architecture of sentences. The middle two layers of nodes represent the hidden layers of the deep neural network. The bottom row represents the input sentence split into $x_1, x_2, ...x_m$ where x_m represents one word in the input sentence. The top layer represents the output from the neural network and $y_1, y_2, ..., y_m$ represents the classifications the model predicts. BERT trains in a similar method as ELMo because it analyzes the sentence forwards and backwards. ELMo's training is very linear, and does not allow for knowledge of all of the words in the sentence while it is iterating through the

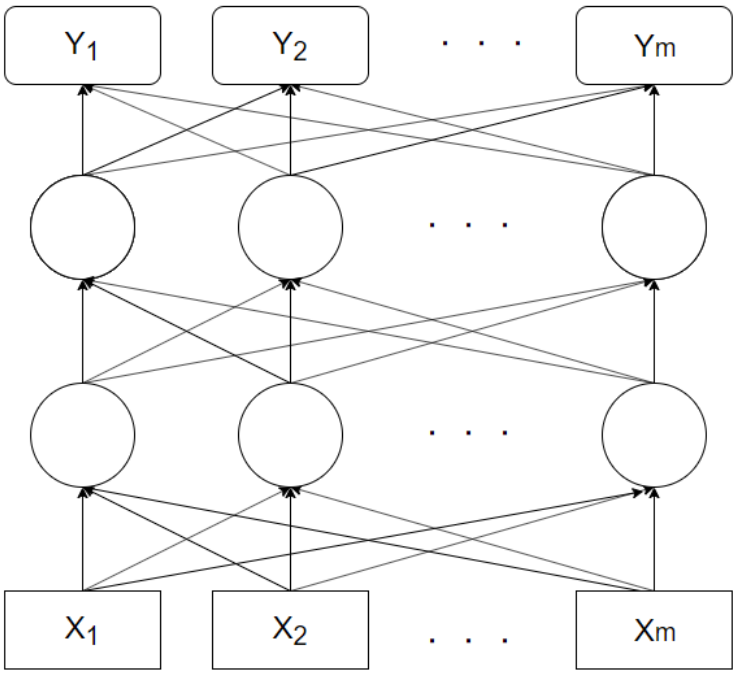

Figure 10: A simple BERT model.

sentence [1], [10]. Figure 11 illustrates ELMo's architecture.

4.8 Parameters

After each dataset was created, each model's parameters were fine-tuned using GridSearchCV through the SKLearn library [27]. Decision Tree, Random Forrest, Support Vector Machine, Naïve Bayes, and XGBoost were implemented through the SKLearn library. Both the ELMo [25] and BERT [10] models use neural networks, and the quantity of epochs was the only parameter that was fine-tuned for these models. The parameters that were adjusted are listed below for each model in the following structure:

- Decision Tree:
  - Criterion (gini, entropy): The function to measure the equation used for splitting the decision tree.
  - Max Depth (1, 2, 3, 4, ..., 24, 25): The maximum number of nodes that are allowed in the depth of the decision tree.

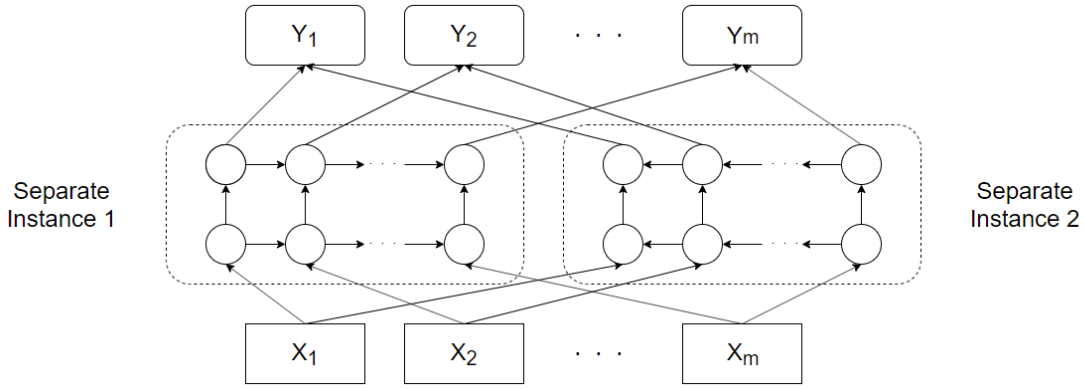

Figure 11: A simple ELMo model.

  - Min Sample Split (2, 3): The number of samples required to split an internal node.
  - Class Weight (None, balanced): The weight for each feature that is an input for the model when training.

- Random Forest:
  - N Estimators (5, 10, 15, 20, . . . , 85, 90, 95, 100): The number of decision trees in the forest.
  - Criterion (gini, entropy): The function to measure the equation used for splitting the decision trees.
  - Max Depth (1, 2, 3, 4, . . . , 24, 25): The maximum number of nodes that are allowed in the depth of the decision tree.
  - Min Sample Split (2, 3): The number of samples required to split an internal node.
  - Class Weight (None, balanced): The weight for the classes that are being split.

- Support Vector Machine:
  - Kernel (linear, polynomial, rbf, sigmoid): The function that is used to transform the data in vector space.

  - Degree (2, 3, 4, ..., 7, 8, 9): The maximum degree of the polynomial for the polynomial kernel.
  - Gamma (1, 2, 3, 4, ..., 12, 13, 14): A kernel coefficient for 'rbf,' 'poly,' and 'sigmoid' kernels.
  - Coef0 (1, 2, 3, 4, ..., 12, 13, 14): This parameter is only applicable to the polynomial kernel function and represents a constant in the poly and sigmoid kernel functions.
  - Shrinking (True, False): Whether or not to use the shrinking heuristic when training the model to fit the entire model into RAM.
  - Random State (4): A random seed that can be set for reproducability, otherwise this is generated randomly by default.

- Complement Naïve Bayes
  - All of the default parameters were used in this model.

- Gaussian Naïve Bayes
  - All of the default parameters were used in this model.

- XGBoost:
  - Silent (True, False): Whether or not messages are printed during training of the model.
  - Subsample (.5, .6, .7, .8, .9, 1.0): The ratio of the training features for each tree in the boosted model.
  - Max Depth (2, 3, 4, ..., 7, 8, 9): The maximum depth for base learners in each tree.

- ELMo:
  - Epoch (1-10): The number of times that the model can iterate through the dataset.

- Bert:
  - Batch Size (32): The group of data that is being processed at one time by BERT.
  - Learning Rate (2e-5): The rate at which the model trains.
  - Epochs (1, 2, 3, ... , 8, 9, 10):
  - Warmup Proportion (.1): The ratio of time that the model spends loading the data into memory before training.
  - Save Checkpoint steps (500): The number of observations that are trained before the model creates a saved checkpoint.
  - Save Summary Steps (100): The number of observations that are trained before the model saves a summary.

This concludes the steps taken in this study for the preprocessing and parameters used for modeling. The parameters were adjusted as necessary to provide optimal data outcomes during the process of modeling. The modeling phase of this study was notably limited by time constraints and resources available.

5 Experimental Results and Discussion

This section explains the metrics used and results from the datasets. There are five metrics that were used for evaluation of the models. The evaluation metrics are F1 Scores, Accuracy, Area Under the Curve (AUC) of a Receiver Operating Characteristic (ROC) Graph, Precision, and Recall. The possible values for these metrics are between 0 and 1. Discussion regarding these results is included for each metric in Section 5.3.

5.1 Performance Metric

The evaluation metrics included for this study indicate different categories of performance for the models. Each of the evaluation metrics is comprised of values from a confusion matrix. True positive, true negative, false positive, and false negative values make up a confusion matrix. Explanations for each value are:

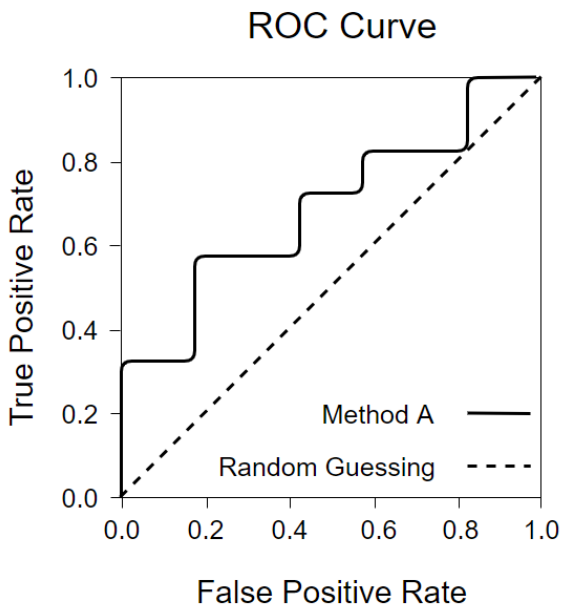

Figure 12: An example of a ROC Curve.

1. True Positives: the number of times the model predicted true and the actual value was true.

2. True Negatives: the number of times the model predicted false and the actual value was false.

3. False Positives: the number of times the model predicted true and the actual values was false.

4. False Negatives: the number of times the model predicted false and the actual value was true.

A ROC Graph is used to plot the False Positive rate (x-axis) vs the True Positive rate (y-axis). The dashed line in the figure represents random guessing. The solid line represents how the model performed. The result is a curve similar to what is illustrated in Figure 12. If the models line is above random guessing, then the model has outperformed a 50/50 prediction rate for each observation in the test set.

Accuracy is another metric observed for this study, and is defined as a percentage that represents the frequency with which the model correctly predicted true positives and true

negatives. The equation for accuracy is:

$$\text{accuracy} = \frac{\text{true positive} + \text{true negative}}{\text{true positive} + \text{true negative} + \text{false positive} + \text{false negative}}$$

This determines how many times the model achieved the correct classification. Problems arise from using this metric when the data is imbalanced. An imbalanced dataset has the minority of observations that are positive classifications. Since this metric can still obtain a high score by labeling everything as negative, this metric is not commonly used for imbalanced datasets. This is particularly prevalent in the datasets created for this project. In cases where the dataset is balanced, 83.5% would be a high value. This known fallacy for using accuracy in an imbalanced set makes this metric unreliable for the first three datasets in this study. However, this metric becomes more valuable when looking at Datasets 4 and 5, because the set is balanced with a 50/50 distribution. Overall, the accuracy metric will not hold as much weight in analysis for the datasets in this thesis.

Recall and precision metrics hold more value than accuracy for the distribution of data in this project. The equations to calculate precision and recall are:

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

Recall determines the total number of results classified as positive by the algorithm out of the total number of actual positive classifications in the dataset. This value determines how often the model generates true positives. Precision and recall are constantly in conflict with each other. This also means that insight can be gleaned from looking at the results from these metrics side-by-side.

If the value for precision is high and recall is low, this indicates that the cases selected which are positive have a high probability of being positive. However, not all existing positive cases will be appropriately identified as positive, so this method does not necessarily provide the most optimal coverage if thorough identification of positive cases is warranted.

If the model scores low in precision but high in recall, the model will inaccurately result in a lot of positive cases. This is the idea taken most often with cancer treatment, because the goal of chemotherapy is to eradicate all cancer cells from the patient doing chemotherapy treatment. Therefore, the radiation is not selective in which cells it removes.

High precision and high recall occurs when the model is selective about what it classifies as correct, and provides broader, more accurate coverage. This is the most desired outcome, however it is also the most difficult to achieve.

Low precision and low recall is the opposite of high precision and high recall. The model does not classify the observations correctly and does not give the best coverage to positive classifications. This situation also does not happen often, and it is the least desired outcome of metric results.

The next metric that is being used to assess model performance in this study is the F1 Score. This metric is a combination of precision and recall. The calculation for the F1 Score is:

$$\text{F1 Score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

5.2 Results

This section discusses the results from this study. The results are depicted in Figures 13 - 17. Each highlighted cell corresponds to the model's highest scoring result for that dataset. The bold values correspond to the highest value in the row, where the row represents the model. Each column represents the dataset. This thesis uses k-fold cross validation for each of the datasets during training, with the value of $k = 10$. In regards to ELMo and BERT, all of the results from each epoch are discussed, with the exception of BERT's epochs 5-10. These epochs were excluded due to redundancy, as these metrics had the exact same results for each dataset tested. Only the results which scored the highest during testing are discussed from Decision Tree, Naïve Bayes, XGBoost, and SVM due to the extensive amount of iterations completed during this study.

The metric results for AUC are displayed in Figure 13 for each dataset and model. The most noteworthy evaluation metric scores were acquired in Dataset 1 using BERT's epochs 1, 3 and 4 which all received a score of 0.836; the Decision Tree algorithm secondarily

AUC Results	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Decision Tree	0.833	0.617	0.826	**0.940**	0.924
Naïve Bayes (Complement)	0.650	0.627	0.503	**0.886**	0.834
Naïve Bayes (Gaussian)	0.653	0.707	0.560	**0.946**	0.716
SVM	0.689	0.634	0.595	**0.985**	0.834
Random Forrest	0.782	0.755	0.782	**0.926**	0.894
XGBoost	0.739	0.533	0.596	**0.941**	0.895
ELMo (Epoch 1)	0.708	0.500	0.708	**0.909**	0.820
ELMo (Epoch 2)	0.632	0.500	0.632	**0.954**	0.804
ELMo (Epoch 3)	0.676	0.545	0.676	**0.955**	0.834
ELMo (Epoch 4)	0.714	0.545	0.714	**0.977**	0.849
ELMo (Epoch 5)	0.685	0.762	0.685	**0.979**	0.804
ELMo (Epoch 6)	0.825	0.588	0.825	**0.974**	0.895
ELMo (Epoch 7)	0.714	0.807	0.714	**0.964**	0.895
ELMo (Epoch 8)	0.698	0.634	0.698	**0.969**	0.896
ELMo (Epoch 9)	0.743	0.588	0.743	**0.990**	0.880
ELMo (Epoch 10)	0.708	0.630	0.708	**0.966**	0.865
BERT (Epoch 1)	0.836	0.722	0.680	**0.977**	0.761
BERT (Epoch 2)	0.836	0.722	0.694	**0.977**	0.955
BERT (Epoch 3)	0.778	0.722	0.680	**0.977**	0.955
BERT (Epoch 4)	0.836	0.722	0.680	**0.977**	0.955

Figure 13: The combined results for AUC.

F1 Score	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Decision Tree	0.747	0.589	0.688	**0.942**	0.925
Naïve Bayes (Complement)	0.694	0.633	0.503	**0.888**	0.832
Naïve Bayes (Gaussian)	0.646	0.642	0.546	**0.947**	0.716
SVM	0.709	0.681	0.628	**0.985**	0.833
Random Forrest	0.729	0.681	0.729	**0.927**	0.894
XGBoost	0.758	0.533	0.631	**0.942**	0.895
ELMo (Epoch 1)	0.730	0.493	0.730	**0.910**	0.819
ELMo (Epoch 2)	0.668	0.493	0.668	**0.955**	0.803
ELMo (Epoch 3)	0.716	0.577	0.716	**0.955**	0.833
ELMo (Epoch 4)	0.743	0.577	0.743	**0.977**	0.849
ELMo (Epoch 5)	0.719	0.732	0.719	**0.980**	0.802
ELMo (Epoch 6)	0.771	0.626	0.771	**0.975**	0.895
ELMo (Epoch 7)	0.743	0.751	0.743	**0.965**	0.895
ELMo (Epoch 8)	0.728	0.681	0.728	**0.970**	0.896
ELMo (Epoch 9)	0.746	0.626	0.746	**0.990**	0.880
ELMo (Epoch 10)	0.730	0.650	0.730	**0.967**	0.865
BERT (Epoch 1)	0.730	0.500	0.441	**0.979**	0.765
BERT (Epoch 2)	0.730	0.500	0.467	**0.979**	0.958
BERT (Epoch 3)	0.706	0.500	0.441	**0.979**	0.958
BERT (Epoch 4)	0.730	0.500	0.441	**0.979**	0.958

Figure 14: The combined results for F1Score.

received a score of 0.833. Dataset 2's most notable results were acquired using ELMo (Epoch 6) which received an evaluation metric score of 0.807. Dataset 3's most noteworthy score was received using the Decision Tree Model with an evaluation metric of 0.826, and secondarily Elmo (Epoch 6) with an evaluation metric of 0.825. ELMo (Epoch 9) received the highest score of 0.990 for Dataset 4. Dataset 5's most notable score was from BERT's epochs 2 through 4. The highest values among all datasets come from Dataset 4.

Figure 14 displays the results for the F1 Score metric. Dataset 1's most noteworthy evaluation metric score was acquired from ELMo (Epoch 6), receiving a score of 0.771. Dataset 2's most notable results were acquired from ELMo (Epoch 6) which received an evaluation metric score of 0.751, and ELMo (Epoch 5) received a score of 0.732. Dataset 3's most noteworthy score was received from ELMo (Epoch 6) with an evaluation metric of 0.771. ELMo (Epoch 9) from Dataset 4 received the highest evaluation metric of 0.990

Accuracy Results	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Decision Tree	0.825	0.943	0.929	**0.943**	0.925
Naïve Bayes (Complement)	0.880	**0.963**	0.944	0.890	0.836
Naïve Bayes (Gaussian)	0.800	0.945	0.922	**0.948**	0.716
SVM	0.860	0.975	0.963	**0.985**	0.836
Random Forrest	0.825	**0.953**	0.825	0.928	0.896
XGBoost	0.880	0.950	**0.964**	0.943	0.896
ELMo (Epoch 1)	0.870	**0.973**	0.870	0.910	0.821
ELMo (Epoch 2)	0.870	**0.973**	0.870	0.955	0.806
ELMo (Epoch 3)	0.880	**0.975**	0.880	0.955	0.836
ELMo (Epoch 4)	0.880	0.975	0.880	**0.978**	0.851
ELMo (Epoch 5)	0.875	0.968	0.875	**0.980**	0.806
ELMo (Epoch 6)	0.855	0.973	0.855	**0.975**	0.896
ELMo (Epoch 7)	0.880	**0.968**	0.880	0.965	0.896
ELMo (Epoch 8)	0.875	**0.975**	0.875	0.970	0.896
ELMo (Epoch 9)	0.865	0.973	0.865	**0.990**	0.881
ELMo (Epoch 10)	0.870	0.968	0.870	**0.968**	0.866
BERT (Epoch 1)	0.915	0.975	0.967	**0.978**	0.761
BERT (Epoch 2)	0.915	0.975	0.968	**0.978**	0.955
BERT (Epoch 3)	0.925	0.975	0.967	**0.978**	0.955
BERT (Epoch 4)	0.915	0.975	0.967	**0.978**	0.955

Figure 15: The combined results for Accuracy.

and model SVM secondarily received a score of 0.985 with both of these values scoring the highest F1 scores out of all of the datasets. In Dataset 5, BERT's 2nd-4th epochs scored the highest with a score 0.958. Dataset 4 has the highest scoring data values for each model as also noted in the AUC results.

Figure 15 displays the results using accuracy metrics. Dataset 1's most noteworthy evaluation metric score was acquired from BERT (Epoch 3), receiving a score of 0.925. ELMo 3rd epoch, 4th epoch, 8th epoch, and all of BERT's epochs scored the highest for Dataset 2 with a score of 0.975. Dataset 3's most noteworthy score was noted from the BERT (Epoch 2) Model with an evaluation metric of 0.968. Dataset 4's Elmo (9 Epoch) received the most significant evaluation metric of 0.990 and model SVM receiving a score of 0.985. Dataset 5's highest scoring model was a three way tie between BERT's 2nd - 4th epochs.

Precision Results	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Decision Tree	0.474	0.167	0.291	**0.900**	0.872
Naïve Bayes (Complement)	**0.833**	0.300	0.043	0.825	0.767
Naïve Bayes (Gaussian)	0.389	0.238	0.109	**0.912**	0.714
SVM	0.583	0.600	0.438	**0.981**	0.780
Random Forest	0.469	0.300	0.469	**0.897**	0.829
XGBoost	0.654	0.091	0.467	**0.904**	0.865
ELMo (Epoch 1)	0.625	0.000	0.625	**0.898**	0.775
ELMo (Epoch 2)	0.750	0.000	0.750	**0.924**	0.756
ELMo (Epoch 3)	0.750	**1.000**	0.750	0.966	0.780
ELMo (Epoch 4)	0.682	**1.000**	0.682	0.958	0.800
ELMo (Epoch 5)	0.684	0.429	0.684	**0.963**	0.744
ELMo (Epoch 6)	0.532	0.500	0.532	**0.954**	0.886
ELMo (Epoch 7)	0.682	0.438	0.682	**0.937**	0.886
ELMo (Epoch 8)	0.667	0.600	0.667	**0.945**	0.909
ELMo (Epoch 9)	0.581	0.500	0.581	**0.981**	0.842
ELMo (Epoch 10)	0.625	0.375	0.625	**0.941**	0.821
BERT (Epoch 1)	0.742	0.556	0.542	**0.958**	0.765
BERT (Epoch 2)	0.742	0.556	0.560	**0.958**	0.919
BERT (Epoch 3)	0.947	0.375	0.542	**0.958**	0.919
BERT (Epoch 4)	0.742	0.375	0.542	**0.958**	0.919

Figure 16: The combined results for Precision.

The majority of models performed the best with Dataset 4, however unlike the scores from AUC and F1 score, Dataset 2 held the best results for Naïve Bayes (Complement), Random Forest, ELMo (Epoch 1), ELMo (Epoch 2), ELMo (Epoch 3), ELMo (Epoch 7), and ELMo (Epoch 8). Dataset 3 held the best results for XGBoost. Dataset 4 has the best results for F1 Score and AUC.

The next metric to identify is precision. Figure 16 displays the results for this metric. Dataset 1's most noteworthy evaluation metric score was acquired from the BERT model with a score of 0.947. Dataset 2's most notable results were acquired from ELMo's 3rd and 4th epochs which received an evaluation metric score of 1.000. Dataset 3's most noteworthy score was received from ELMo's 2nd and 3rd epochs with an evaluation metric of 0.750. Dataset 4's most significant score came from Elmo's 2nd and 3rd epochs which received an evaluation metric of 0.981. Dataset 5's highest scoring result was from BERT's 2nd - 4th epochs with a score of 0.919. Naïve Bayes (Complement) had the highest scoring precision

Recall Results	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Decision Tree	0.844	0.273	0.714	1.000	1.000
Naïve Bayes (Complement)	0.313	0.273	0.029	1.000	0.971
Naïve Bayes (Gaussian)	0.438	0.455	0.171	**0.995**	0.735
SVM	0.438	0.273	0.200	**0.990**	0.941
Random Forrest	0.719	0.545	0.719	0.971	1.000
XGBoost	0.531	0.091	0.200	**0.995**	0.941
ELMo (Epoch 1)	0.469	0.100	0.469	**0.932**	0.912
ELMo (Epoch 2)	0.281	0.100	0.281	**0.995**	0.912
ELMo (Epoch 3)	0.375	0.091	0.375	**0.947**	0.941
ELMo (Epoch 4)	0.469	0.091	0.469	1.000	0.941
ELMo (Epoch 5)	0.406	0.545	0.406	1.000	0.941
ELMo (Epoch 6)	0.781	0.182	0.781	1.000	0.912
ELMo (Epoch 7)	0.469	0.636	0.469	1.000	0.912
ELMo (Epoch 8)	0.438	0.273	0.438	1.000	0.882
ELMo (Epoch 9)	0.563	0.182	0.563	1.000	0.941
ELMo (Epoch 10)	0.469	0.273	0.469	1.000	0.941
BERT (Epoch 1)	0.719	0.455	0.371	1.000	0.765
BERT (Epoch 2)	0.719	0.455	0.400	1.000	1.000
BERT (Epoch 3)	0.563	0.273	0.371	1.000	1.000
BERT (Epoch 4)	0.719	0.273	0.371	1.000	1.000

Figure 17: The combined results for Recall.

value from Dataset 1. ELMo (Epoch 3) and ELMo (Epoch 4) had the highest precision values from Dataset 2.

After analyzing the precision metric, recall was also considered for determining the outcomes of this study. Figure 17 displays the results gathered for recall. Dataset 4 acquired the most significant data overall compared to the rest of the datasets in this thesis as visualized in Figure 17 per the highlighted cells. The Dataset 1's most noteworthy evaluation metric score was acquired from the Decision Tree model with a score of 0.844. Dataset 2's most notable results were acquired from ELMo (Epoch 7) which received an evaluation metric score of 0.636. Dataset 3's most noteworthy score was received from ELMo (Epoch 6) with an evaluation metric of 0.781, and secondarily the Random Forest model received an evaluation metric of 0.719. The highest possible score for recall is 1.0, and in this study

out of twenty seven models and epochs, Dataset 4 contained twenty models and epochs which received a score of 1.0. The fifth datset's highest scoring results were from Decision Tree, Random Forest, BERT (Epoch 2), BERT (Epoch 3), and BERT (Epoch 4). All of these values had a score of 1. Overall, Dataset 4 has the majority of highest values, except Dataset 5 in the case of Random Forest, BERT (Epoch 2), BERT (Epoch 3), and BERT (Epoch 4) which are the highest scoring values from Dataset 5.

5.3 Discussion

These results show various trends from modeling different types of data. Dataset 1's results overall performed in the middle. AUC results from Dataset 1 indicate BERT's 1st, 2nd, and 4th epochs were significant. Dataset 2 and Dataset 3 provided false readings because they were imbalanced datasets. Dataset 4 encouraged overfitting to produce better results because of the number of repeated positive classifications from creation of the dataset. Dataset 5 shows significant results indicating BERT outperformed the other models by a large margin, which is what would be expected from a balanced dataset per current research in the field [10]. In order to fix the imbalance in Dataset 3, oversampling of positive observations and undersampling of negative observations were completed to create Dataset 4 and Dataset 5. Each of the various datasets display different overall trends from the resulting metrics, however datasets four and five produce better results using these processes for balancing the data.

Some of the metrics used are not valid for determining how the model performed because of the distribution of the data. The first example is the accuracy metric when looking at Dataset 2 and Dataset 3. Since these datasets contain approximately 2% of positive tweets, the dataset received a very high accuracy for every model. Accuracy is a better metric when the dataset is a balanced dataset such as Dataset 4 and Dataset 5.

AUC is another metric that has been noted to produce inaccurate results. Rice [9] observed that discusses there are many different RoC Curves that can receive the same AUC value. The AUC metric is "incredibly inaccurate" for comparing models because the sample size is not large enough for a single ROC value to be significantly different from one model to another. When comparing two different AUC Curves that are closely aligned

to each other, then comparing two close curves could be worse than a coin toss. Average squared error has been shown historically to be a better measure overall. This could be due to the fact that the AUC metric is a newer metric, and Average squared error is not. For this reason AUC results will not be further discussed in great detail, except where it further validated BERT's performance in datasets one and five.

The most interesting results came from models that received the highest F1 Score. The model that performed the best in terms of an F1 Score using unaltered data is ELMo (Epoch 6) from Dataset 3 with a score of 0.771. The second best score from an unaltered dataset is ELMo (Epoch 7) from Dataset 2 with a score of 0.751. The best score overall is 0.990 from ELMo (Epoch 9) of Dataset 4. The F1 Score metric is meaningful compared to other metrics, because it is not affected by imbalanced datasets. Due to the amount of highly imbalanced data that was gathered during this study, it is fair to state that use of an F1 Score would be reliable in future work for live health care tracking, unless there is a filtering process to balance and clean the incoming data.

High precision is an alternative approach to using an F1 Score, and indicates which model is more selective in regard to properly identifying positive classifications for live health care tracking. This can promote more appropriate resource dissemination as a consequence. For example, if a targeted population is identified to be in need of a scarce resource such as a vaccine (i.e. tamiflu), then making precise predictions to indicate where to send this resource is invaluable. A model with high precision and low recall will have a greater likelihood to accurately select an area which will truly be in need of this specific resource and could enable organizations including hospitals to stock tamiflu in anticipation of the identified potential outbreak. The combination of high precision, low recall, and F1 Scores as evaluation metrics are arguably more reliable than the F1 Score alone. F1 Score is a valid measure overall to assess how well a model is performing, because it relies on both recall and precision to create the metric. However in most practical situations it is not possible to obtain both a high precision value and a high recall value, and the results are often an inverse relationship in most real world scenarios. Therefore, acknowledging this difference and analyzing this disparity will help provide another perspective regarding how to critically analyze the models. ELMo (Epoch 2) from Dataset 3 performed the best for

high precision and recall. ELMo (Epoch 2) received a precision score of 0.750 and a recall score of 0.281 from Dataset 3. These results indicate that ELMo would be the recommended model to use in the event that there is a need for tracking live health care information, as it will predict positive classifications less frequently, with greater precision compared to the other models. This would assure that health care providers could appropriately distribute resources without waste.

High recall is significant for determining where to send resources that are plentiful, which is not necessarily appropriate for dissemination of health care resources unless they exist in excess. This combination of metrics guarantees coverage, because the model exhibits an increased rate of inaccurate positive classifications for tweets related to the flu. Outcomes from datasets one and three indicate that the Decision Tree is the best choice for low precision and high recall. In most cases the health care system is looking to provide cost effective health care, not disperse resources without convincing evidence that a need exists for providing services.

The primary goal and purpose of this study was to identify models that can make predictions related to flu prevalence based on incoming unfiltered, noisy data from Twitter. Several models indicate false positive occurrence based on the data trained for the model. ELMo (Epoch 7) performed the best overall when considering the realistic implementations of a model, especially since incoming data would typically be imbalanced. BERT (Epoch 2) from Dataset 5 indicated the best results if the incoming data is balanced. A balanced dataset could potentially be obtained through refining the stream filter terms. If the dataset has a small percent of reoccurring positive classifications, then the best model would be ELMo for an imbalanced dataset. As indicated previously there are different perspectives which define the type of analysis that could be indicated when using a model for live health care tracking and epidemiological reporting. The Decision Tree demonstrated to be the best performing model if low precision high recall is illustrated for live tracking with broad coverage, and less accurate predictions. Overall ELMo scored the best per F1 Score and high precision low recall for dissemination of scarce resources, and improving the efficiency of live health care tracking and epidemiological information.

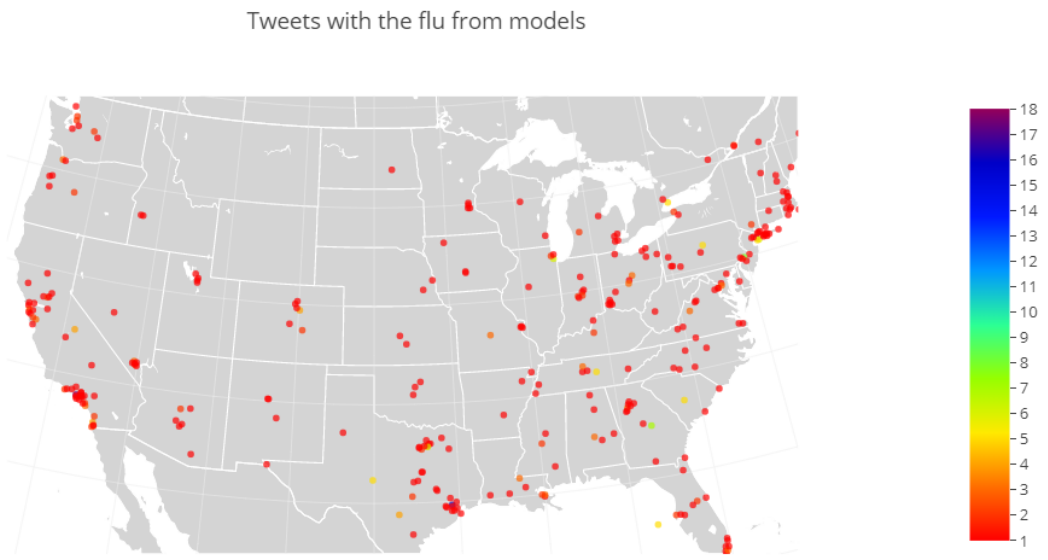

Figure 18: A heatmap of some tweets recorded for the project related to the flu.

5.4 Heatmap

The heatmap is used to visually exhibit the results gathered during this project. The Plotly library was used for generating Figure 18. This visual is a representation of where the Tweets were positively identified for the influenza virus. The heatmap uses tweets that were collected from the stream and plots the coordinates associated with the tweets onto the heatmap. The code combines all of the tweets with the same coordinates and keeps track of those records before plotting the heatmap. An ensemble method is used to classify the tweets. The best results from datasets two and three were assessed for each model and implemented into the ensemble to generate the heatmap.

6 Conclusion and Future Work

In this thesis, we examined the effectiveness of various machine learning models to classify real-time Twitter tweets and stream them into a live heatmap for epidemiological reporting. This thesis illustrates results from over 100,000 different trained models. Five metrics were used to evaluate the efficacy of these models. ELMo achieved the highest F1 Score of 0.771. ELMo also outperformed the other models when considering the metric of high precision. These results show substantial implications when considering their

utilization for epidemiological reporting in order to identify need and distribute scarcely available materials. The ELMo model displayed precise classifications that are necessary for a medical system that has scare resources that should not be squandered. The ELMo model performed the best in F1 score as well as having the best scores for high precision, and overall would be the best model of the various models tested in this thesis. These results are worth consideration and could benefit the healthcare system tremendously when analyzing the timely management and distribution of healthcare resources.

There is a lot of opportunity to improve upon the research presented in this thesis. The investigators of this study gathered and archived over 180,000 tweets, and secondary to time constraints only 9,000 of them were hand-classified. If time constraints were not a barrier, more data cleaning would have occurred and a higher number of tweets would have been hand classified. Corrupted tweets, tweets containing '@' mention, and retweets would be removed in an attempt to keep the data as clean as possible for analysis and training. This would also be maintained in the stream's incoming data from Twitter in order to generate the live heatmap. Training on all of the data collected would greatly improve the models results, as seen from the third iteration of modeling compared to the second iteration.

N-gram feature generation is advisable for future studies which require preprocessing for each tweet. As seen in Carchiolo's [6] research, this step groups words together (e.g. noun-verb-object) in order to improve the accuracy of the model. N-grams would enable word-based pairings correlated to the positively classified tweets. By implementing n-gram analysis as an alternative to bag-of-words, the investigator could potentially place more weight on specific word pairings, and ultimately improve the preprocessing step of data preparation before modeling occurs.

The data processing portion of this process removed the tweets that were from bi-lingual users of Twitter. Tweets from bi-lingual users accounted for approximately 45% of tweets that were removed from the dataset. If these tweets were classified, then the coverage of tweets could be greatly increased.

Another element of note for improvement is including other data stream feed from other media sources to increase coverage on people that use social media. Including feeds from Facebook, Instagram or other popular media platforms would reduce the bias introduced

into the project from using Twitter as a data source. Thus increasing the demographic of people that this project could impact.

Another option to improve coverage of the heatmap is to introduce parallelization. In order to have the capacity to incorporate parallelization, moving data onto a cloud and utilizing this cloud as a mobile resource is recommended. This step would allow the investigator wider access to stored data from the project (i.e. anywhere the investigator has a computer and internet access). The cloud storage also allows the project scale to be increased and inclusive of parallelization (i.e. multiple Twitter streams gathering and storing data at once). Parallelization would improve the time efficiency of data gathering, however repeat tweets would have to be accounted for in the filtering process. This would introduce the capacity for multiple streams to be processed at once, and simultaneously produce the results by generating one heatmap for all of the Twitter streams.

References

[1] J. Alammar, *The illustrated bert, elmo, and co. (how nlp cracked tranfer learning)*, `http://jalammar.github.io/illustrated-bert/`, Accessed: 2019-08-05 (cit. on p. 25).

[2] A. Alessa and M. Faezipour, "A review of influenza detection and prediction through social networking sites", *Alessa and Faezipour Theoretical Biology and Medical Modelling*, 2018 (cit. on pp. 3, 7).

[3] S. Aslam, *Twitter by the numbers: Stats, demographics fun facts*, `https://www.omnicoreagency.com/twitter-statistics/`, Accessed: 2019-07-31 (cit. on p. 1).

[4] S. Briand, A. Mounts, and M. Chamberland, "Challenges of global surveillance during an influenza pandemic", *Public Health*, vol. 125, pp. 247–256, 5 May 2011 (cit. on p. 2).

[5] J. Cai, *Using machine learning to analyze twitter for real time influenza surveillance*, `https://medium.com/@justinzcai/using-machine-learning-to-analyze-twitter-for-real-time-influenza-surveillance-585981462eac`, Accessed: 2018-11-29 (cit. on pp. 8, 14).

[6] V. Carchiolo, A. Longheu, and M. Malgeri, "Using twitter data and sentiment analysis to study diseases dynamics", in *Proceedings of the 6th International Conference on Information Technology in Bio- and Medical Informatics - Volume 9267*, ser. ITBAM 2015, Valencia, Spain: Springer-Verlag New York, Inc., 2015, pp. 16–24, ISBN: 978-3-319-22740-5 (cit. on pp. 5, 41).

[7] I. Chung-Hai Fung, Z. Tsz Ho Tse, and F. King-Wa, "The use of social media in public health surveillance", *Western Pacific Surveillance and Response Journal*, vol. 6, 3–6, 2015 (cit. on p. 1).

[8] A. Culotta, "Towards detecting influenza epidemics by analyzing twitter messages", in *Proceedings of the First Workshop on Social Media Analytics*, ser. SOMA '10, Washington D.C., District of Columbia: ACM, 2010, pp. 115–122, ISBN: 978-1-4503-0217-3. [Online]. Available: `http://doi.acm.org/10.1145/1964858.1964874` (cit. on pp. 3, 5, 6).

[9] R. Danial, *Is the auc the best measure?*, `https://www.kdnuggets.com/2010/09/pub-is-auc-the-best-measure.html`, Accessed: 2019-08-01 (cit. on p. 37).

[10] J. Devlin, M.-W. Change, K. Lee, K. Toutanove, and G. A. Language, "Bert: Pre-training of deep bidirectional transformers for language understanding", 2018 (cit. on pp. 9, 15, 25, 37).

[11] C. of Disease Control and Prevention, *Flu symptoms and complications*, `https://www.cdc.gov/flu/symptoms/symptoms.htm`, Accessed: 2018-01-20 (cit. on p. 12).

[12] S. Doan, L. Ohno-Machado, and N. Collier, "Enhancing twitter data analysis with simple semantic filtering: Example in tracking influenza-like illnesses", *CoRR*, vol. abs/1210.0848, 2012. eprint: `1210.0848` (cit. on pp. 5, 6).

[13] DOMO, *Data never sleeps*, `https://www.domo.com/solution/data-never-sleeps-6`, Accessed: 2019-07-31 (cit. on p. 1).

[14] A. Fogg, *Anthony goldbloom gives you the secret to winning kaggle competitions*, `https://www.import.io/post/how-to-win-a-kaggle-competition//`, Accessed: 2019-10-26 (cit. on p. 22).

[15] Q. Fottrell, *People spend most of their waking hours staring at screens*, https://www.marketwatch.com/story/people-are-spending-most-of-their-waking-hours-staring-at-screens-2018-08-01, Accessed: 2019-07-28 (cit. on p. 1).

[16] J. Ginsberg, M. Mohebbi, R. S Patel, L. Brammer, M. Smolinski, and L. Brilliant, "Detecting influenza epidemics using search engine query data", *Nature*, vol. 457, pp. 1012–4, Dec. 2008 (cit. on p. 2).

[17] J. Han, M. Kamber, and J. Pei, *Data Mining Concepts and Techniques*, Third. Morgan Kaufmann Publishers (cit. on p. 19).

[18] A. Lamb, M. J. Paul, and M. Dredze, "Separating fact from fear: Tracking flu infections on twitter", in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 789–795 (cit. on pp. 5, 8).

[19] D. Lazer and R. Kennedy, *What we can learn from the epic failure of google flu trends*, https://www.wired.com/2015/10/can-learn-epic-failure-google-flu-trends/, Accessed: 2019-07-31 (cit. on p. 2).

[20] K. Lee, A. Agrawal, and A. Choudhary, "Real-time disease surveillance using twitter data: Demonstration on flu and cancer", in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13, Chicago, Illinois, USA: ACM, 2013, pp. 1474–1477, ISBN: 978-1-4503-2174-7 (cit. on p. 4).

[21] W. H. Organization, *Influenza seasonal*, https://www.who.int/news-room/fact-sheets/detail/influenza-(seasonal), Accessed: 2019-08-01 (cit. on p. 1).

[22] P. Palanisamy, V. Yadav, and H. Elchuri, "Serendio: Simple and practical lexicon based approach to sentiment analysis", in *SemEval@NAACL-HLT*, 2013 (cit. on p. 4).

[23] M. J. Paul and M. Dredze, "You are what you tweet: Analyzing twitter for public health", in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011 (cit. on p. 3).

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011 (cit. on pp. 13, 19).

[25] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, *Deep contextualized word representations*, 2018. arXiv: 1802.05365 [cs.CL] (cit. on pp. 9, 15, 25).

[26] A. Signorini, A. M. Segre, and P. M. Polgreen, "The use of twitter to track levels of disease activity and public concern in the u.s. during the influenza a h1n1 pandemic", *PLOS ONE*, vol. 6, no. 5, pp. 1–10, May 2011. DOI: 10.1371/journal.pone.0019467. [Online]. Available: https://doi.org/10.1371/journal.pone.0019467 (cit. on pp. 3, 6, 7).

[27] SKLearn, *Scikit-learn: Machine learning in python*, https://scikit-learn.org/stable/, Accessed: 2019-11-23 (cit. on pp. viii, 25).

[28] S. Wakamiya, M. Morita, Y. Kano, T. Ohkuma, and E. Aramaki, "Overview of the ntcir-13: Medweb task", pp. 40–49, Dec. 5, 2017 (cit. on p. 7).

Author:	Elisha David Brunette
Place of Birth:	Spokane, Washington
Undergraduate School Attended:	Washington State University
	Eastern Washington University
Degrees Awarded:	Bachelor of Science, 2016, Eastern Washington University
Honors and Awards:	Graduate Assistantship, Computer Science Department
	2017 - 2019 Eastern Washington University
Professional Experience:	RiskLens, Spokane, WA:
	Junior Data Scientist: 2019
	Data Scientist Intern: 2018
	Software Engineer Intern: 2018