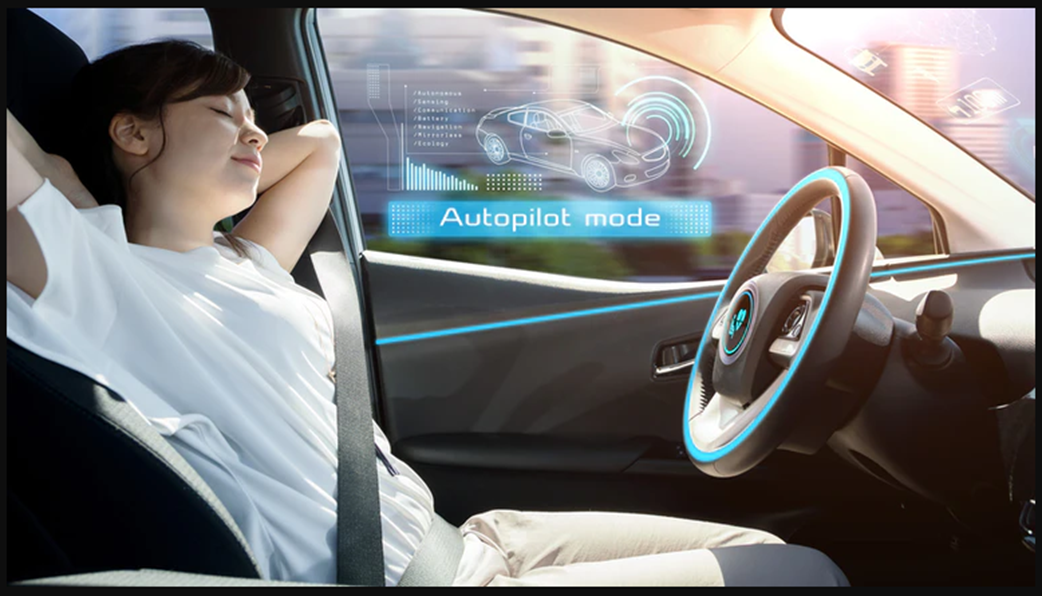
****

**IST 736 – TEXT MINING**

**Dr. Ami Gates**

**Self-Driving Cars –  
The Importance and Significance**

**By**

**Jiang Zhang | Lucas Zarzeczny | Patrick Aslakson**

**Introduction**

As we keep moving forward as a society in the twenty-first century, so too does technology and the advancements that it brings. Autonomous self-driving cars will be a part of the future as we know it. However, at the present time there are no truly autonomous self-driving on the market. Many conventional car makers have cars that will ‘self-park’ with some assistance from the driver, this is an example of an automated vehicle. None are quite up to the task of being a truly autonomous driving car. Tesla the leader in electric vehicle manufacturing has in recent years introduced an option in its vehicles with some capabilities to drive itself with little or no involvement of the driver. We would like to discuss with the importance and significance towards autonomous vehicles and self-driving cars in general and give an example of what it would be like to write a program for a self-driving car.

The history of the automobile is now just about 135 years old. Many great car manufacturers have come and unfortunately many have gone, with only the greatest of innovators still in business and making cars that are on the road today. In fact, at the turn of the 20th century there were as many as thirty manufacturers who produced approximately 2,500 cars. Just ten years later, the number of automobile manufacturers had swelled to nearly 500 including one of the most iconic, Henry Ford’s Ford Motor Company. By 1929, this number was a mere 44 left producing vehicles for the everyday consumer.

Throughout this short history, experiments for automated automobiles have been conducted since the 1920s and came into fruition in trial in actual automobiles in the 1950s. It was not until the gas crisis of the late 1970s that the first working semi-automated vehicle was actually tested in Japan. The vehicle utilized cameras, a quasi-computer system, and was assisted by type of raised rail system to guide the vehicle.

In the mid-1980s, a joint project with Carnegie Mellon University and the United States Defense Advanced Research Projects Agency (DARPA) began a program that resulted in droving the vehicle 2,848 miles across America in 1995, with 98% of it driven autonomously. Carnegie Mellon’s Navlab record stood unmatched for two decades until 2015, when Delphi improved upon the technology and piloting an Audi over 3,400 miles through 15 states while remaining in self-driving mode 99% of the time. Surprisingly, that title does not go to the electric vehicle manufacturer, Tesla, whom most people believe and attribute as being the front runner in autonomous driving vehicles today.

Fast forward to today, 2020, the National Transportation Safety Board (NTSB) chairman stated that no self-driving cars were available for consumers to purchase in the US in 2020:

” There is not a vehicle currently available to US consumers that is self-driving. Period. Every vehicle sold to US consumers still requires the driver to be actively engaged in the driving task, even when advanced driver assistance systems are activated. If you are selling a car with an advanced driver assistance system, you’re not selling a self-driving car. If you are driving a car with an advanced driver assistance system, you don’t own a self-driving car.”(1)

Yet, Waymo, formerly the Google Self-Driving Car Project, was the first company to commercialize a fully autonomous taxi service in the US, in Phoenix, Arizona. “In October 2020, Waymo's service was opened to the public.” However, Waymo’s vehicles are not available for consumers.

So, both the importance and significance of the self-driving car can be seen very easily. We as humans are seeing a technology develop in our lifetimes that will be reality long before we leave this consciousness for the ethereal world beyond as current efforts progress. It is an exciting time to witness and perhaps with a little bit of persistence and luck be able to be a part of as a budding data scientist. There are a plethora of great technological advancements that require the professional data scientist in this new millennia including self-driving cars and beyond in the outer reaches of space with numerous advances in manufacturing and Science.

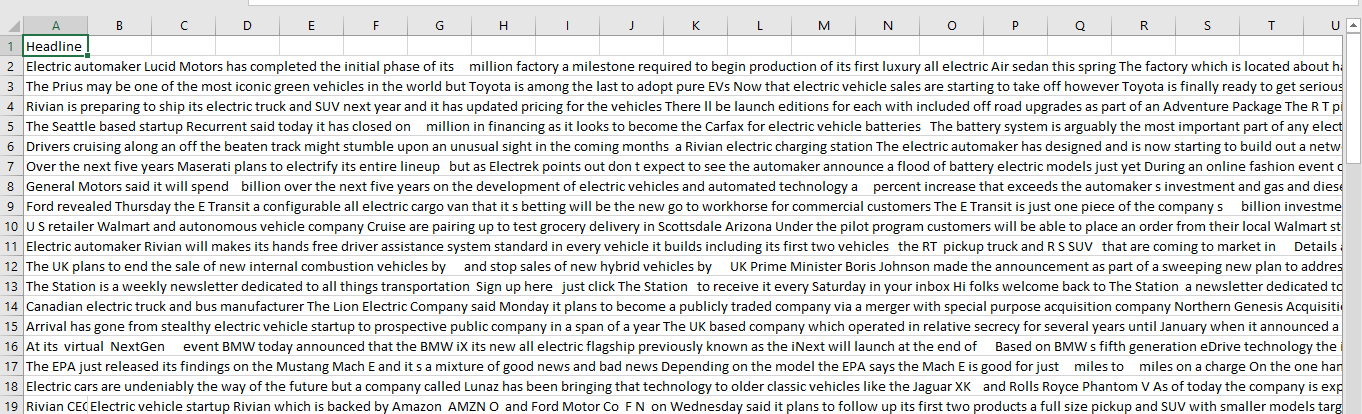
Anyone up for a trip to Mars? It’s on the event horizon.

**Analysis & Models**

About the Data

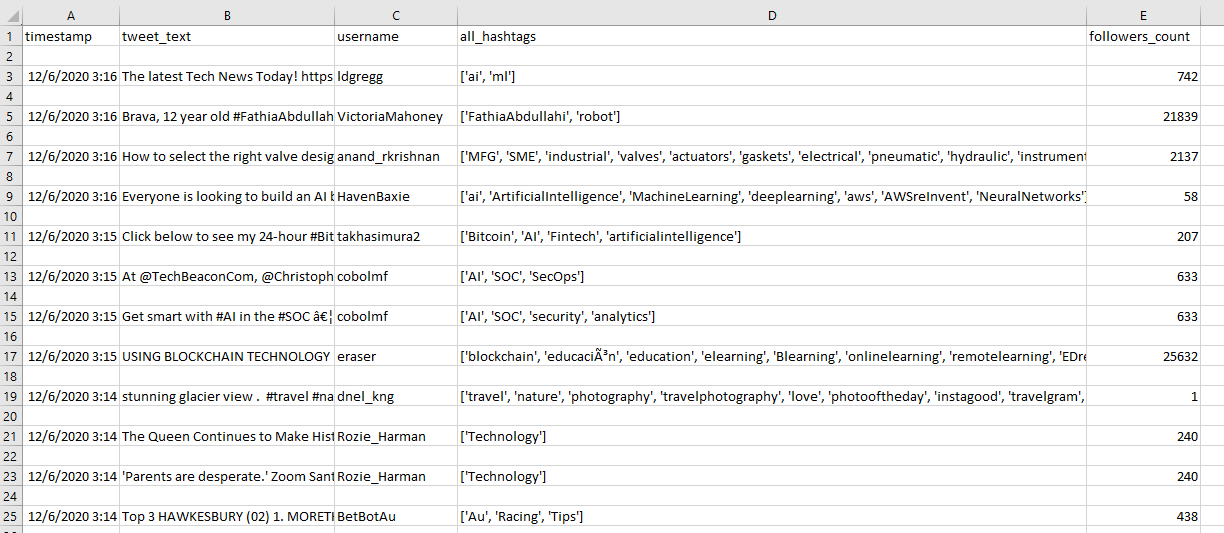
A news API was utilized from https://newsapi.org/ to pull articles from the internet concerning several topics, autonomous vehicles, electric vehicles, and Tesla. The raw data was pulled into three csv files, respectively AV, EV, and Tesla which were accumulated over several days. The data from these models are then organized into one single file with a singular feature, headline. This preprocessing step eliminates the need to later cull any columns from the dataframe that will be created from the data.

Here is screenshot of the raw file with approximately 100 rows, each representing a news article:



In addition to the NewsAPI, a large dataset was acquired from Twitter with the Twitter API where a csv file containing approximately 1000 Tweets utilizing various hashtags. These hashtags included: #autonomous, #driverless, #driverlesscars, #autonomousvehicles, #selfdriving, and #self-driving car. These hashtags were chosen as the most relevant to the topic as a whole. Five features make up the columns of the file including: timestamp, tweet\_text, username, all\_hashtags, and followers\_count.

The following screen shot is the raw Tweets file:



Furthermore, the all-hashtags column in the Twitter dataset shows other hashtags that were also embedded with the Tweets searched hashtags. These additional hashtags could be helpful to use as search criteria in the event that searched hashtags do not provide enough data alone.

Ultimately, the data is then scrubbed utilizing the regular expressions (regex) coding and rewritten to the file for further evaluation, i.e., converted to a pandas dataframe, subjected to count vectorization, and finally run through the LDA model and plotted into a topic plot for a concise visualization summary useful in boardroom presentations. We will elaborate more on this process next in the Models section.

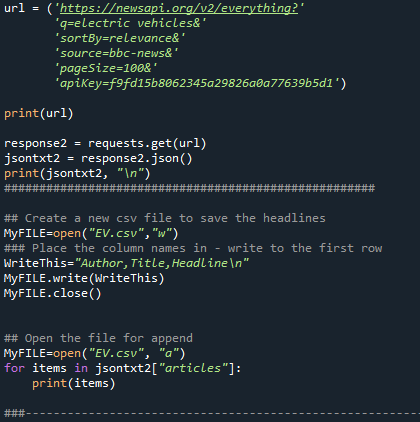
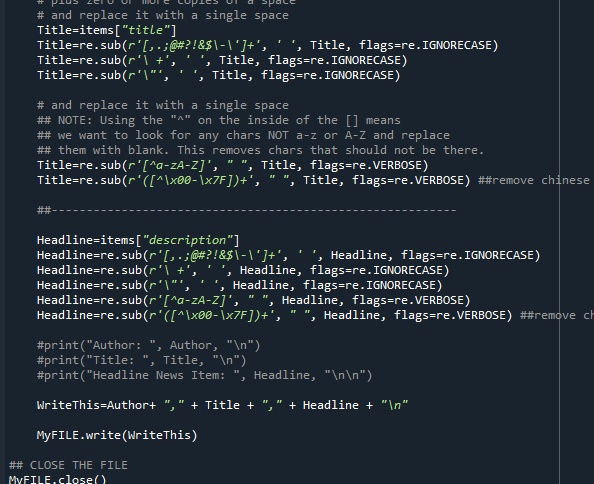
**Preprocessing**

The model for gathering the NewsAPI data for each topic are identical with exception for the topic (q=electric vehicles, q=autonomous vehicle, and q=Tesla). The test set was obtained via Newsapi.org using the News API. News API provides breaking news headlines, blogs, and other information over the web. In order to obtain this information, python code needs to be created to pull this information from the API. What is an API? An API stands for Application Programming Interface. An API lists a bunch of operations that developers can use to pull in information that is stored and owned by different sources. For instance, in order for a developer to obtain tweets and analyze this information, the developer needs to establish a Twitter API that will give the appropriate permissions for the developer to pull information owned by Twitter. After implementing the API, the data is pulled in per JSON format. The data is also unstructured text, not in a dataframe. Per the API criteria, which is described in a dictionary, they keyword “q” was used as the topic to search across BBC News. What is JSON?

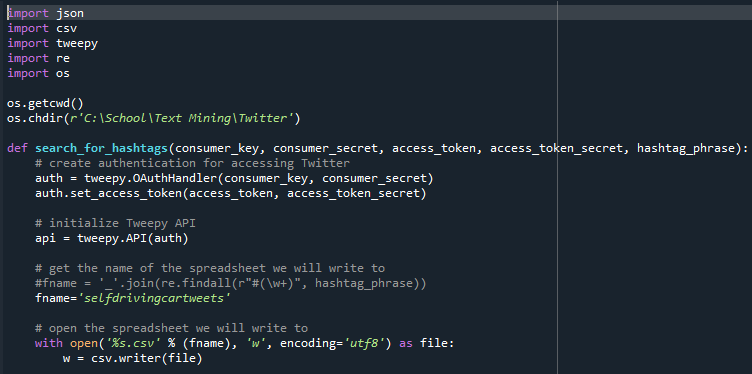
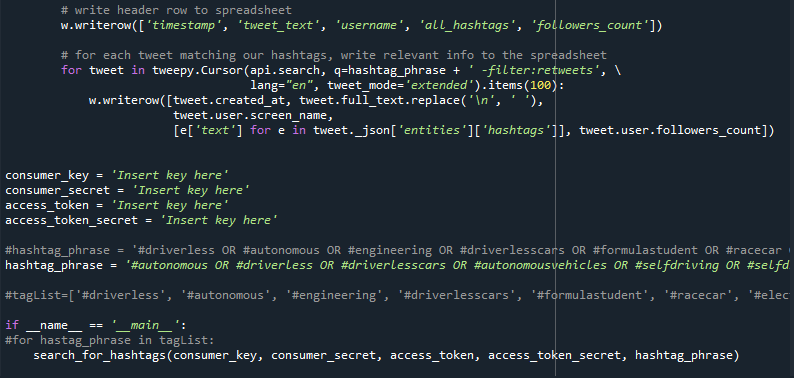
JSON stands for JavaScript Object Notation and is a way of storing and transporting data. JSON is often used when data is sent from a server to a web page or visa versa. JSON code is structured with curly braces ({}). Per the API pull, the below shows the data in JSON format.

Using Python, the data was transformed into a data frame. Each row is a string of content obtained from the API. In other words, each row is a sentence containing information about a given topic . After preprocessing, a word cloud was created to show the most relevant words. A word cloud is a useful tool to determine the most common words within the dataset.

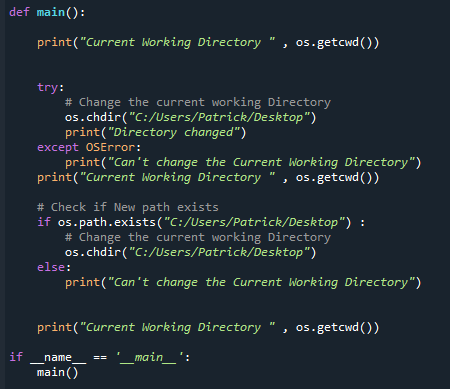
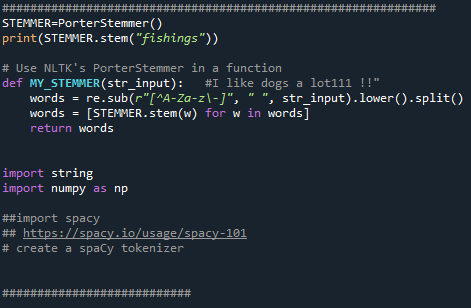
The following screenshots show the code for accessing the API, scrubbing, and writing the data to the csv file:

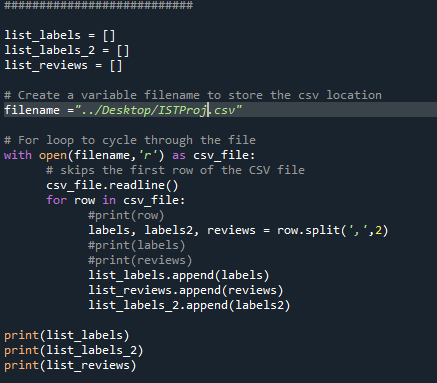
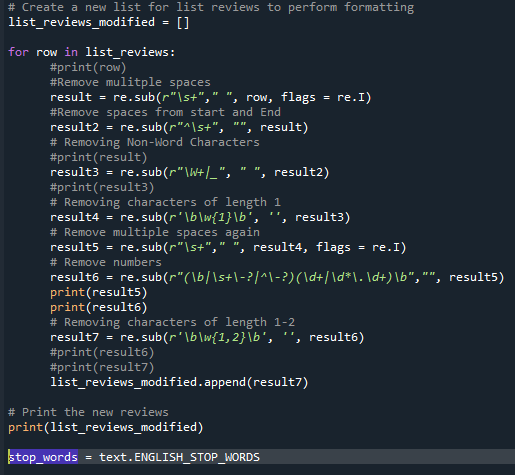
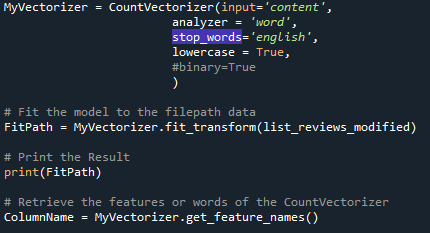
Similarly, the following screenshots show the model for accessing the Twitter API:

Next, we set our working directory to the desktop for ease. Then, run the PortStemmer.

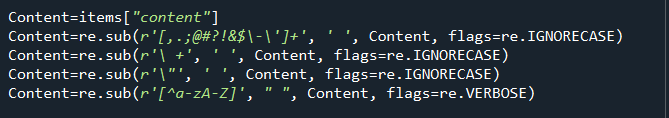
From here, we define our lists for data and labels for the datasets, scrub the datasets, define stop\_words, and append the cleaned data to new lists. Now, the datasets are run through the instantiated Count Vectorizer, and converted to a Pandas dataframe where they are fit to an array.

The first step included importing the data via python into the Spyder IDE.

The second preprocessing step included calling the News API/Twitter API and converting that information into a JSON format.

The third preprocessing step included applying regular expressions to the content and writing the end result to a csv file. The following regular expressions were applied:



The fourth step included reading the csv file and converting the content into a Dataframe using the Pandas pd.read\_csv function. Three regular expression were used to remove punctuation, convert all characters to lowercase, and to remove square brackets. All of these steps were used to clean the data and remove the junk. Essentially, the objective is to capture the meaningful words only.

The fifth NLP preprocessing step included removing spaces from the start and end of a row. Not only is it important to remove extra white space, it is important to remove whitespace after a sentence. Again, the purpose of this preprocessing step is to ensure that the labels can be extracted.

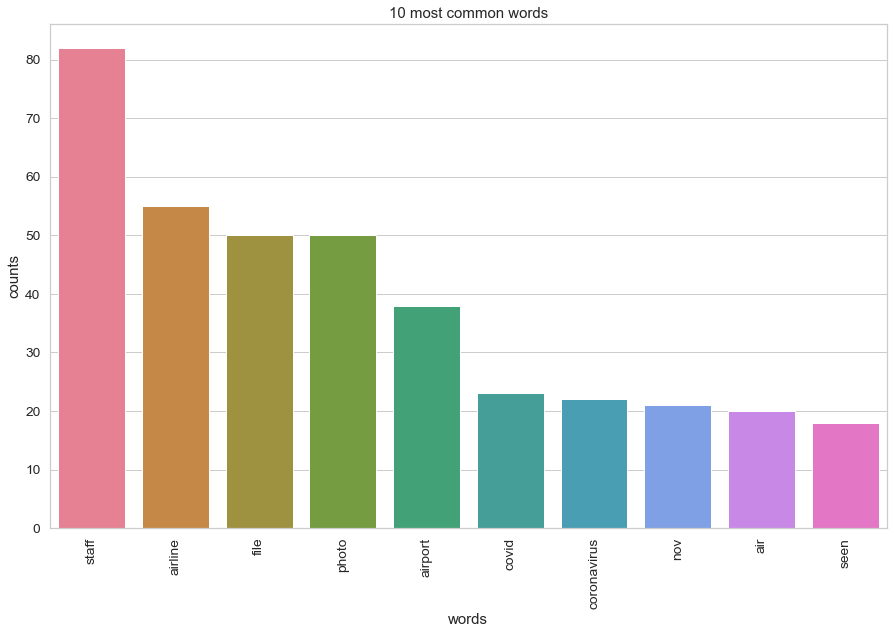
The sixth NLP preprocessing step included removing non-word characters. This process is similar to removing stop words. However, this step is completed to remove any unwanted characters after the labels. The regex function used is below:

The seventh NLP preprocessing step included removing characters of length 1-2. This is completed because there are numerous words and letters such as: [I, up, to, a, v, s, on, of] that do not add any value to the overall sentiment. These words and letters are similar to stop words, but the stop word’s NLTK library does not remove everything that is needed in this analysis. The regex function used is below:

The eighth NLP preprocessing step included removing numbers. Numbers without context provide no additional meaning, unless these numbers were within bigrams and trigrams.

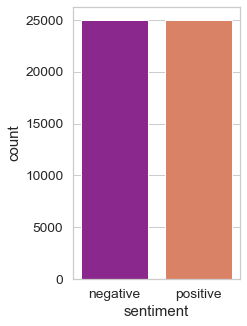
The next step included removing stop words. Stop words are considered to be extremely common words like “the” which would add little to no value to the meaning or context of the document. This paper will not dive deep into the technicalities of stop word creation. The paper used the NLTK’s stop word list. A couple words were flagged as irrelevant after a word cloud was created, which shows some of the most common words. The words ‘reuters’ and ‘char’ appeared. Both of the words are irrelevant and don’t add much meaning. Char is not really a word and reuters is just a news source. These two words were combined to the original stop word list.

Next, a function was created to plot the ten most common words. This helps confirm the word cloud results. It is important to note that this function was created after the stop word list was implemented. Thus, the words ‘char’ and ‘reuters’ should not be included.



In addition, the Sklearn’s CountVectorizer class was implemented to convert the dataframe into tokens. The next preprocessing step included returning all tokens as lowercase. As mentioned before, tokenization is case sensitive. However, whether a word is uppercase, or lowercase does not change the sentiment in the context of this paper. This preprocessing step was performed during the initialization of Count Vectorizer.

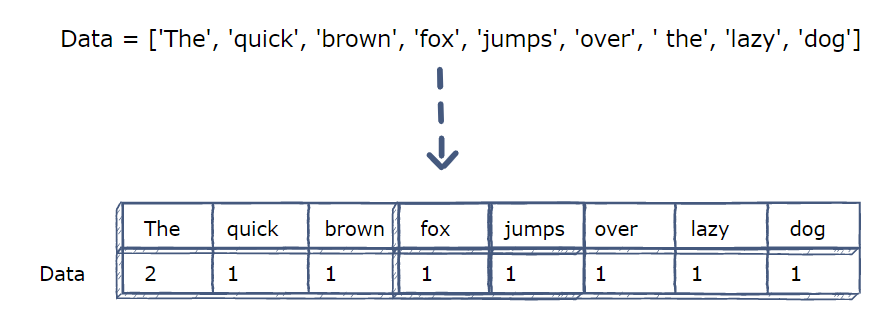
In preparation to then run tests on this data, we must train our models on a training dataset, we chose the IMDB movies sentiment dataset to train our models because it is large enough with fifty-thousand lines equally distributed with twenty-five thousand positive and negative sentiment lines of data. The equal distribution of the positive and negative sentiment counts eliminates much of the risk of bias in this training dataset, at least in this instance that is the intention. We will elaborate further on the entire process



**Models**

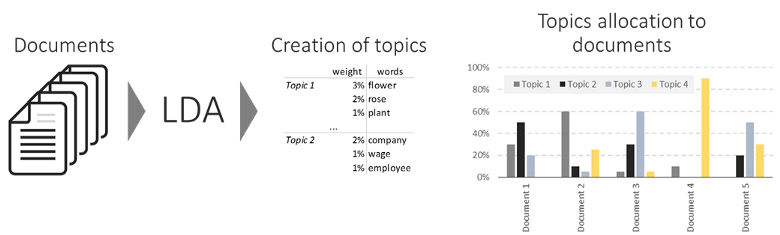
The main models used in this analysis were LDA, Count Vectorizer, TFIDF Vectorizer, Bernoulli Multi Nominal Classification, Support Vector Machines, and Recurrent Neural Networks.

CountVectorizer is a library within SKLEARN. CountVectorizer essentially creates tokens by parsing the text into individual components. Each word becomes a token. So, if a sentence had 5 words, CountVectorizer would create 5 tokens. However, computers only understand numerical values. Thus, these tokens are encoded as integers with each integer representing the frequency of occurrence of each token. This is called feature extraction or vectorization. Sklearn’s CountVectorizer is commonly used to convert a collection of text (analyst reviews) to a vector of token counts. In other words, it provides a frequency of the occurrence of each word. For this analysis, there are three main labels (Positive, Neutral, and Negative). Thus, for each observation or row (tweet) under the label, a vector of numbers representing the frequency of the word (column/feature) are generated - these are the values within the data frame. Each value will be at the intersection of the observation and a particular column or feature. For instance, if one observation under Positive included the word “Delta” two times, this would be the value of 2. Another example is below:



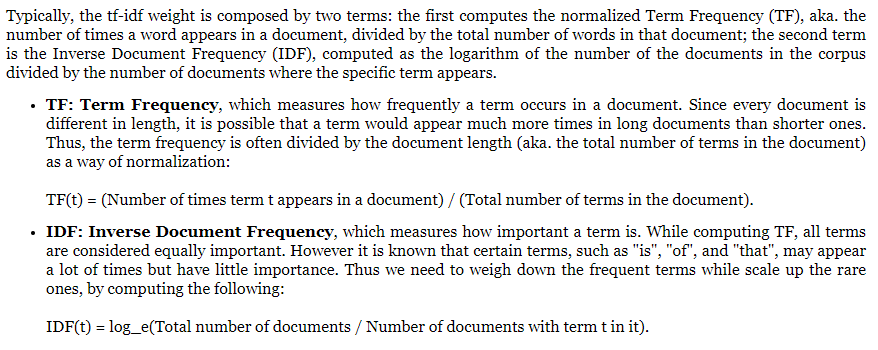
CountVectorizer has many parameters. This analysis used three parameters. The input parameter was set to ‘content’ since the input source was an array of string values. The stop words parameter was set to the default NLTK stop word list. Finally, even though this is the default, it was important to show that lowercase was set to true. If this was set to False, the same word, if appearing once with lowercase and second with uppercase) would consist of two tokens. In other words, tokenization is case sensitive. This analysis was completed on all words being lowercased. Case sensitivity would not have an impact on sentiment in this context. Finally, the ngram\_range was set to (1,1) meaning that each character/word was considered one token, it was not paired with any other word or token. The max\_features parameter was set to 150, meaning the top 150 features will only be included within the prediction.

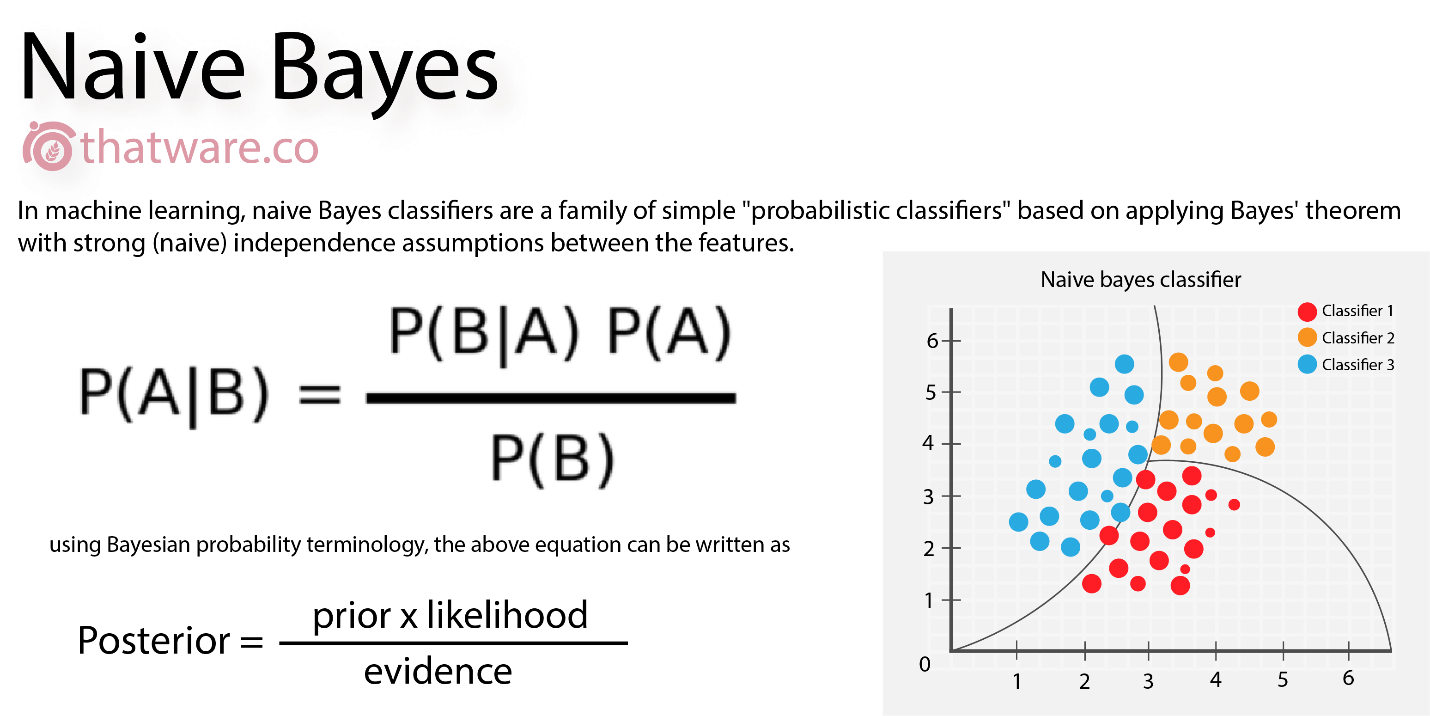
The next model was the Latent Dirichlet Allocation model from Sklearn. This model is used for topic modeling. Topic modeling provides the top topics that best describe the data. These topics will only be provided during the topic modeling process (latent). The LDA imagines a fixed set of topics. Each topic represents a set of words. The goal of the LDA is to map all the documents to the topics in a way such the words in each document are mostly captures by the topics.



The third model was Sklearn’s TF-IDF Vectorizer. None of the parameters changed for this model compared to using the CountVectorizer Model. Thus, the initialization of TF-IDF Vectorizer was exactly the same as the initiation of the CountVectorizer. What does TF-IDF mean?

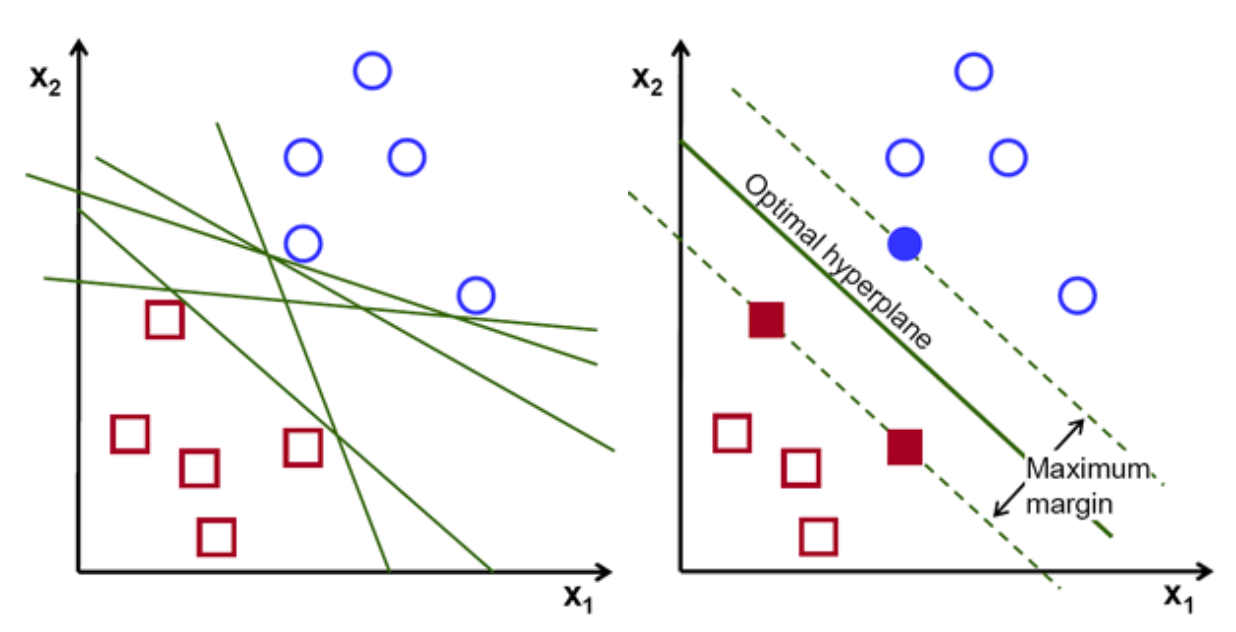
TF-IDF stands for term frequency-inverse document frequency. The tf-idf weight is a weight often used in information retrieval and text mining. This is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Below is a table that shows the formulas for TF-IDF.





The fourth model was Sklearn’s Bernoulli Model. The Bernoulli distribution is the discrete probability distribution of a random variable which takes one value 1 with probability p and the value 0 with probability 1-p. Thus, this model results in a yes/no type of response or 0 and 1. The outcomes of this model tend to be Boolean-valued: a single outcome whose value is success/yes/true with probability p and failure/no/false with probability 1-p. Thus, for Sklearn’s Bernoulli class, it is designed for a binary/Boolean feature. The Bernoulli model is very similar to the Multinomial Naïve Bayes.

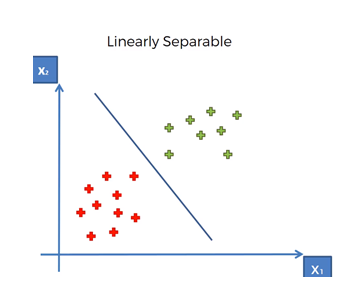
The fifth model was Sklearn’s Support Vector Machine. The support vector machine is a supervised machine learning algorithm because it trains on labeled data sets. The support vector machine helps to classify datapoints based on hyperplanes. In other words, it finds a hyperplane in an N-Dimensional space (N- the number of features) that distinctly classifies the data points



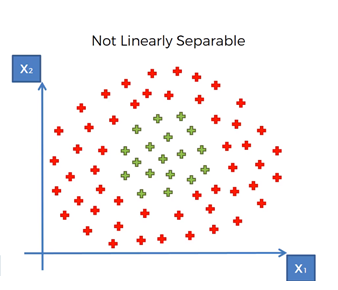
The objective of the SVM is to find a plane that has the maximum margin, the maximum distance between datapoints of both classes (or multiple classes). Maximizing the distance of the margin helps improve the accuracy of the model as well as provide reinforcement so future data points can be classified with better confidence. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Support vectors are the data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. The margin is maximized using the support vectors.

This research paper uses the svm.SVC class from sklearn. The analysis is completed on four instances of this model by adjusting the kernel parameter. The kernel parameter specifies the kernel type used in the algorithm. Think of the kernel as the activation function. This paper uses the following kernels: linear, poly, rbf, and sigmoid.

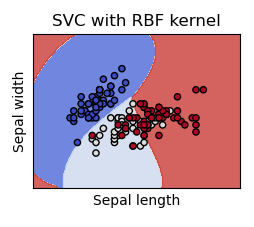
The linear kernel function is the simplest of the kernel functions and is best used when the data is less complex and linearly separable. The below image shows that a linear line perfectly divides the dataset into two classes.



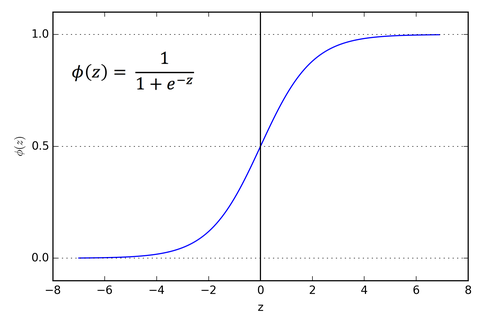
The polynomial kernel is better suited for more complex data, higher dimensional data, and nonlinear data. It is also better suited when the training data is normalized, and this kernel does have adjustable parameters: alpha (slope) and c (constant) and d (polynomial degree). The below image shows that a linear function would not be suitable, but a polynomial function is given the complex nature of the data.



The rbf or Radial Basis Function is the most generalized form of kernelization and is one of the most widely used kernels due to its similarity to the Gaussian Distribution. The RBF kernel function computes the similarity or how close two data points are to each other. The RBF kernel does a tremendous job at classifying data points that are non-linearly separable.

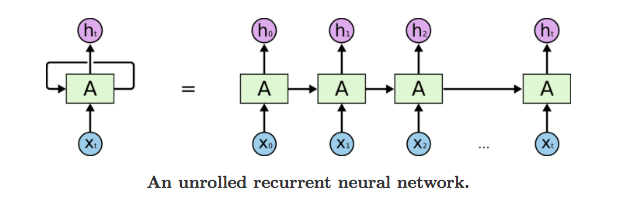


Finally, the Sigmoid kernel is similar to the sigmoid activation function which is generally used within neural networks. The sigmoid function helps a neural network predict the binary predicted outcome of the input data. The sigmoid function predicts the probability of a result occurring between 0 and 1. (1 means it occurs and 0 means it does not occur).

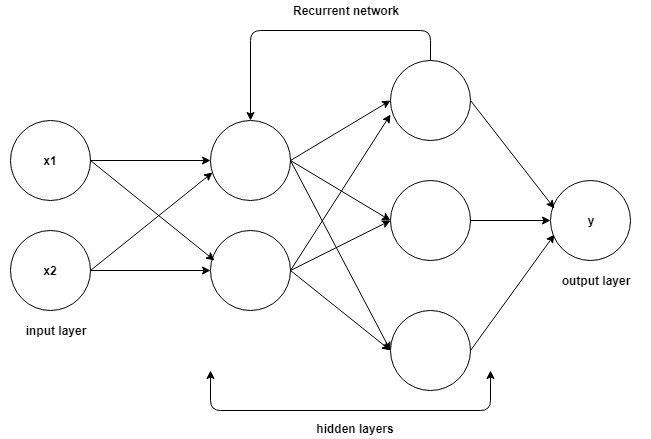


The final model used was a Recurrent Neural Network, specifically the LSTM which stands for Long Short-Term Memory. Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For predicting, it considers the current input and the output that it has learned from the previous input. RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.

A LSTM Memory Cell:



A Neural Network:



The paper specifically uses the following neural network to predict the sentiment of the Twitter data and News data.

**[Deep Learning Recurrent Neural Network – LSTM – Model Summary]**

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_2 (Embedding) (None, 33, 256) 1280000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 33, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_4 (LSTM) (None, 33, 256) 525312

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_5 (LSTM) (None, 256) 525312

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 3) 771

=================================================================

Total params: 2,331,395

Trainable params: 2,331,395

Non-trainable params: 0

The network produces the following training accuracies after all ten epochs were ran.

**[Deep Learning Recurrent Neural Network – Epochs and Accuracy Score]**

Epoch 1/10

366/366 - 177s - loss: 0.0912 - accuracy: 0.9680

Epoch 2/10

366/366 - 201s - loss: 0.0849 - accuracy: 0.9699

Epoch 3/10

366/366 - 180s - loss: 0.0722 - accuracy: 0.9740

Epoch 4/10

366/366 - 183s - loss: 0.0684 - accuracy: 0.9763

Epoch 5/10

366/366 - 190s - loss: 0.0606 - accuracy: 0.9771

Epoch 6/10

366/366 - 209s - loss: 0.0538 - accuracy: 0.9800

Epoch 7/10

366/366 - 193s - loss: 0.0464 - accuracy: 0.9833

Epoch 8/10

366/366 - 181s - loss: 0.0453 - accuracy: 0.9848

Epoch 9/10

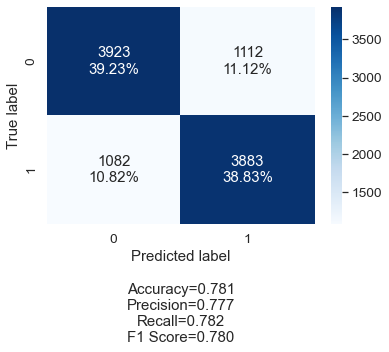
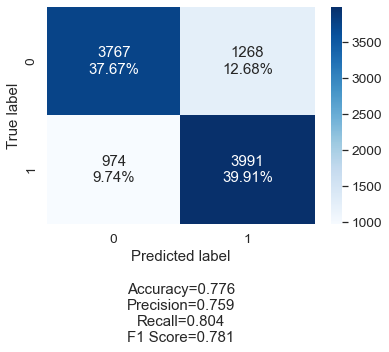
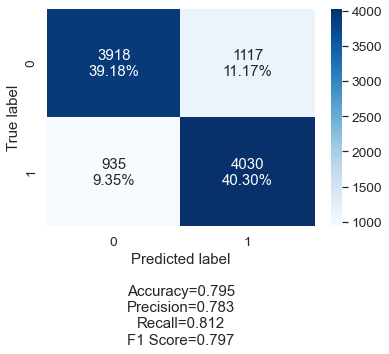
366/366 - 178s - loss: 0.0496 - accuracy: 0.9817

Epoch 10/10

366/366 - 184s - loss: 0.0447 - accuracy: 0.9838

**Results**

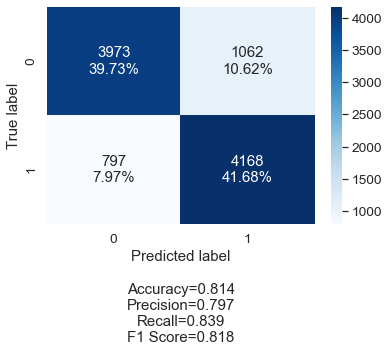
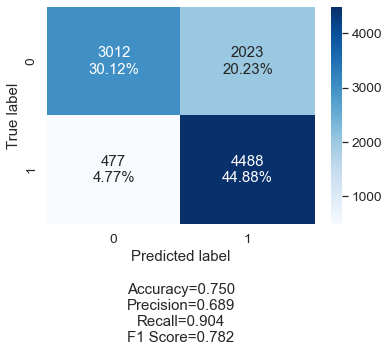
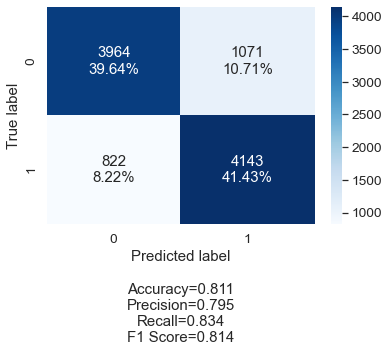
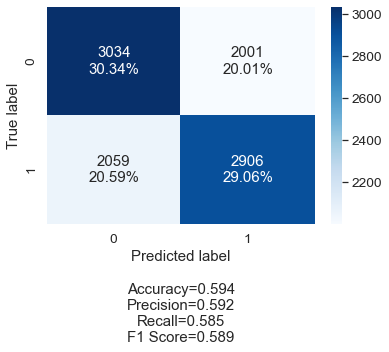
We can now run our Count Vectorizer for the test dataset, and we can see the accuracy from the confusion matrix was approaching a very acceptable F1 score of 78%. The Bernoulli and TDIDF Vectorizers were also run on the data with similar results.

Count Bernoulli Tdidf

In order to further the building of the accuracy of our model, we next utilize the Support Vector Machine (SVM) and we see the outcome for the various kernels used for each SVM.

Linear Polynomial RBF Sigmoid

We can see that Linear and RBF kernels both help to increase the accuracy of the model while the Polynomial kernel stays about the same as and the Sigmoid kernel performs the worst of all by decreasing accuracy nearly 20%. This is curious why the Sigmoid kernel significantly reduces the accuracy, possible due to the nature of that algorithm.

The final step is to run the data through the Linear Dirichlet Allocation (LDA) model. We ran the data for ten different topics including:

**Topic #0: movie film one like book**

**Topic #1: great one best role cast**

**Topic #2: film one story time films**

**Topic #3: show series episode one episodes**

**Topic #4: film war like movie one**

**Topic #5: film one movie like even**

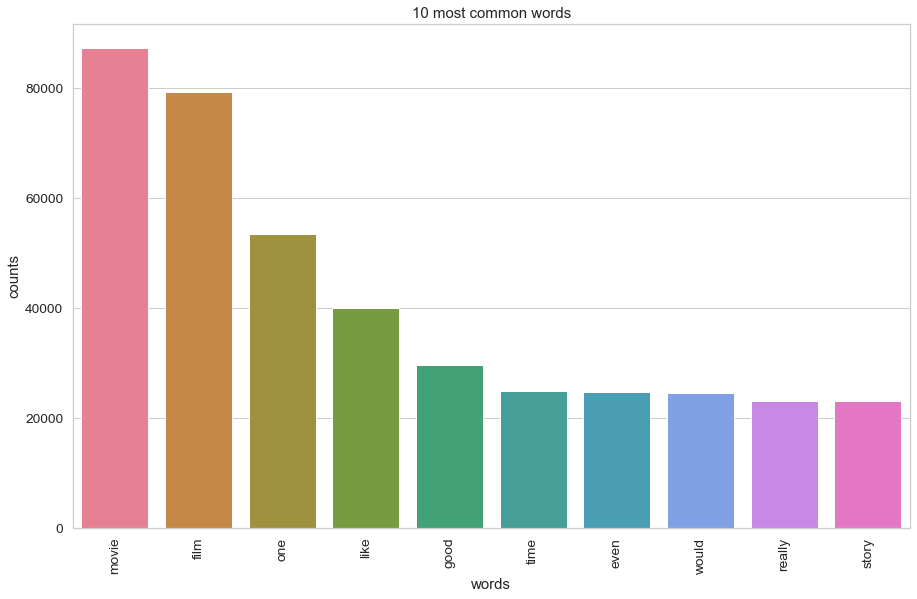
**Topic #6: movie like one good film**

**Topic #7: film movie story one characters**

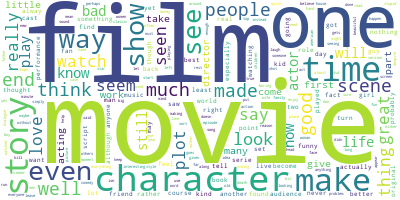
**Topic #8: film one man young two**

**Topic #9: one character film role love**

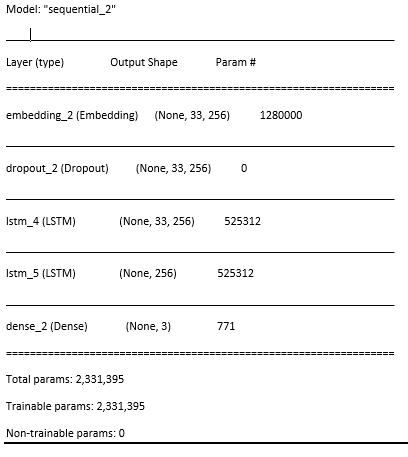
We used this to visualize the results in the following graph of the most popular ten words:



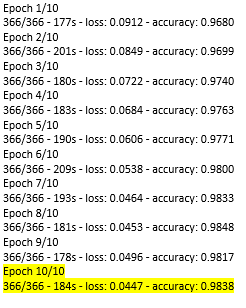
LDA will also allow for other kinds of graphic visualizations as well including word clouds which are very useful in boardroom application where good concise visualizations are needed to quickly explain data to others who might not be as well versed in data science.



To take our project to the next level, we utilized yet another type of model into the analysis of the data. This model is called the Deep Learning Recurrent Neural Network – LSTM. This model runs through a corpus and trains in a continual loop building the accuracy of the model. The parameters for our model are defined like this:

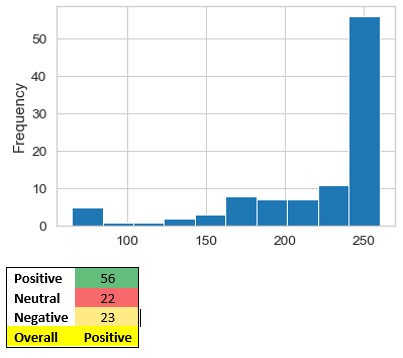
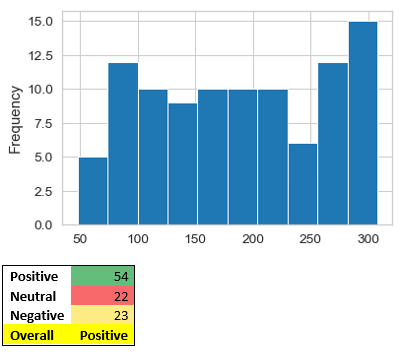


LSTM runs through all ten iterations or epochs of the model building accuracy through each iteration. We can see the progression of the process here:

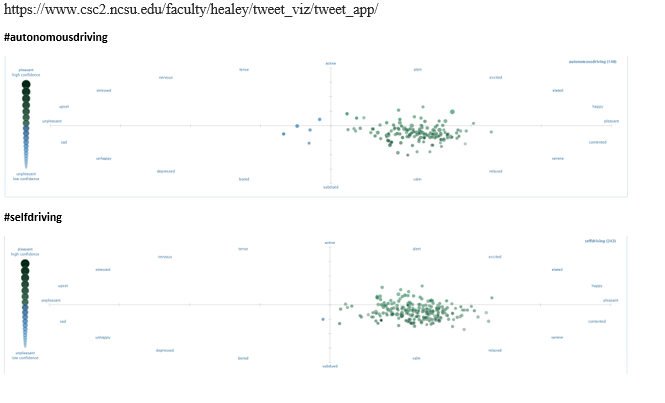


The overall accuracy utilizing the deep learning model is more than impressive when we evaluate the final tenth epoch with an accuracy score of 98.38%. The best thing about this model is that it can be used on any closed corpus.

The overall sentiment of the NewsAPI data and Twitter data can be seen as positive for both datasets. Our research used the machine learning model to test the overall sentiment for both of these datasets.

In order to verify our results from our analysis, we implemented the North Carolina State University website below to verify the results from out model to real-time sentiment. The site allow real-time sentiment analysis of various hashtags, where we input our top two most relevant hashtags, #autonomousdriving and #selfdriving. As shown in the graphs below, the overall sentiment of these hashtags in real-time is again positive. This lends credence to the outcome from out model that it is a viable model.



**Conclusions**

Self-driving cars in general rely heavily on AI and Deep Learning to be able to respond to the many obstacles that they may encounter while on the road. The past decade has been a boon of exploration and advancement in both the areas of AI and DL. Researchers anticipate that the next decade will also be a great time of further advancements based on those of the previous decade.

Yet, autonomous self-driving cars have not become a reality to the optimum potential that is believed they can achieve. This mostly due to the difficulty in navigation of roads with a high degree of accuacy and reliabity that would allow drivers, pedstrians, and property to all be safe.

“Much of the problem is the need for lots of training data. The ideal way to train a self-driving car would be to show it billions of hours of footage of real driving, and use that to teach the computer good driving behavior. Modern machine learning systems do really well when they have abundant data, and very poorly when they have only a little bit of it. But collecting data for self-driving cars is expensive. And since some events are rare — witnessing a car accident ahead, say, or encountering debris on the road — it’s possible for the car to be out of its depth because it has encountered a situation so infrequently in its training data.”(2)

Auto companies will continue to work on this problem until they can solve the many issues that can and will come to pose as unknown situations behind the wheel of a car on any given day. It is difficult to know what the future of autonomous vehicles will look like since there are many issues still to work out and the common consumer is still split on the whether they would even utilize this technology. A recent Gallup pole shows that approximately 50% of Americans polled would not even use the technology.

Regardless, it is a very special and exciting time to be part of the data science world and even more exciting being a data scientist involved in the many advancements that can and will be explored in the very near future. The possibilities are limitless, we will have to build on AI and DL to make the miraculous a definite possibility.

**Resources**

(1) <https://en.wikipedia.org/wiki/Self-driving_car>

<https://www.ucsusa.org/resources/self-driving-cars-101>

<https://www.history.com/.amp/topics/inventions/automobiles>

<https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/>

<https://www.Twitter.com>

<https://www.NewsAPI.org>

(2) <https://www.vox.com/future-perfect/2020/2/14/21063487/self-driving-cars-autonomous-vehicles-waymo-cruise-uber>

**Index**

**i)**

**Levels of Driving Automation**

Level 0: The automated system issues warnings and may momentarily intervene but has no sustained vehicle control.

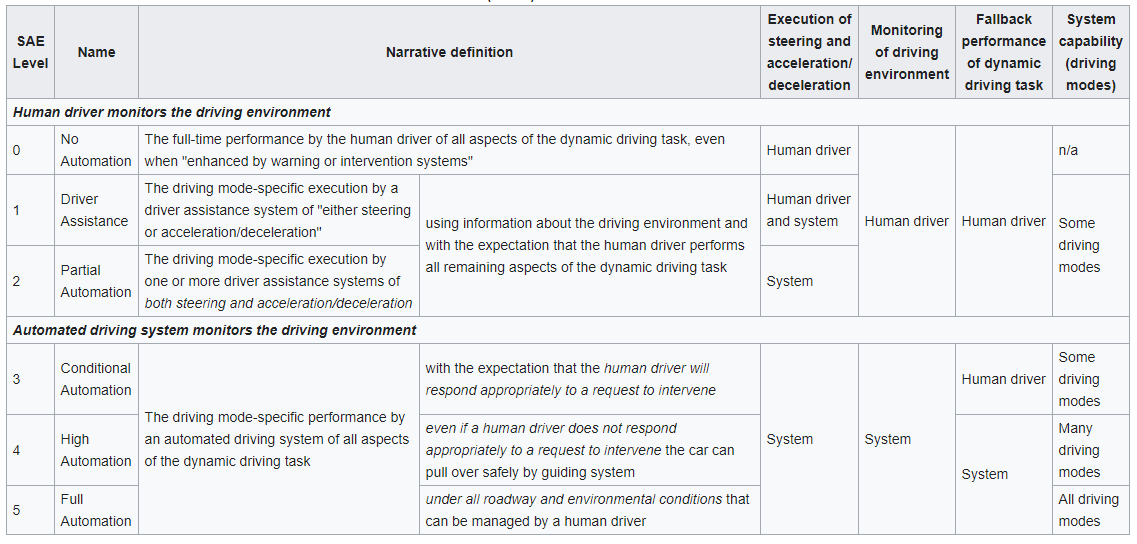
Level 1 ("hands on"): The driver and the automated system share control of the vehicle. Examples are systems where the driver controls steering and the automated system controls engine power to maintain a set speed (Cruise Control) or engine and brake power to maintain and vary speed (Adaptive Cruise Control or ACC); and Parking Assistance, where steering is automated while speed is under manual control. The driver must be ready to retake full control at any time. Lane Keeping Assistance (LKA) Type II is a further example of Level 1 self-driving. A automatic emergency braking which alerts the driver to a crash and permits full braking capacity is also a Level 1 feature, according to Autopilot Review magazine Level 2 ("hands off"): The automated system takes full control of the vehicle: accelerating, braking, and steering. The driver must monitor the driving and be prepared to intervene immediately at any time if the automated system fails to respond properly. The shorthand "hands off" is not meant to be taken literally – contact between hand and wheel is often mandatory during SAE 2 driving, to confirm that the driver is ready to intervene. The eyes of the driver might be monitored by cameras to confirm that the driver is keeping their attention to traffic.

Level 2 ("hands off"): The automated system takes full control of the vehicle: accelerating, braking, and steering. The driver must monitor the driving and be prepared to intervene immediately at any time if the automated system fails to respond properly. The shorthand "hands off" is not meant to be taken literally – contact between hand and wheel is often mandatory during SAE 2 driving, to confirm that the driver is ready to intervene. The eyes of the driver might be monitored by cameras to confirm that the driver is keeping their attention to traffic.

Level 3 ("eyes off"): The driver can safely turn their attention away from the driving tasks, e.g. the driver can text or watch a movie. The vehicle will handle situations that call for an immediate response, like emergency braking. The driver must still be prepared to intervene within some limited time, specified by the manufacturer, when called upon by the vehicle to do so. You can think of the automated system as a co-driver that will alert you in an orderly fashion when it is your turn to drive. An example would be a Traffic Jam Chauffeur,[62] another example would be a car satisfying the international Automated Lane Keeping System (ALKS) regulations.[63]

Level 4 ("mind off"): As level 3, but no driver attention is ever required for safety, e.g. the driver may safely go to sleep or leave the driver's seat. However, self-driving is supported only in limited spatial areas (geofenced) or under special circumstances. Outside of these areas or circumstances, the vehicle must be able to safely abort the trip, e.g. slow down and park the car, if the driver does not retake control. An example would be a robotic taxi or a robotic delivery service that covers selected locations in a specific area.

Level 5 ("steering wheel optional"): No human intervention is required at all. An example would be a robotic vehicle that works on all kinds of surfaces, all over the world, all year around, in all weather conditions.



**ii)**

**Pros vs Cons**

***Pros***

* Environmentally Cleaner
* Potentially Safer – 81% of accidents are due to human error
* Reduced Cost - Maintenance, Insurance.
* Reduce Traffic Congestion
* Faster Travel Speeds

***Cons***

* Cost Prohibitive
* Safety Concerns
* Possible Loss of Jobs – Taxis
* Economic Dependance on Fossil Fuels
* Risk of Hacking