CS4225: Big Data Systems for Data Science

Project Proposal

Group: 23

Pankaj Bhootra (A0144919W)
Kritartha Ghosh (A0191466X)
Ronald Lim (A0147890U)

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

Keywords: Social Media Data Analysis, Spark, Machine Learning using MLLib, Data Visualization, Spark Streaming, Tableau, Kafka, Twitter Feeds, Word Embeddings, Classification, Natural Language Processing, Text Analytics

---

Topic Introduction

Sentiment defines the feelings, emotions and opinions expressed by people in various forms. Sentiment Analysis is the practice of applying Natural Language Processing and Text Analytics techniques to identify and extract an expressed sentiment from a given piece of text. Sentiment Analysis is used in multiple applications, ranging from product reviews to 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.

Sentiment classification can be broken down into three levels: Document level, Sentence level and Aspect or Feature level. At the Document level the entire document is classified to either be positive or negative. Sentence level sentiment classification classifies sentences into positive, negative or neutral classes. At the lowest level, Aspect or Feature level sentiment classification aims to identify and extract product features from the source data.

There are two main techniques for Sentiment Analysis: Machine-Learning based and Lexicon based. Our project will be based on machine learning based techniques.

Previous Work and How they relate to this Topic

[1] is a comparative study where the authors introduce the idea of Sentiment Analysis and discuss the various techniques used. Different datasets and their corresponding techniques are discussed along with their results.

Supervised Learning is mainly used for Machine Learning based Sentiment Analysis techniques. When using machine learning, two sets of documents are needed: training set and testing set. The training set is used by an automatic classifier to learn the differentiating characteristics of the documents and a testing set is used to check how well the classifier performs. In Supervised machine learning, features are extracted from the dataset and labels are attached to every record or review in the dataset. It is with these labels and features that the classifier can learn the differentiating characteristics of the documents.

Common supervised machine learning techniques used in sentiment analysis include Naive Bayes (NB), maximum entropy (ME), and support vector machines (SVM). These methods have all achieved great results with sentiment analysis.

After the collection of data for the training and test datasets, as well as the selection of the appropriate machine learning technique for the problem, the next important step is the selection of features. Features are characteristics which are representative of the documents and with the right selection of features, we can accurately tell how the documents are represented as the classifier would then be able to perform best in labelling documents. If features are incorrectly selected, classifier performance would decrease.

We will next discuss the most commonly used features in sentiment classification.

1. Term presence and their frequency: These features include unigrams or n-grams and their frequency or presence. Pang et al. [2] claim that unigrams gives better results than bigrams in movie review sentiment analysis, but Dave et al. [3] report that bigrams and trigrams give better product-review polarity classification.

2. Part of speech information: In POS tagging each term in sentences will be assigned a label, representing its position or role in the grammatical context. With POS tags, we can identify adjectives and adverbs which are normally used as sentiment indicators [4].

3. Negations: Negations are important because they can reverse a sentiment [5].

4. Opinion words and phrases: Opinion words and phrases express positive or negative sentiments. The main approaches to identify the semantic orientation of an opinion word are statistical-based or lexicon-based.

Additionally, below is a summary of the datasets used along with their techniques and results published in various reseach papers.

Dataset	**Technique (precision, %)**
IMDB	NB (81.5), ME (81.0), SVM (82.9)
Epinions	PMI (66)
Amazon, CNET	SVM (85.8-87.2), NB (81.9-87.0)
Amazon, CNET	Lexicon (84.0)
U.S. & Middle Eastern web forum postings	SVM(95.55)
IMDB, Skytrax, Tripadvisor	Lexicon (86.6)
Luce, Yoka	Lexicon (82.62)
Multi-Domain Sentiment Dataset	ML + Lexicon (66.8)
CNET, IMDB	ML + Lexicon (82.30)

Precision of sentiment analysis using different techniques according to the data reported by authors. [1]

These previous works relate to our project because it gives us an overview of the techniques we can subsequently explore and the possible performance that we might achieve in our goal of sentiment analysis of Twitter data.

Experimental Approach (specific tools, methodologies, experiments)

Our solution for Twitter Sentiment Analysis is divided into two phases: Training Phase to train our Machine Learning model to perform successful sentiment analysis and Inference Phase to use the trained model for real-time sentiment analysis as an application.

For each phase, we will build a separate workflow, and the tasks for each are designed as below:

1. Training Phase

Task 1: Acquiring a dataset of Twitter feeds - Twitter, being an online microblogging tool, acquires millions of messages per day in the form of twitter feeds, that are aimed at almost all industries and issues. A unique feature of the platform is to put various events in trending mode using hashtags, by accounting for the high rate at which people post tweets for such events. This allows us to easily collect tweets pertaining to specific events by filtering using hashtags.
The first task is to collect tweets pertaining to the top 1000 trending hashtags as of current time, which should give us at least a million tweets easily. Data can be extracted by creating a Twitter application which gives us access to their Web API. There are various libraries available in languages such as Scala and Python that use this Web API to download tweets using hashtags as input. Once this dataset is downloaded, it will be tidied up and loaded onto the HDFS storage on the Azure account provided for this project. *Tools used* : Twitter Web API, and scripting languages such as Python or Scala to write a download script.

Task 2: Preprocessing - Once the tweets are collected, they need to be cleaned and labeled before training a classification model. Hence, we will use Spark to apply various preprocessing changes to the raw data by writing multiple MapReduce jobs for tasks such as stopword removal, converting to lowercase, removing punctuations and extra spaces, lemmatizing words, etc. Once the dataset is cleaned, each tweet can then be labeled with a sentiment using an existing sentiment analyzer package, which can be found as libraries in both R and Python. We aim to express upto 10 different sentiments such as happy, anger, joy, positive, neutral, negative, etc. *Tools used* : Spark to clean up each tweet and R or Python libraries to label each tweet with 1 out of 10 possible sentiments.

Task 3: Vectorizing the dataset - Using Spark MapReduce jobs, the cleaned tweets are further tokenized by whitespace to get the words and these words are vectorized using Word Embeddings. The most popular techniques are using Word2vec or FastText models to vectorize each word. The label for each tweet (anger, joy, etc.) can be encoded as a one-hot vector of size 10 (corresponding to 10 classes, or sentiments). *Tools used* : Spark, Word2vec, FastText, One-Hot Encoding.

Task 4: Training the sentiment analysis model (classifier) - The vectorized dataset can now be used for training various Machine Learning Classification models which are available in Spark MLLib package. The library also provides API for performing various Machine Learning tasks such as K-fold cross validation, testing, and evaluation using metrics such as accuracy, precision and F1 Score. *Tools used* : Spark and Spark MLLib.

Task 5: Data Visualization of Test Cases - Once the model has been trained and optimized to an acceptable accuracy, it can be used for real time prediction of sentiment by providing any random tweet (or a collection of tweets for a given hashtag) as input. For instance, one use case would be to visualize the sentiments for a given hashtag as a line graph, where the x axis is the type of sentiment (anger, joy, etc.) and the y axis is the number of tweets corresponding to that sentiment. We can also visualize the tweets by examining a Word Cloud of the same. A Word Cloud takes in a list of tweets as input and shows which words are most commonly used across these tweets, giving us a better idea of the sentiments expressed in those tweets. Both line graphs and word clouds can be generated using visualization tools such as Tableau. *Tools used* : Spark for generating the output in desired format for visualization, Tableau for visualization using Line Graphs and Word Clouds.

2. Inference Phase

Task 1: Implement a Streaming mechanism - To perform real-time sentiment analysis of tweets, we would need a streaming mechanism so that visualization results are progressively updated as the MapReduce jobs are running in the background, hence giving a "streaming update" effect. Various streaming tools and frameworks exist to solve this problem, as presented in the figure below. *Tools used* : Streaming tools such as Spark Streaming and Kafka to input data as streams, a data storage system such as NoSQL or HBase to store and update results on a streaming basis.

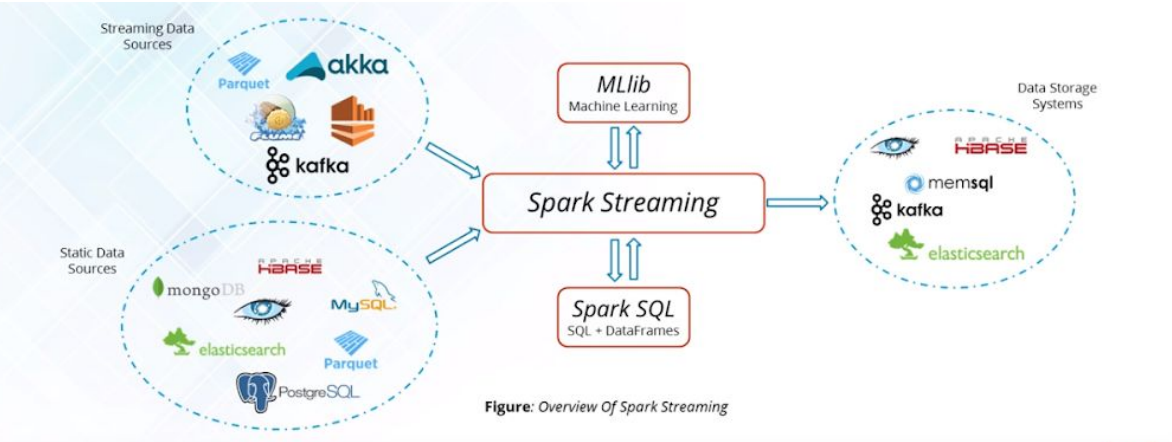


Figure courtesy of Edureka, describing the workflow of Spark Streaming with MLLib

Task 2: Implement an end-to-end pipeline - We also aim to have an end software product that integrates the streaming service, the sentiment analysis service and the data visualization service into one unit which can be triggered by the user for real time sentiment analysis. The aim is to provide an interface where the user enters a hashtag and gets the visualization results showing the sentiments towards that hashtag (event). This can be achieved either using a web application based user interface, or using Tableau's streaming tools to update visualizations based on updated results from the streaming service. *Tools used* : Spark, MLLib, Scala, NoSQL/HBase, Web development frameworks or Tableau streaming tools.

Datasets to be Used

Two major segments of dataset will be needed, one for training the model and the other for testing through the series of tweets that we would collect from Twitter. Tweets are fetched from Twitter's Streaming API, with the assumption that the data would not contain any empty tweets. We would be training our models based on datasets used in research fields for multi class sentiment analysis/emotion analysis. We would be able to rely on these labeled datasets, also we will use third party libraries to label the datasets.

A few key features of a tweet text are as follows:

- Number of words in a tweet
- Number of words followed by hashtags
- Number of words in capital letters - Used to express emotions
- Number of words ending with exclamation mark
- Number of emojis present in the tweet
- Type of punctuation at end - Such as '?' or '!'

In any Natural Language Processing step, cleaning the raw text data is important. Since the model should only get trained using the important characters and words rather than the unwanted ones. Hence, data cleaning needs to be done on the tweets, and following steps would be needed:

- Convert all the words to lowercase,
- Remove the hashtags '#' but not the words followed by it,
- Remove all the punctuations and urls,
- Remove the emojis,
- Remove digits,
- Remove stopwords like "is", "are", "has", "there", etc.,
- Normalize the words to its base form i.e. Say 'Loving', the root word for this is love thus we need to reduce it to its root. This can be accomplished using techniques like stemming, lemmatizing and morphing.

The labeled dataset is used for training and validation purpose.

There are some known issues we could face in classification. With the existing research on sentiment analysis, it has been found that a text containing sarcasm is often misclassified. We would try to address the problem if possible, by looking out for patterns in sarcasm statements vs truth statements and extracting features that differentiate between them. Also, TF-IDF (Term Frequency-Inverse Document Frequency) would be helpful to remove the words that occur many times but do not contribute to predicting the sentiment, these words can be identified by comparing them against the inverse document frequency and filtered out if they are too frequent.

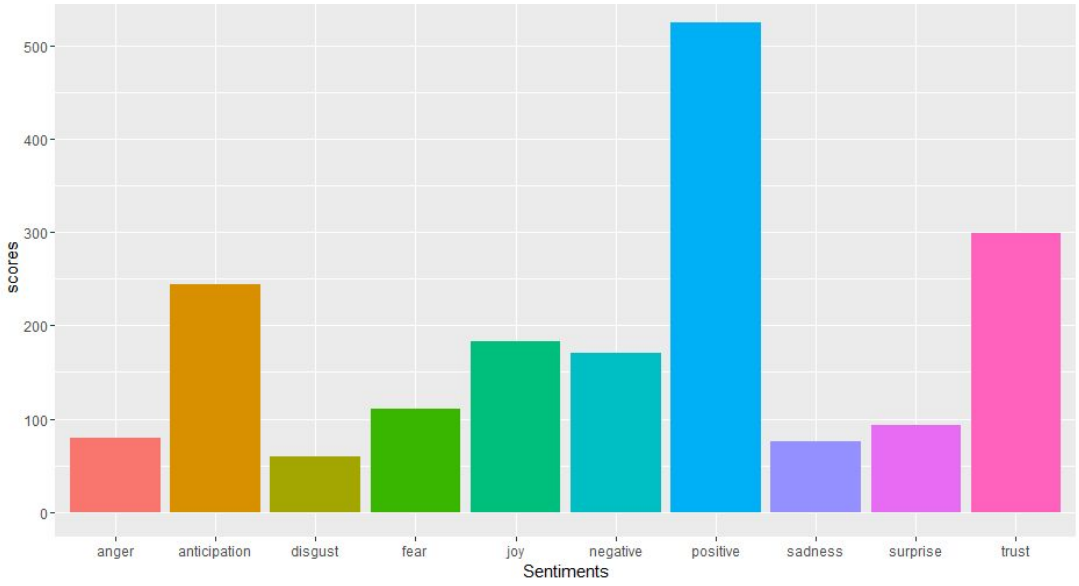
Expected Results

Our goal is to predict the emotions of a stream of tweets. The important words can be plotted into a bar graph against number of times it has been used in the series of tweets. We would also be using **Word Clouds** for showing the most appropriate emotion of the tweet. The word cloud emphasizes more on the word that are frequent and classifies it into one of the ten classes and shows it larger. An example Word Cloud is shown on the right, with most frequent word being “miss”.

The word size is in proportion to its frequency within its sentiment. We can use this visualization to view the most frequently used words across different tweets but the sizes are not comparable across sentiments.

Heatmaps would be used for showing the frequency of each emotion throughout all the tweets, where higher occurrence of the tweets would be visualized by red cells. Example: For around 500 tweets, it could provide frequency (by percentage) metric for each class as follows: Happy: 72%, Sad: 10%, Neutral: 15%, Anger: 3%

The distribution of tweets across sentiments can be shown using **Bar Charts**, highlighting which sentiments are expressed the most in the given set of tweets. An example would be as below:



Furthermore, we can also classify top words across different sources such as tweets from Android, iOS, Web Client, iPad, etc. It would help us to understand the relationship between the source of tweets and the emotions, if any.

Project Summary

To summarise, we are using Big Data technologies such as Spark and MLLib 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 Deep 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 line graph showing the counts for various sentiments expressed in these tweets.

We also discussed various Natural Language Processing and Deep Learning techniques that we are using along with Big Data tools such as Spark to achieve this purpose. Our solution trains a Machine Learning model on millions of labeled tweets which have been preprocessed using techniques such as lemmatizing and stopword removal and also vectorized using pretrained Word2vec and FastText models. Once the model achieves a desired level of accuracy, it is integrated into a real-time streaming service that inputs tweets in batches, predicts sentiments for each and progressively updates the visualizations by updating the stored results.

These visualizations will be generated using Tableau or Web APIs depending on which is more flexible. They will correspond to the results of the Sentiment Analysis performed for tweets of a given hashtag and represent two different metrics. One is a line graph plotting 10 types of sentiments against the count for each (eg. 25% tweets express 'joy', 40% express 'happiness', etc). The other is a Word Cloud (a.k.a. Tag Cloud) which shows what words are used more frequently across these tweets, thus giving further indication of sentiments expressed.

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.

References

1. Vohra, S.M. & Teraiya, J.B.. (2013). A comparative Study of Sentiment Analysis Techniques. Journal JIKRCE. 2. 313-317.

2. B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," Proceedings of the ACL-02 conference on Empirical methods in natural language processing, vol.10, 2002, pp. 79-86.

3. K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," Proceedings of WWW, 2003, pp. 519–528.

4. P. Turney, "Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews", Proceedings of the Association for Computational Linguistics (ACL), 2002, pp. 417–424.

5. B. Pang and L. Lee, "Opinion mining and sentiment analysis," Foundations and Trends in Information Retrieval 2(1-2), 2008, pp. 1–135.

6. Wikipedia (Ret. 2018). Sentiment Analysis. https://en.wikipedia.org/wiki/Sentiment_analysis

7. Wikipedia (Ret. 2018). Word2vec. https://en.wikipedia.org/wiki/Word2vec

8. Wikipedia (Ret. 2018). FastText. https://en.wikipedia.org/wiki/FastText

9. Wikipedia (Ret. 2018). Tag Cloud. https://en.wikipedia.org/wiki/Tag_cloud

10. Twitter (Ret. 2018). About Twitter's APIs. https://help.twitter.com/en/rules-and-policies/twitter-api

11. Dobko (2018). NLP: Text Data Cleaning and Preprocessing. https://medium.com/@dobko_m/nlp-text-data-cleaning-and-preprocessing-ea3ffe0406c1

12. Apache (Ret. 2018). MLLib | Apache Spark. https://spark.apache.org/mllib/

13. DataBricks (Ret. 2018). Spark Streaming with Apache Spark. https://databricks.com/glossary/what-is-spark-streaming

14. Azar (2017). What Is Kafka? https://dzone.com/articles/what-is-kafka