Twitter Sentiment Analysis and Visualization Using Apache Spark and Elasticsearch

Maragatham G¹, ShobanaDevi A²
¹Research Supervisor, ²Research Scholar,
Department of Information & Technology,
SRM University, Chennai, India.
maragathamhaarish@gmail.com
shobanak07@gmail.com

July 25, 2018

Abstract

Sentiment analysis on Twitter data has paying more attention recently. The system's key feature is the immediate communication with other users in an easy, fast way and user-friendly too. Sentiment analysis is the process of identifying and classifying opinions or sentiments expressed in source text. There is a huge volume of data present in the web for internet users and a lot of data is generated per second due to the growth and advancement of web technology. Nowadays, Internet has become best platform to share everyone's opinion, to exchange ideas and to learn online. People are using social network sites like Facebook, Twitter and it has gained more popularity among them to share their views and pass messages about some topics around the world. In the form of tweets, status updates and blog posts, the social media is generating a vast amount of sentiment rich data. This user generated sentiment analysis data is very useful in knowing the opinion of the people crowd. When compared
to general sentiment analysis the twitter sentiment analysis is much difficult due to its slang words and misspellings. Twitter allows 140 as the maximum limit of characters per message. The two strategies that are mainly used for text analysis is knowledge base approach and machine learning approach. In this paper, we analyzed the twitter generated posts using Machine Learning approach. Performing sentiment analysis in a specific domain, is to identify the effect of domain information in sentiment classification. we classified the tweets as positive, negative and extract different peoples' information about that particular domain. In this paper, we developed a novel method for sentiment learning using the Spark coreNLP framework. Our method exploits the hashtags and emoticons inside a tweet, as sentiment labels, and proceeds to a classification procedure of diverse sentiment types in a parallel and distributed manner.

1 Introduction

Nowadays, people be likely to disseminate information, using short 140-character messages called "tweets", for various aspects on Twitter. Additionally, they follow other users in sequence to receive their status updates on tweets. In nature, Twitter has a wide distribution instant messaging platform and people are using that to get informed about world news, current scientific advancements, etc. Unavoidably, a variety of view clusters that includes a wealthy sentiment information is formed. Sentiment is termed as "A way of expressing one’s thought, view, or attitude, particularly based mainly on emotion instead of reason" and it also describes someone’s mood or critic towards a particular entity or domain[3, 5].

The information about overall sentiment tendency towards a specific topic may be used enormously in certain cases. For representation, assume a mechanical organization would be intrigued to think about their client’s perspectives about the most recent item, with a specific end goal to get accommodating criticism that will use in the creation of the next device[6, 7].

In this way, clearly a comprehensive feeling examination for a day and age after the arrival of the new item is required. In addition [4, 6], client produced content that catches opinion data has
turned out to be important among numerous web applications and
data frameworks, for example, web crawlers or proposal systems.
In the setting of this work, we use hash tags and emojis as sup-
position marks to perform characterization of different slant writes
[23]. Hash tags are a tradition for including extra setting and meta-
data and are widely used in tweets. Their use is twofold: they give
arrangement of a message as well as feature of a point and they en-
hance the seeking of tweets that allude to a typical subject. A hash
tag is made by prefixing a word with a hash symbol (e.g. love).
Emoji alludes to a computerized symbol or an arrangement of con-
sole images that serves to speak to an outward appearance, as :-
( for a pitiful face. Both, hashtags and emojis, give a fine-grained
supposition learning at tweet level which makes them reasonable to
be utilized for assessment mining.

The issue of opinion investigation has been contemplated widely
amid late years. The greater part of existing arrangements is lim-
ited in brought together conditions and base on characteristic di-
ialect handling strategies and machine learning approaches. Be that
as it may, this sort of strategies are tedious and computationally
serious [16, 22]. Thus, it is restrictive to process in excess of a cou-
ple of thousand tweets without surpassing the abilities of a solitary
server. Unexpectedly, a large number of tweets are distributed day
by day on Twitter. Subsequently, underline arrangements are nei-
ther adequate nor appropriate for conclusion mining, since there
is an enormous bungle between their preparing abilities and the
exponential development of accessible information [16]. It is more
than clear that there is a basic need to swing to high adaptable ar-
rangements. Distributed computing innovations give instruments
and foundation to make such arrangements and deal with the in-
formation distributed among various servers. The most conspicuous
and strikingly effective instrument is the MapReduce programming
model [7], created by Google, for handling extensive scale informa-
tion. [13, 21]
2 Preliminaries

2.1 Previous Work

In spite of the fact that the idea of notion examination, or supposition mining, is generally new, the exploration around this area is very broad. Early examinations center around report level supposition investigation concerning motion picture or item audits [11, 30] and posts distributed on site pages or online journals [29]. Because of the multifaceted nature of record level supposition mining, numerous endeavors have been made towards the sentence level sentiment analysis. The arrangements exhibited in inspect expressions and dole out to every last one of them a conclusion extremity (positive, negative, nonpartisan) [25, 26, 28]. A less researched zone is the theme based assessment examination [15, 17] because of the trouble to give a satisfactory meaning of point and how to consolidate the slant factor into the conclusion mining undertaking. The most widely recognized ways to deal with go up against the issue of sentiment examination incorporate machine learning as well as characteristic dialect preparing procedures. In, the creators utilize [20] Naive Bayes, Maximum Entropy and Support Vector Machines to characterize film audits as positive or negative, and perform a correlation between the techniques as far as order execution. Then again, Nasukawa and Yi [18] endeavor to distinguish semantic connections between the estimation articulations and the subject. Together with a syntactic parser and an opinion vocabulary their approach figures out how to expand the precision of assumption investigation within web pages and online articles. Besides, Ding and Liu [8] characterize an arrangement of semantic principles together with another conclusion total capacity to distinguish slant introductions in online item surveys. Amid the most recent five years, Twitter has gotten much consideration for assumption investigation. In [2], the creators continue to a 2-step order process. In the initial step, they isolate messages as subjective and objective and in the second step they recognize the subjective tweets as positive or negative. Davidov et al. [6] assess the commitment of various highlights (e.g. n-grams) together with a kNN classifier. They exploit the hashtags and smileys in tweets to characterize assumption classes and to keep away from
manual explanation. In this paper, we embrace this approach and incredibly extend it to help the investigation of substantial scale Twitter information. Agarwal et al. [1] examine the utilization of a tree piece model for distinguishing opinion introduction in tweets. A three-advance classifier is proposed in [12] that takes after an objective ward estimation order system. Besides, a diagram based model is proposed in [23] to perform assessment mining in Twitter information from a point based viewpoint. A later approach [27], fabricates a supposition and emoji vocabulary to help multi-dimensional slant investigation of Twitter information. A vast scale arrangement is introduced in [14] where the creators assemble an opinion vocabulary and characterize tweets utilizing a MapReduce calculation and a disseminated database show. In spite of the fact that the precision of the technique is great, it experiences the tedious development of the supposition vocabulary. Our approach is substantially less complex and completely misuses the abilities of Spark system. To our best learning, we are the first to introduce a Spark-based extensive scale approach for conclusion mining on Twitter information without the need of building an assumption vocabulary or continuing to any manual information explanation.

2.2 Spark Framework

Apache Spark [13, 21] is a quick and general motor for extensive scale information handling. Basically, it is the advancement of Hadoop [10, 24] structure. Hadoop is the open source execution of the MapReduce demonstrate and is broadly utilized for conveyed preparing among various servers. It is perfect for cluster based procedures when we have to experience all information. Be that as it may, its execution drops quickly for certain issue writes (e.g. when we need to manage iterative or chart based calculations). Spark is a brought together pile of different firmly coordinated segments and conquers the issues of Hadoop. It has a Directed Acyclic Graph (DAG) execution motor that backings cyclic information stream and in-memory registering. Subsequently, it can run programs up to 100x quicker than Hadoop in memory, or 10x speedier on plate.

Start incorporates a heap of libraries that consolidate SQL, gushing, machine learning and diagram handling in a solitary motor. Start offers some abnormal state systems, for example, reserving and
makes simple to construct circulated applications in Java, Python, Scala and R. The applications are converted into MapReduce employments and keep running in parallel. Besides, Spark can get to various information sources, for example, HDFS or HBase [30].

3 Sentiment Analysis Framework

3.1 Spark CoreNLP

Our pipeline system was initially designed for internal use. Previously, when combining multiple natural language analysis components, each with their own ad hoc APIs, we had tied them together with custom glue code. The resulting Annotation, containing all the analysis information added by the Annotators, can be output in XML or plain text forms. annotation pipeline was developed in 2006 in order to replace this jumble with something better. A uniform interface was provided for an Annotator that adds some kind of analysis information to some text. An Annotator does this by taking in an Annotation object to which it can add extra information. An Annotation is stored as a type safe heterogeneous map, following the ideas for this data type presented by Bloch (2008). This basic architecture has proven quite successful, and is still the basis of the system described here.

The inspirations were:

- To have the capacity to rapidly and effortlessly get semantic comments for a content.
- To shroud varieties crosswise over segments behind a typical API.
- To have a negligible theoretical impression, so the framework is anything but difficult to learn.
- To give a lightweight system, utilizing plain Java objects (as opposed to something of heavier weight, for example, XML or UIMA’s CommonAnalysis System (CAS) objects).

In 2009, at first as a feature of a multi-site grant project, the framework was stretched out to be all the more effortlessly usable
by a more extensive scope of clients. We provided a command line interface and the capacity to write out an Annotation in different arrangements, including XML. Additionally work prompted the framework being released as free open source programming in 2010. From one perspective, from a compositional point of view, Stanford CoreNLP does not endeavor to do everything. It is simply a straightforward pipeline engineering. It gives just a Java API. It does not endeavor to give various machine scale-out (however it provides multi-threaded processing on a solitary machine). It gives a simple concrete API. In any case, these prerequisites satisfy a substantial level of potential clients, and the resulting simplicity makes it less demanding for clients to get started with the system. That is, the essential preferred standpoint of Stanford CoreNLP over bigger systems like UIMA (Ferrucci and Lally, 2004) or GATE (Cunningham et al., 2002) is that clients do not need to learn UIMA or GATE before they can get began; they just need to know a little Java [25, 28].

By and by, this is an extensive and essential differentiator. If more mind-boggling situations are required, for example, various machine scale-out, they can regularly be accomplished by running the investigation pipeline inside a framework that spotlights on appropriated workflows such as Hadoop or Spark. Different frameworks endeavor to give all the more, for example, the UIUC Curator (Clarke et al., 2012), which incorporates bury machine customer server correspondence for preparing and the storing of normal dialect examinations. Yet, this usefulness includes some significant downsides. The framework is unpredictable to introduce and complex to get it. In addition, by and by, an association may well be focused on a scale-out arrangement which is not the same as that gave by the characteristic dialect examination toolbox. For instance, they might utilize Kryo [30] or Google’s proto yet for double serialization as opposed to Apache Thrift which underlies Curator. For this situation, the client is ideally serviced by a genuinely little and independent common dialect examination framework, instead of something which accompanies a great deal of things for a wide range of purposes, the greater part of which they are not utilizing.

Then again, most clients advantage enormously from the arrangement of an arrangement of steady, powerful, high. All things considered, it can call an investigation segment written in different
dialects through a fitting wrapper Annotator, and thus, it has been wrapped by numerous individuals to give Stanford CoreNLP ties to different dialects. Quality semantic investigation parts, which can be effortlessly summoned for normal situations. While the developer of a bigger framework may have settled on general plan decisions, for example, how to deal with scale out, they are probably not going to be a NLP master, and are henceforth searching for NLP segments that simply work. This is a colossal preferred standpoint that Stanford CoreNLP and GATE have over the vacant tool compartment of an Apache UIMA download, something tended to some extent by the advancement of all around incorporated part bundles for UIMA, for example, ClearTK (Bethard et al., 2014) [29], DKPro Core(Gurevych et al., 2007), and JCoRe (Hahn et al., 2008). In any case, the arrangement gave by these bundles stays harder to learn, more intricate and heavier weight for clients than the pipeline portrayed here.

Practically speaking, this is a huge and essential differentiator. In the event that more unpredictable situations are required, for example, different machine scale-out, they can ordinarily be accomplished by running the examination pipeline inside a framework that spotlights on dispersed workflows(such as Hadoop or Spark). Different frameworks endeavor to give all the more, for example, the UIUC Curator(Clarke et al., 2012), which incorporates bury machine customer server correspondence for preparing and the storing of characteristic dialect examinations. In any case, this usefulness includes some major disadvantages. The framework is perplexing to introduce and complex to get it. In addition, practically speaking, an association may well be focused on a scale-out arrangement which is unique in relation to that gave by the normal dialect investigation toolbox. For instance [9, 10], they might utilize Kryo or Google’s proto yet for paired serialization as opposed to Apache Thrift which underlies Curator.

For this situation, the client is ideally serviced by a genuinely little and independent regular dialect investigation framework, instead of something which accompanies a considerable measure of stuff for a wide range of purposes, the greater part of which they are not utilizing. Then again, most clients advantage extraordinarily from the arrangement of an arrangement of steady, powerful, high. In any case, it can call an examination part written in different di-
alects by means of a suitable wrapper Annotator, and thusly, it has been wrapped by numerous individuals to give Stanford CoreNLP ties to different dialects. Quality semantic investigation segments, which can be effortlessly conjured for regular situations. While the manufacturer of a bigger framework may have settled on general outline decisions, for example, how to deal with scale out, they are probably not going to be a NLP master, and are consequently searching for NLP segments that simply work. This is an enormous favorable position that Stanford CoreNLP and GATE have over the void tool compartment of an Apache UIMA download, something tended to a limited extent by the advancement of all around incorporated part bundles for UIMA, for example, ClearTK (Bethard et al., 2014), DKPro Core(Gurevych et al., 2007) [21, 23], and JCoRe (Hahn et al.,2008). Be that as it may, the arrangement gave by these bundles stays harder to learn, more unpredictable and heavier weight for clients than the pipeline portrayed here. The framework comes bundled with models for English. Isolate display bundles offer help for Chinese and for the case-obtuse handling of English. Support for different dialects is less total, yet a considerable lot of the Annotators likewise bolster models for French, German, and Arabic (see supplement B), and building models for facilitating dialects is conceivable utilizing the hidden instruments. In this area, we plot the gave annotators, concentrating on the English variants. It ought to be noticed that a portion of the model’s basic annotators are prepared from explained corpora utilizing directed machine learning, while others are lead-based parts, which all things considered frequently require some dialect assets of their own. tokenize [2, 15] Tokenizes the content into an arrangement of tokens. The English segment gives a PTB style tokenizer, stretched out to sensibly deal with boisterous and web content. The relating segments for Chinese and Arabic give word what’s more, clitic division. The tokenizer spares the character balances of every token in the information content. Clean XML Removes most or all XML labels from the records split Splits a grouping of tokens into sentences genuine case Determines the imaginable genuine instance of tokens in content (that is, their conceivable case in very much altered content), where this data was lost, e.g., for all capitalized content. This is actualized with a discriminative model utilizing a CRF arrangement tagger (Finkel et al., 2005) [19, 22].
3.2 Elasticsearch

Elasticsearch is an Apache Lucene-based search server. It was created by Shay Banon and distributed in 2010. It is currently kept up by Elasticsearch BV. Its most recent variant is 2.1.0. Elasticsearch is a continuous, conveyed, and open-source full-content search and examination motor. It is available from RESTful web service interface and utilizes patternless JSON (JavaScript Object Notation) reports to store information. It is based on Java programming dialect, which empowers Elasticsearch to keep running on various stages. It empowers clients to investigate extensive measure of information at rapid. The general highlights of Elasticsearch will be, Elasticsearch is versatile up to petabytes of organized and unstructured information. Elasticsearch can be utilized as a substitution of archive stores like MongoDB and RavenDB. Elasticsearch utilizes denormalization to enhance the search execution. Elasticsearch is one of the well-known enterprise search engines, which is presently being utilized by numerous enormous associations like Wikipedia, The Guardian, Stack Overflow, GitHub, and so on. Elasticsearch is open source and accessible under the Apache permit version 2.0.

3.2.1 Elasticsearch Key Concepts

The key ideas of Elasticsearch are as per the following:

- **Node**: It alludes to a solitary running occurrence of Elasticsearch. Single physical and virtual server obliges various hubs relying on the abilities of their physical assets like RAM, Stack, and processing power.

- **Cluster**: It is an accumulation of at least one hubs. Cluster gives aggregate ordering what’s more, look capacities over every one of the hubs for whole information.

- **Index**: It is an accumulation of various kind of reports and record properties. File additionally utilizes the idea of shards to enhance the execution. For instance, a set of report contains information of a person to person communication application.

- **Type/Mapping**: It is an accumulation of records sharing an arrangement of normal fields introduce in a similar record.
For instance, an Index contains information of a social organizing application, and afterward there can be a particular kind for client profile information, another write for informing information and another for remarks information.

- **Document**: It is an accumulation of fields in a particular way characterized in JSON design. Each record has a place with a sort and lives inside a file. Each archive is related with a remarkable identifier, called the UID.

- **Shard**: Indexes are evenly subdivided into shards. This implies every shard contains every one of the properties of record, yet contains less number of JSON objects than list. The flat partition makes shard an autonomous hub, which can be store in any hub. Essential shard is the first even piece of a list and after that these essential shards are recreated into reproduction shards.

- **Replicas**: Elasticsearch enables a client to make copies of their lists and shards. Replication not just aides in expanding the accessibility of information if there should be an occurrence of disappointment, yet additionally enhances the execution of seeking via completing a parallel hunt activity in these reproductions.

### 3.2.2 Elasticsearch Advantages

- Elasticsearch is created on Java, which makes it good on relatively every stage.

- Elasticsearch is ongoing, as it were following one moment the additional archive is accessible in this motor.

- Elasticsearch is appropriated, which makes it simple to scale and incorporate in any enormous association.

- Creating full reinforcements are simple by utilizing the idea of the entryway, which is available in flexible hunt.

- Handling multi-tenure is simple in Elasticsearch when contrasted with Apache Solr.
• Elasticsearch utilizes JSON questions as reactions, which makes it conceivable to summon the versatile hunt server with countless programming dialects.

• Elasticsearch underpins relatively every report write with the exception of those that don’t bolster content rendering.

4 Proposed framework

In the first place, we got the labels from the tweets, check how often it (a tag) shows up and sort them by the tally. From that point forward, we hold on the outcome to point Splunk (or some other apparatus for this issue) to it. We could fabricate some intriguing dashboards utilizing this data so we can track the most slanting hashtags. In view of this data my associate could make battles and utilize these prominent labels to draw in a greater group of onlookers. In the wake of trying different things with various applications to process spilling information like Spark streaming, flume, kafka, storm and so on gives now a chance to take a gander at how assumption scores can be created for tweets using Spark stanford CoreNLP and assemble representation dashboards on this information utilizing elasticsearch and kibana.

Figure 1. The Proposed Framework of Sentiment Analysis
5 Implementation

This work is implemented using ApachSpark. First the tweets are collected from the twitter’s official website. Once we register in twitter site, the latest tweets of about size 1GB can be downloaded. The twitter site will generate security keys and oAuth tokens for every user. It is used mainly for coding in Spark.

Next step is to create a scala maven project and the corresponding pom.xml file is updated with the dependencies that are required for this work. A scala object file was created, to receive the streaming data that are collected from the twitter. Using the twitter data, the sentiment scores are detected on each tweet by importing the package stanford coreNLP library. Once the analysis has been done, the output can be visualized by creating an index in elasticsearch and the output is written in that index. The index created in this paper is named as twitter_092517/tweet. Elasticsearch basically requires index content that can be translated into a document. Before storing the content in the created index (twitter_092517/tweet), each RDD in Spark is transformed to a Map object. The stanford university provided a useful natural language processing library coreNLP in Spark, to parse and detect the sentiments of each tweet data.

Stanford coreNLP gives a device pipeline as far as annotators utilizing which distinctive phonetic investigation instruments might be connected on content. Following annotators are incorporated into this case:

- **tokenize** - Divides content into an arrangement of words
- **-** - Split the content into sentence. Distinguish fullstop, outcry and so forth and split sentences
- **POS**- Reads message and appoints parts of discourse to each word, for example, thing, verb, descriptive word, and so on. Ex. "This is a basic sentence" will be labeled as "This/DT is/VBZ a/DT test/NN sentence/NN"
- **lemma** - Group together types of a word so they can be examined as a solitary thing.
- **parse** - Provides syntactic investigation
- sentiment - Provides show for assumption investigation. Joins a binarized tree of the sentence. The hubs of the tree at that point contain the comments from RNNCoreAnnotations showing the anticipated class and scores for that subtree. The slant estimations of the individual words are accumulated at the base of the binarized tree.

Sentiment score is then found the middle value of in view of length of each sentence as longer sentence must convey more weight in the general opinion of the content.

6 Results and Discussion

Once the implementation is done, the results can be viewed visually by using Elastic search and Kibana. Using the index created twitter_092517/tweet before, the contents of output are transformed to ElasticSearch. From the total number of 4,315 tweets data, the Spark calculates sentiment score for each tweet data and based on the score it is classified as either positive, negative neutral, not understanding and very negative categories. In this given tweet data 4,315 tweets are classified into 3,249 of negative, 138 of positive, 847 of neutral, 75 of not understood and 8 of very negative categories.

Figure 2. Elastic search view of results based on the created index twitter_092517

The results are displayed below with total counts of tweet data for analysis and the list of classified tweets based one sentiments and the quarter hourly analysis of classified tweets and the pie chart view of classified tweets and the trending Hash tags list and the graphical view of text based sentiment classification. Thus this
work of sentiment analysis using Spark coreNLP gives a clear visualization of classified tweets data and more accurate results with less time consuming.

Figure 3. The Total Tweets count for Analysis

Figure 4. List of classified Tweets based on Sentiments
Figure 5. Quarter Hourly Analysis of Classified Tweets

Figure 6. The PIE Chart View of Classified Tweets

Figure 7. Trending Hashtags
7 Conclusion and Future work

In the context of this work, we presented a novel technique for sentiment learning in the Spark framework and visualizing the results using Elasticsearch and Kibana. The proposed Stanford natural language processing library coreNLP is very efficient and most useful to process the text using the NLP functions and it helps us to classify the text based on sentiment scores. Since, this classification used a faster distributed and parallel computing engine framework Spark, the performance is much better compared to other works that are discussed before in this paper. The visualization frameworks Elasticsearch and Kibana are used to extract the output from Spark and it is visualized in different formats with the created index. So, the users can easily understood the results and can identify which sentiment (positive, negative and neutral) has received more tweets on that particular topic or domain. It also gives the most trending hashtags of that topic. In the near future, this work can be still extended for sentiment analysis with massive data in terms of Terabytes/Petabytes and can be distributed among multiple nodes of cluster to achieve less time consuming and more accurate results for large data sets. In addition to this, the tweets data can also be classified for multiple domains and the results can be visualized by comparing the different topics or domains. This would help in future to get more and efficient knowledge from the views that are posted by various kinds of people using this twitter application.
References


