CS4225: Big Data Systems for Data Science

Project : Final Report

***Group* : 23**

Kritartha Ghosh (A0191466X)

Pankaj Bhootra (A0144919W)

Ronald Lim (A0147890U)

Topic : Sentiment Analysis of Twitter Feeds using Apache Spark and MLLib

***Keywords* :** Twitter Feeds, Social Media Data Analysis, Scala, Apache Spark, Natural Language Processing, Data Preprocessing, Classification, Machine Learning with Spark MLLib, Data Management with Spark SQL, Text Analytics, Data Visualization with Tableau, Spark Streaming, Kafka

ABSTRACT

Sentiment Analysis is a well established technique to capture the emotions of people from textual data. It is used in multiple applications such as product reviews, movie reviews, etc. to gauge a better understanding of how people are reacting to the same. We have implemented Sentiment Analysis on Twitter feeds, to understand the reactions of people to certain news, issues or trends at hand. The solution is implemented using techniques in Text Preprocessing and Machine Learning and in order to scale the learning algorithms to train on a large dataset of tweets (approximately 31 million), Big Data technologies such as Spark, MLLib and Spark SQL are used. Multiple Machine Learning models were experimented with and optimized for accuracy using k-fold cross validation and grid search for hyperparameter tuning. The trained models selected are those of Deep Neural Networks and used to predict sentiments on tweets for a given hashtag. This gives us the number of tweets for each sentiment as results, which are stored in a SQL database and further visualized using bar charts in Tableau. While this is our primary use case, another mechanism has been implemented to check how the sentiments for a given hashtag change over time. Our solution is demonstrated for tweets with the hashtags #got and #brexit, one being for Game of Thrones and the other for the Brexit Referendum and the results further validate the opinions expressed in the news about these trends. For Game of Thrones, we also observed a stark difference in sentiments before and after the Season 8 premiere episode aired, giving a visual demonstration of how sentiments change for a given issue over time. While our solution is complete and fully meets our planned objectives, there are future improvements that can be made to improve model performance and provide a 'near' real-time sentiment analysis.

CONTENTS

1. Topic Introduction

Sentiment Analysis is the practice of applying Natural Language Processing and Text Analytics techniques to identify and extract an expressed emotion, or opinion from a given piece of text. Sentiment Analysis is used in multiple applications, like product reviews and movie reviews and it is useful in capturing the "real voice of people" towards various events, issues or services.

For the average consumer, this means that they can find out the opinions that others have towards a product before making a decision to buy it. Businesses are also better able to gauge interest in their products among different demographics and refine the products or their marketing efforts. In politics and governance, leaders can make better or more popular choices depending on the sentiments of the masses. It is clear that sentiment analysis has a myriad of uses in everyday life and exhibits great impact on the world in the age of Big Data.

There are two main techniques for Sentiment Analysis: Machine Learning based and Lexicon based. Our project is based on applying Machine Learning techniques to perform Sentiment Analysis of Twitter feeds. Twitter users commonly associate their tweets with certain hashtags, which describe a certain issue, or trending news at hand. Thus, tweets for a given trend can be easily indexed based on its hashtag, and sentiments can be further examined on the same. But we face many challenges related to volume and veracity of tweets as around 500 million tweets are posted on a daily basis and most of these tweets have text such as usernames and hyperlinks that do not contribute much to the sentiments. Thus, using Big Data tools and through Text Preprocessing and Machine Learning techniques, these challenges have been tackled to perform Sentiment Analysis of Twitter feeds accurately.

2. Previous Work in Sentiment Analysis

Sentiment Analysis, or Opinion Mining, has seen a huge increase in popularity and demand in the recent years. This is mainly due to exponential growth in the amount of data as individuals are now freely able to express their sentiments towards any issue or trend on social media. As such, a few existing applications of Sentiment Analysis and the methods with which they are implemented are discussed here, along with discussion on how these solutions, with their benefits and drawbacks, serve as motivation for our project.

A. Sentiment Analysis using Product Review Data (Fang et al, 2015)

Here, the authors are running Sentiment Analysis on a dataset of 5.1 million product reviews from Amazon, with the products belonging to 4 classes: beauty, books, electronics and home. In order to train their Machine Learning model, they take into account more factors than just the actual review text, such as the reviewer's info, number of upvotes/downvotes on the review, and how many people found the review helpful. They also used a part-of-speech tagging mechanism to classify words of a review into 46 possible tags and are specially tracking negation words such as 'no' and 'not'. They trained Naive Bayes and Random Forest models to predict a sentiment polarity score for each review. While this work uses Machine Learning approaches similar to those implemented in our project, it is predicting a polarity score which is not as useful as classifying a text into discrete sentiments (like anger, anticipation, etc). Also, they have not worked with Big Data technologies such as Spark MLLib which does not provide them scope for parallel computation and is thus less time efficient. This project inspired us to consider a Machine Learning approach instead of a Lexicon based approach for better performance.

B. Exploring Public Sentiments to find Liveable Places (You et al, 2016)

This project proposes a crowd-calibrated, geo-sentiment analysis mechanism to perform three analyses, namely sentiment, clustering and time series analysis on messages exchanged on social media, which are tagged with geographic location. The idea is that geographic location of a person can play a major role in deciding his or her sentiment towards a given issue. This solution is used as part of a bigger project "Liveable Places", to support local authorities, urban designers, and city planners and to help them to better understand public sentiments regarding place redesign in any given location. They can thus better measure people satisfaction, and evaluate the reception of completed facilities in a given area. An extended application is to evaluate the influence of changes or events such as reconstruction so that service users can better understand the reactions of people and thus make better decisions. This solution inspired us to evaluate the change in sentiments on Twitter, for any given issue or trend, as a result of significant events that could strongly affect people reactions.

C. Investor Classification and Sentiment Analysis (Chatterjee et al, 2016)

This project estimates the effects of an investor's bias on the volatility of stocks in the market by performing Sentiment Analysis on the tweets of prominent investors, who continuously share their opinions of various stocks on Twitter. The project assumes that investors can be sentiment driven which could cause stocks to be overrated or underrated. This solution assigns positive, negative and neutral scores to each such tweet, and this allows any novice investor to make informed decisions about which stock to invest in, by closely tracking the sentiments of investors with respect to these stocks. This inspired us to evaluate tweet sentiments for any given hashtag.

D. Forecasting Price Shocks with Social Media Sentiment Analysis (Zhang et al, 2016)

In this project, data from Chinese stock market has been considered along with social media activities to predict price shocks for various stocks. The method involves identifying price shock for a given stock as either negative, near-zero or positive by tracking features such as stock price trends, and social media activities for the stock such as likes, reposts or comments. This solution highlights various social media features that can heavily contribute to a sentiment besides the actual post/text itself and tries to optimize model accuracy by training the same on all these engineered features. While all these features can be extremely useful, it is important to not overfit the model as the actual post/text should still be considered the primary factor.

E. Efficient Adverse Drug Event Extraction using Sentiment Analysis (Peng et al, 2016)

This project identifies adverse drug events, or ADEs, by performing Sentiment Analysis on 4 months of Twitter data. The motivation is that many pharmaceutical companies want to identify ADEs after drugs are released into the market, as several cases may go unidentified. They have worked with technologies such as Apache Spark and Hive to perform data preprocessing and used an existing tool named WEKA for Sentiment Analysis. They managed to capture 20% extra unidentified ADEs as a result of this work.

Overall Verdict: Most of these solutions only attempt to categorise sentiments as positive, negative or neutral, which we feel is not sufficient as we want to get a better variety of sentiments (like fear, anticipation, disgust, joy, etc). Hence, while some of our implemented processes are adapted from previous work, our solution is unique as it can predict a variety of sentiments (10 to be precise, as explained later) on a given set of tweets.

3. Methodology and Experimentation

3.1 Dataset Collection

To perform Sentiment Analysis on Twitter feeds, we need to train Machine Learning models on a large and varied collection of tweets labeled with sentiments to ensure good model performance. We looked at some Twitter statistics and found that on a daily basis, around one million english tweets are posted and so, we decided to collect a month's worth of tweets as this would cover a large variety of topics and give us over 30 million tweets. Our dataset is downloaded from an online archive, which continuously records tweets for every single day and indexes them by month. We managed to download a complete list of tweets posted during the month of October 2018, across multiple languages. We further filtered this massive list to retain only tweets posted in english, so that our models do not get confused due to different languages. We were left with a total archive of **31,130,257** english tweets (around **15 GB**). We stripped this dataset of all unnecessary metadata (such as the user who posted the tweet, the number of replies to the tweet, etc.) and retained only the actual tweet text to bring the dataset size down to **3.01 GB**, which is what we used as our training and testing set. We also labeled each of these tweets with sentiments using an existing word-based sentiment analyzer, which tags sentences with upto 10 sentiments (anger, anticipation, disgust, fear, joy, negative, positive, sadness, surprise, trust) based on what words are found in the sentence. This labelling library is dictionary based and does not use Machine Learning approaches, but it serves our need efficiently by providing a sentiment labeled dataset for model training. Thus, we collected over 31 million english tweets, where each of them were labeled with upto 10 possible sentiments.

3.2 Data Preprocessing

Data preprocessing is an important step before training Machine Learning models. We are using Twitter feeds to train our models so that they can predict sentiments for tweets of a given hashtag. This training is done by providing the classifier with several examples of tweets (input) and their corresponding sentiments (target). Preprocessing is an important step in text mining to modify the raw data into simpler, cleaner representations so that the model can learn better. In our case, preprocessing for tweets is extremely necessary because a typical tweet can contain several noisy details which do not contribute to its sentiment, and so they must be removed so that the model does not get confused. Hence, using regular expressions, a raw tweet is cleaned to remove noise such as hyperlinks, user tags (like @PresidentOfUSA), punctuations, contractions, stopwords and extra blank spaces. Some of the main preprocessing steps are as follows:

1. Replacing Contractions: Dealing with contractions (haven't, isn't, aren't, weren't, etc.) is an important step because when these contractions are expanded, it influences the meaning of all words around it, like adding negation, etc. Not expanding contractions can lead to errors in sentiment prediction. The Machine Learning model can learn to provide different weights to the individual words if they are expanded. If contractions are not broken down, then the contraction and its expanded form would have different weights and the model would not be enriched with proper word constructs. In a research study (Angiani et al, 2016), the authors expand all possible contractions as it adds possible negations to the sentence (haven't becomes *have not*, etc). Our approach is loosely based on the same idea but after expanding the contractions, we are also removing the unnecessary words by removing stopwords, as mentioned next page.

2. Removing Stopwords: These are frequently occurring words that hardly carry any information or orientation (eg. *is*, *are*, *have*, *will*, etc.). An experiment in a previous study found that the traditional sentiment classifier showed an improvement in accuracy from 50% to 58.6% when stopwords were removed (Ghag et al, 2015). We have considered words that occur too frequently and hardly influence the sentiment, like prepositions, common verbs, etc.

3. CamelCase: This is needed in case of hashtags, which may have more than one word joined together, for example #GameOfThrones. For CamelCase, we treat capitalization as indicating the word boundaries. The CamelCase words are split into individual words by identifying these boundaries, since the words individually would add more meaning to the tweet and help to increase the accuracy of sentiments predicted (as opposed to the clubbed word). So #GameOfThrones is split to give "#Game Of Thrones" and the "#" symbol is removed.

Similarly, all the other preprocessing steps are applied using regular expressions to identify and remove text patterns such as hyperlinks, usernames, punctuations, and extra blank spaces. Finally, the tweet is converted to lowercase, so that the same words which come in different cases (eg. *Game* vs. *game*) can be identified as identical terms.

4. Removing empty tweets: This is the last step. Initially, the number of raw tweets from the collected dataset were approximately 31 million but after undergoing the different stages of preprocessing, some of the tweets ended up as empty (which makes sense since we are removing many components like stopwords, punctuations, hyperlinks and usernames). We were left with 15 million preprocessed, non-empty tweets which is what is used for training the model.

3.3 Hashing TF, IDF and Normalization

Having preprocessed our data, we need to choose a suitable text vectorization method to obtain features from our tweets that our Machine Learning models can learn from. We chose one of the most commonly used methods for vectorizing text, namely Term Frequency - Inverse Document Frequency. TF-IDF has been shown to perform sufficiently well in Sentiment Analysis with an accuracy score of 89% on a movie reviews dataset in previous work (Das et al, 2018).

Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF reflects the importance of a term to a document in our entire set of documents, or in our project, the importance of a word to each tweet in our entire dataset of tweets. Term Frequency denotes the number of times that a word appears in a particular tweet, while Document Frequency represents the number of tweets that contains a particular word. Inverse Document Frequency is a numerical measure of how much information is provided by a word. If Term Frequency is used alone without Inverse Document Frequency to measure importance, it is easy to over-emphasise commonly appearing words that hold little information regarding a tweet.

Hashing TF, IDF

Spark MLLib splits the implementation of TF-IDF into two separate components: TF and IDF.

Their version of TF makes use of the hashing trick. In text processing, a “set of terms” might be a bag of words. A word is mapped to an index through the use of the MurmurHash 3 hash function. Term frequencies are then calculated based on the mapped indices. This approach

avoids the need to compute a global word-to-index map as in regular TF which is both computationally and memory expensive. One side effect is that the words become unrecognisable after hashing. This introduces the risk of hash collisions where different words are mapped to the same index. This issue can be easily mitigated by increasing the number of buckets of the hash table through the use of a function setNumFeatures() which Spark MLLib provides for HashingTF. The result is a TF vectorizer which uses less memory and runs faster than a regular implementation of TF. We set the number of features as 10000 in our project.

In Spark, the IDFModel takes our word vectors created from HashingTF and scales each word by down weighting words which frequently appear in our set of documents or tweets. This is to prevent common words which actually hold little information regarding a tweet to have too high of an importance assigned to it.

Normalization

We normalize our feature vector through the use of the StandardScaler available within MLLib. This is a common preprocessing step which can improve model performance. The StandardScaler standardises features by scaling the input vectors to values with zero mean, unit variance and/or removing the mean using column summary statistics. The reason why this step is important is because models like Deep Neural Network expect values to be in a well defined range (such as -1 to 1 or 0 to 1) and when this is not maintained, the model introduces an unfair bias by prioritising the features with higher values, which can negatively affect performance as the model will not learn the patterns (weights) that we expect it to learn.

3.4 Machine Learning using Spark MLLib

In general, there are two classes of machine learning classification problems: MultiClass and MultiLabel. MultiClass classification problems are those with multiple categories or classes in which a single item can be classified. Every item can only be classified to a single category. In MultiLabel classification, every item you try to classify can belong to multiple categories or have multiple labels attached to them. For Twitter Sentiment Analysis, we wanted to assign upto 10 different sentiment labels to each tweet (anger, anticipation, disgust, fear, joy, negative, positive, sadness, surprise, trust), categorizing this problem as a multilabel classification problem.

In another popular non-distributed machine learning library, scikit-learn, multilabel classification can simply be done using the OneVsRest technique where multiple classifier models are trained and each item can be classified with multiple labels. Spark MLLib is a machine learning library using Spark where model training and computation is done in a distributed fashion compared to scikit-learn which trains solely on a single machine. MLLib also has a function which implemented this technique but it did not support multilabel classification and only supported multiclass classification. Thus, we had to write our own implementation of the OneVsRest technique with MLLib. We do so by creating a Pipeline where 10 different models are trained on our feature set. Each model is responsible for classifying if a sentiment is found for a particular tweet and the output is binary: Belong or Do not Belong.

Each line of preprocessed data comes in the following form:

daedalus created wings feathers wax assist sons escape,0,1,0,1,0,0,0,1,1,2

The tweet is followed by the count for each sentiment found in the tweet. We first process this by converting the sentiment count to a binary variable representing sentiment presence. We then pass this data into our pipeline where Hashing TF, IDF, Standard Scaler and our 10 different models are run. For each model we experiment with, we perform Grid Search with K-fold Cross Validation to discover the best hyperparameters to use for each model. Grid Search allows us to test with different hyperparameters while Cross Validation forms multiple train and test sets by randomly assigning each input from the model training data we pass in. Performance is then evaluated on these multiple train and test sets and the results are averaged. If we increase the number of folds used in Cross Validation, we increase the number of randomly formed train and test sets used and the probability that the performance of our trained model is too specific to a particular set of data decreases. When these two techniques are combined we can find the best hyperparameters for the best generalization performance of a model where the model classifies items on previously unseen data.

To find a model with the best performance for our sentiment analysis, we experiment with several different models:

- Deep Neural Network
- Logistic Regression
- Random Forest
- Naive Bayes
- Decision Tree

Deep Neural Network: We implement our Deep Neural Network using MultiLayer Perceptrons. The multilayer perceptron classifier is based on the feedforward artificial neural network and each classifier consists of multiple layers of nodes. Each layer in the network is fully connected to the next one. Nodes in the input layer represent the input data or the number of features that were defined in Hashing TF and nodes in the last layer represent the number of possible outcomes which for us is 2 since we only require a binary outcome for each sentiment.

Logistic Regression: Logistic Regression is a popular method used in predicting categorical responses like animal, pet or in our case the sentiments. It is a special case of Generalized Linear models that predicts outcome probabilities. Since we only require a binary outcome for each model we use Binary Logistic Regression instead of Multinomial Logistic Regression which can classify more than 2 classes.

Decision Tree: Decision Trees are a non-parametric supervised learning method used for classification and regression. The model predicts the value of a target variable by learning simple decision rules inferred from the data features.

Random Forest: Random Forests are ensembles or groups of decision trees. It combines many decision trees in order to reduce the risk of overfitting.

Naive Bayes: Naive Bayes classifiers are a family of simple probabilistic, multiclass classifiers based on the application of Bayes' Theorem with strong (naive) independence assumptions between every pair of features. During prediction, Bayes' theorem is applied to compute the conditional probability distribution of each label given an observation.

3.5 Storing Results using Spark SQL

The Machine Learning models trained, optimized and saved to disk can now be used for predicting sentiments on any given set of tweets to provide useful insights. In this context, two kinds of results are estimated for any given set of tweets. One is predicting the counts of each sentiment across these tweets, done using the trained Machine Learning models. The other is finding the top 150 most frequent words used across these tweets, done using a Word Count approach. These results need to be saved in a format that can be easily queried by visualization tools, and so, the results are stored in an Azure SQL database that can be queried remotely. Writing results to this database is done using Spark SQL, as it provides user friendly APIs to connect and write to remote relational databases. For tweets with a given hashtag, results are written to two tables: "`<hashtag>_sentiments`" table stores the sentiments for each tweet, and "`<hashtag>_wordcounts`" stores the top 150 words found across these tweets.

3.6 Data Visualization using Tableau

Tableau is a powerful data visualization tool to provide interactive visualization products focused on business intelligence. While the full extent of features provided by Tableau is not needed in our project, Tableau still provides us with two convenient facilities: (i) it can easily connect to our remote Azure SQL database where the results are stored, and (ii) it can easily generate bar charts and word clouds, where the former is used to plot the counts of each sentiment across a given set of tweets, and the latter plots the top 150 most frequent words across these tweets, where the more frequent words appear bigger than the rest. These visualizations summarize people reactions by showing what sentiments and words are expressed in the tweets.

4. Results and Discussion

4.1 Model Performance and Comparison

Estimating model performance is done by predicting sentiments on a testing set which was split from the original dataset (20% of the original dataset is kept as testing set). Hence, the models have never seen this test data, and we can run a match of the predicted sentiments against the actual sentiments. We use accuracy as our evaluation metric and the results for each model, which were trained for each of the sentiments, is highlighted in the table below:

Sentiments	Logistic Regression	Naive Bayes	Decision Tree	Random Forest	Deep Neural Network
Anger	93.5 %	81.1 %	77.4 %	71.4 %	**98.6 %**
Anticipation	94.2 %	81.6 %	69.3 %	60.0 %	**98.1 %**
Disgust	94.6 %	80.8 %	84.7 %	79.9 %	**98.3 %**
Fear	93.0 %	82.1 %	76.2 %	71.5 %	**97.6 %**
Joy	94.8 %	81.9 %	72.5 %	63.5 %	**98.5 %**
Sadness	93.8 %	81.3 %	78.7 %	73.4 %	**98.4 %**
Surprise	95.5 %	82.0 %	84.9 %	78.9 %	**98.9 %**
Trust	92.8 %	82.2 %	63.0 %	57.4 %	**98.1 %**
Negative	90.4 %	81.5 %	59.0 %	58.8 %	**96.0 %**
Positive	92.3 %	83.4 %	65.0 %	65.0 %	**98.1 %**

The following table highlights the average accuracy obtained across sentiments for each Machine Learning algorithm:

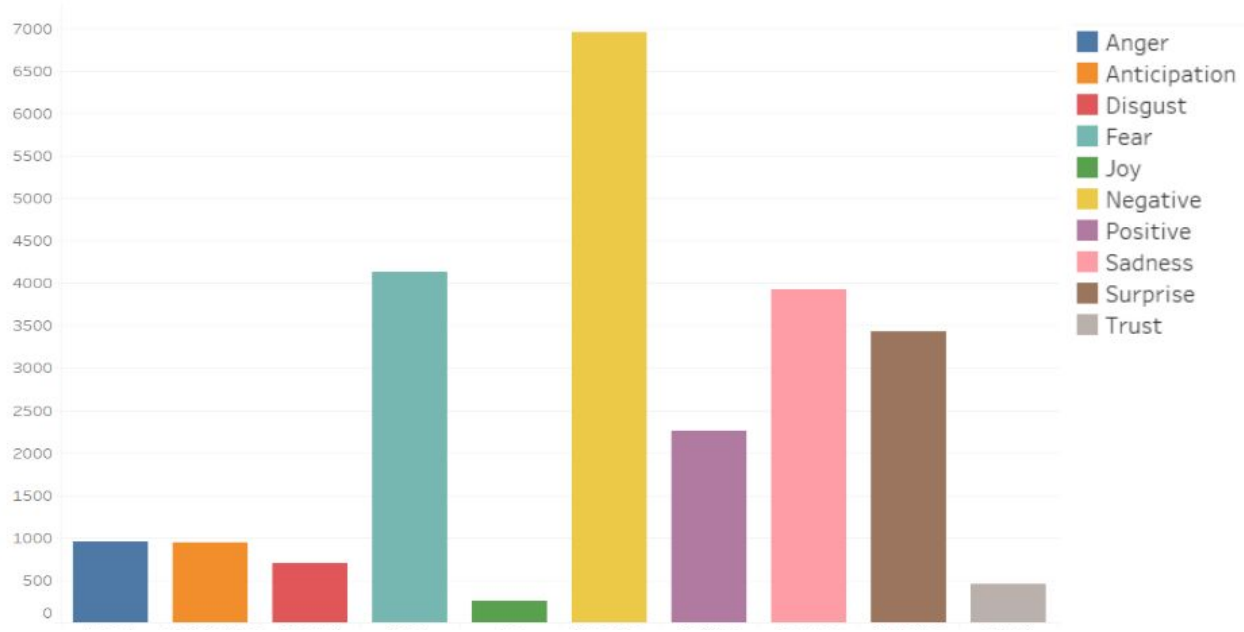
Metric	Logistic Regression	Naive Bayes	Decision Tree	Random Forest	Deep Neural Network
Average Accuracy	93.5 %	81.8 %	73.1 %	68.0 %	**98.1 %**

As inferred from both tables, Logistic Regression (93.5%) and Deep Neural Network (98.1%) provide the best accuracy across all sentiments. These are the only two models whose accuracy scores are able to cross the 90% threshold for all sentiments. However, the performance of our Deep Neural Network still surpasses the Logistic Regression model by almost 5%. This result is expected as Deep Neural Networks using MultiLayer Perceptrons have been shown to perform the best models in previous work (Gupta et al, 2017). Hence, we select Deep Neural Network as the model to use for sentiment analysis of tweets.

4.2 Data Visualization Results

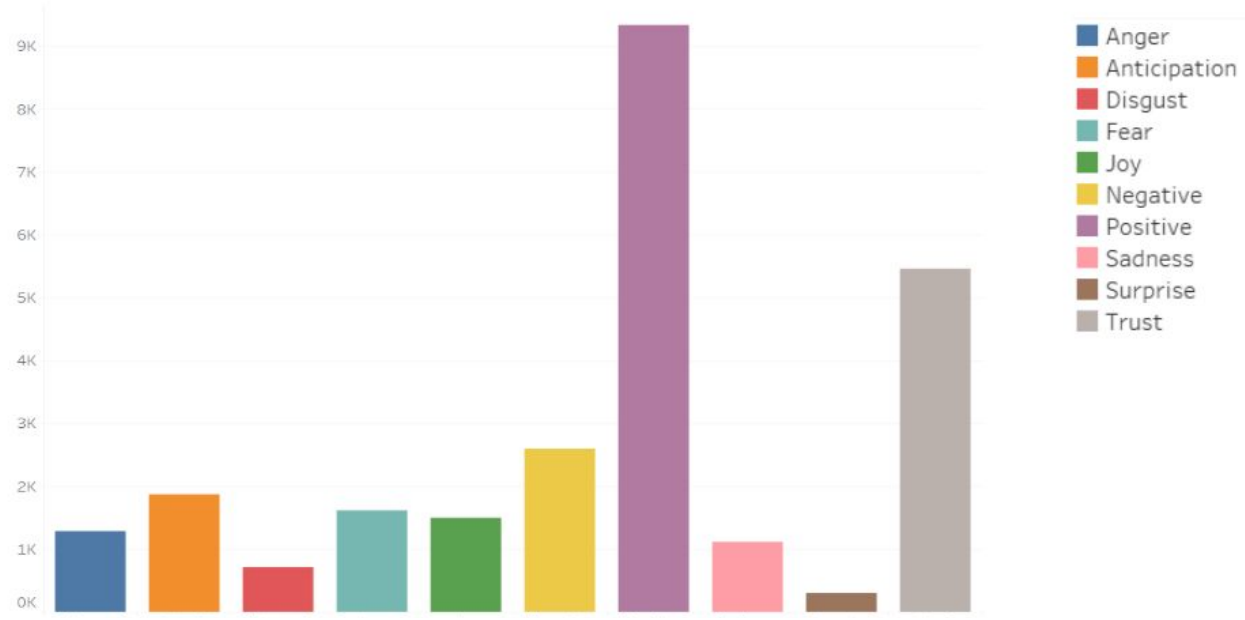
Now that we have trained Deep Neural Network models for each sentiment with high accuracy, they can be easily used for running sentiment predictions on any given set of tweets. To obtain meaningful results, the set of tweets for each test have the same hashtag associated with them. Thus, this allows us to run sentiment analysis for tweets with a given hashtag, and the results are then stored in a database and accessed subsequently for visualization. As mentioned earlier, Spark SQL is used to store the results in an Azure SQL database and Tableau is used to query the database to retrieve the results and perform visualization. For each hashtag, a bar chart showing the counts of various sentiments and a word cloud showing the most frequent words is plotted. The following charts are the results of sentiment analysis for two hashtags: `#brexit` and `#got`.

1. Sentiments for `#brexit` Tweets: showing mostly negative reactions (as expected)

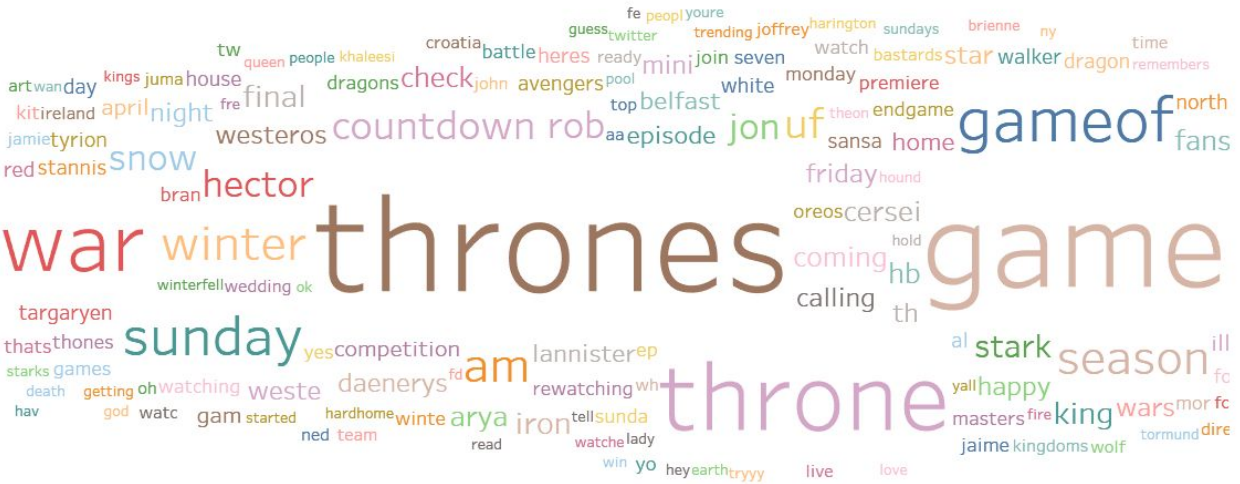


2. Word Cloud for `#brexit` Tweets: showing words like ‘leave’ and ‘breaking’ which justify why the sentiments are mostly negative

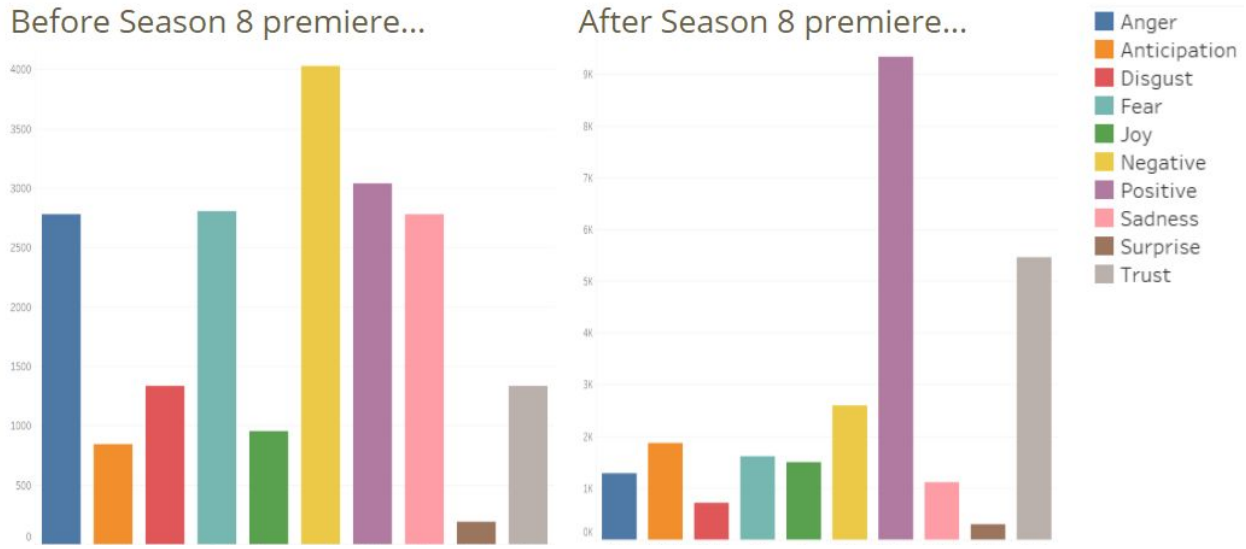
3. Sentiments for `#got` Tweets: showing mostly positive reactions (as expected)



4. Word Cloud for `#got` Tweets: showing words like ‘coming’, ‘calling’ and ‘sunday’ which signify that there is some anticipation for the TV show Game of Thrones (which makes sense as it is a popular show and a new season just started)



5. How sentiments for ‘Game of Thrones’ changed before and after the Season 8 premiere episode (by checking `#got` tweets): This shows that the premiere episode was well received by viewers, with the increase in ‘positive’ and ‘trust’ sentiments.



5. Problems Encountered

There were several key challenges that we encountered at different steps in our methodology, as highlighted below:

1. Difficulty in obtaining a large tweets dataset: Our initial plan was to use Twitter API to download english tweets for a month. However, we later learned that Twitter imposes a rate limit on the API, and we are only allowed to crawl upto 10,000 tweets in every 10 minutes. This severely limited the rate at which we would obtain tweets by self-collection and so, we had to alternatively search for online archives of tweets instead to procure a large dataset. Fortunately, we came across a large archive of tweets, from where we were able to download all tweets for the month of October 2018, giving us a large dataset of **31,130,257** tweets in english.

2. Veracity of tweets: A raw tweet typically consists of several details that do not contribute to its sentiment, such as the presence of usernames, stopwords and hyperlinks. Hence, we had to first examine the raw dataset carefully to identify and eliminate all possible redundancies from each tweet (such as stopwords, usernames, etc.) to improve model performance. This is implemented with regex pattern matching as part of our preprocessing step.

3. Model training: The preprocessing step yields a cleaned list of over 15 million tweets which are used for training the model. Due to the large dataset size, we needed not just Spark MLLib for parallel computation, but also access to large memory hardware so that model training and optimization can be performed more efficiently (especially for Deep Neural Networks). Fortunately, we were given access to an Azure Student Subscription account through which we were able to set up a large memory Spark cluster on Azure Databricks for this purpose.

6. Future Improvements

This project creates a basis with which future projects in Sentiment Analysis of Twitter feeds can be improved. Some possible improvements that build on our solution are as follows:

1. A more extensive preprocessing mechanism: Our solution implemented various text cleaning techniques which are commonly used in Natural Language Processing problems. However, text preprocessing is an area of active research and further techniques could be implemented to preprocess tweets meaningfully, such as part-of-speech tagging for each word in a tweet, and implementing word form reductions like stemming and lemmatization.

2. Extending to other languages: For this project, only tweets written in english were targeted. However, the solution can be extended to other common languages like Mandarin or Spanish, to cover more local news or trends which would be talked about more frequently in the local languages.

3. Using vectorizing techniques such as Word2vec or FastText instead of TF-IDF: TF-IDF is a bag-of-words approach which does not take into account the order of words or the specific context in which these words are used. More recent research has yielded technologies like Word2vec and FastText which could be used instead to provide better word embeddings.

4. Providing a 'near' real-time Sentiment Analysis experience with a streaming mechanism: It is not possible to provide a perfectly real-time Sentiment Analysis of Twitter feeds as the Twitter API currently has a rate limit of around 10000 tweets for every 10 minutes. However, a 'near' real-time experience can still be easily developed by implementing a streaming mechanism using tools such as Kafka or Spark Streaming.

7. Project Summary

To summarise, we are using Big Data technologies such as Spark, MLLib and Spark SQL along with Data Visualization tools such as Tableau to perform Sentiment Analysis on Twitter feeds. Sentiment Analysis is a useful concept that leverages on techniques in Natural Language Processing and Machine Learning to capture the "real voice of people" towards various trending issues, products and events.

Our solution uses Twitter feeds to capture these sentiments, by extracting all the tweets containing a hashtag with respect to a given issue (or an event or even a celebrity/movie/product) and plotting a bar chart showing the counts for various sentiments expressed in these tweets.

Our solution trains a Machine Learning model on millions of labeled tweets which have been preprocessed using techniques such as hyperlink and stopword removal and also vectorized using Hashing TF and IDF.

The sentiments found for each tweet are stored in an Azure SQL database using Spark SQL and visualized using Tableau. Two different visualizations are plotted for each hashtag, where one is a bar chart plotting 10 sentiments against the count for each and the other is a word cloud which shows the top 150 most frequent words across these tweets, thus giving further indication of sentiments expressed. The model performance can be improved with further feature engineering and the solution can be easily extended to support 'near' real-time sentiment analysis.

Hence, this solution successfully gauges the opinions of people, which are used to drive insights that can be useful in decision making for big businesses and also identifying areas of concern that can be improved and enhanced accordingly.

8. Team Member Contributions

We had mostly equal contributions from each team member, more details are as follows:

1. Kritartha Ghosh (A0191466X):

- Performed data preprocessing on the collected tweets.
- Grid search and cross validation implementation for different models.
- Contributed to the project proposal, video, presentation slides and the report.

2. Pankaj Bhootra (A0144919W):

- Collected the large Twitter dataset of over 31 million tweets from an online archive and consolidated it into one CSV for use in our code.
- Wrote the end-to-end script for performing Sentiment Analysis on tweets of a given hashtag, and storing the results in an Azure SQL database using Spark SQL.
- Performed data visualization on the results obtained using Tableau and prepared results for tweets with hashtag #got and #brexit.
- Prepared the slides for the Teaser Talk.
- Contributed to the project proposal, video, presentation slides and the report.

3. Ronald Lim (A0147890U):

- Assessed suitability of Spark MLLib for multilabel classification.
- Implemented OneVsRest technique for multilabel classification.
- Wrote the machine learning script for classifying the sentiments present in tweets.
- Performed model performance evaluation.
- Contributed to the project proposal, video, presentation slides and the report.

9. References

[1] Twitter dataset from: https://archive.org/details/archiveteam-twitter-stream-2018-10

[2] Xing Fang and Justin Zhan (2015). "*Sentiment Analysis using Product Review Data*". Retrieved from https://link.springer.com/article/10.1186/s40537-015-0015-2

[3] Linlin You and Bige Tuncer (2016). "*Exploring public sentiments for liveable places based on a crowd-calibrated sentiment analysis mechanism*". Retrieved from https://ieeexplore.ieee.org/abstract/document/7752312

[4] Arijit Chatterjee and William Perrizo (2016). "*Investor Classification and Sentiment Analysis*". Retrieved from https://dl.acm.org/citation.cfm?id=3192642

[5] Li Zhang, Liang Zhang, Keli Xiao and Qi Liu (2016). "*Forecasting price shocks with social attention and sentiment analysis*". Retrieved from https://doi.ieeecomputersociety.org/10.1109/ASONAM.2016.7752291

[6] Yang Peng, Melody Moh (2016). "*Efficient adverse drug event extraction using Twitter Sentiment Analysis*". Retrieved from https://ieeexplore.ieee.org/document/7752365

[7] Giulio Angiani, Laura Ferrari, Tomaso Fontanini, and Stefano Manicardi (2016). "*A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter*", Universit`a degli Studi di Parma. Retrieved from http://ceur-ws.org/Vol-1748/paper-06.pdf

[8] Ms. Kranti Vithal Ghag, Dr. Ketan Shah (2015). "*Comparative Analysis of Effect of Stopwords Removal on Sentiment Classification.*" Retrieved from https://ieeexplore.ieee.org/document/7375527

[9] Bijoyan Das, Sarit Chakraborty (2018). "*An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation.*" Retrieved from https://arxiv.org/pdf/1806.06407.pdf

[10] Bhumika Gupta, Monika Negi, Kanika Vishwakarma, Goldi Rawat, Priyanka Badhani (2017). "*Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python.*" Retrieved from https://pdfs.semanticscholar.org/c114/7f3d9b46ff0a0c7c43b668123cb15a26120d.pdf