Sentiment Analysis on Streaming Data: Case Study on Twitter

Indah Permatasari	Dading Zainul Gusti	Andre	Apri Kamayudi Kasiran
929578	1001261	920327	1009534
ipermatasar@student.unimelb.edu.au	dgusti@student.unimelb.edu.au	andre1@student.unimelb.edu.au	akasiran@student.unimelb.edu.au

Abstract

Sentiment analysis on social media especially Twitter has been a popular demand among businesses as this offer a new resolution of early warning system to detect negative or positive review of particular products. The need of this early detection makes analysis on streaming data becomes even more required. There are several platforms to support near real time analysis on streaming data such as Hadoop MapReduce, Apache Spark, and Apache Storm. Each of this platform has its own predominance aspects. Similarly, classifying a tweet text into positive, negative, or neutral also can be done in several ways including lexicon-based, machine learning, and deep learning approach which each has different constraints. The challenge to perform sentiment analysis on streaming data is not only deciding which data processing platform and classification approach to use but also preparing the text itself to be ready for the analysis. This paper evaluates all these important aspects and provides intensive evaluation and recommendation of which approach works best for which cases. After reading this paper, readers will have a comprehensive knowledge of what processes need to be done, what technology should be used, and what factors need to be considered before starting sentiment analysis on streaming data.

I. Introduction

Since the rise of social media channels such as Twitter, Facebook, and Instagram, people can easily and intensively point out their comments of any object. Facebook's active users has already surpassed the population of China by the end of 2018, while Twitter's trailing behind human population in the United States. Through this tremendous number, Forrester study mentioned at least 30% of them are actively contribute in providing reviews and recommendations over products and services in the world wide web (Neri, Aliprandi, Capeci, & Cuadros 2012). One of the top platforms to support such activities is Twitter. The latest statistics show Twitter daily active users increased by 9% over year or as much as 321 million users per month with over 200 billion tweet text posted in a year.

Companies have already recognized the benefits of this proliferating trend. Countless brands or service providers leverage their communication channel via popular social media for some purposes such as promotion, customer service, and further to dig insight from their customers. Similarly, celebrities or public figures publishing their activities in hope for more followers, have shaped the market influencer phenomenon in social media platforms, creating sort of paid sponsorship schemas with products or services owner in hope for more customers or opinions about the products or services.

For over the last decade opinion mining or widely known as sentiment analysis has been the top priority in Natural Language Processing (NLP) research since it can extract opinions from big text data, such as Tweets, to define human perceptions and eventually affect their decision-making process (Zhang, Wang, Liu 2018). Neri et al. (2012) observe that in general sentiment analysis is meant to acknowledge the characteristics of person who comment, the topic of that comment, the moment when it is posted, and the attitude towards the topic. the later one is the main task of this analysis. At very least, attitude can be classified into "positive" and "negative" and it can be extended in many ways (Balahur, 2013). By knowing this is in real time, businesses can set up recommendation system for their customers and take effect on people's behavior.

Many organizations conduct sentiment analysis using stream database from social media posts. As in 2010, in average there are approximately 3 billion Twitter API request per day (Twitter Chirp developer conference, 2010). However, to perform sentiment analysis on streaming data comes with great challenges because the data comes rapidly in massive amount, thus, the analysis approach has to deal with storage and time constraints to perform real time predictions (Bifet and Frank 2010). Other challenges are to pre-processes the text and to conduct the analytical approach. Data mining in sentiment analysis context also has various approaches including lexicon-based approach, machine learning and deep learning which each has trade-offs in terms of accuracy, complexity, data demand, and processing time.

Accordingly, this research poses a comprehensive study across sentiment analysis with an extensive evaluation on different approaches in harvesting streaming data, processing text data, and performing sentiment analysis. Most of the discussion take Twitter as the study case. This research has three main sections, the first one is discussing different big data platforms to perform analysis in streaming data, the second one is text processing to prepare data for further analysis, and the last one is comparison of several analysis approaches for sentiment classification. This research evaluates every detail process to conduct sentiment analysis on streaming data and recommendation-based constraints.

II. System Model

In this section, the literature survey done focuses on big data technologies utilized in sentiment analysis experiments. In general, those technologies have capabilities to ingest high-velocity data, store and distribute high-volume of data, and process data in parallel (either as batch or real-time processing). The combination of data ingestion, data storage, and data processing platforms shown in Figure 1 were found from several system architectures proposed by researches for supporting their sentiment analysis. Overall, they required data ingestion platform to capture streaming data from a certain data source (in this case Twitter) and then flow those data to either data storage or data processing platform (depend on the specified approach). Afterwards, the result could be provided to be visualized or consumed by the client systems. The following subsections describe how some researches employed certain platform from each block for supporting the works.

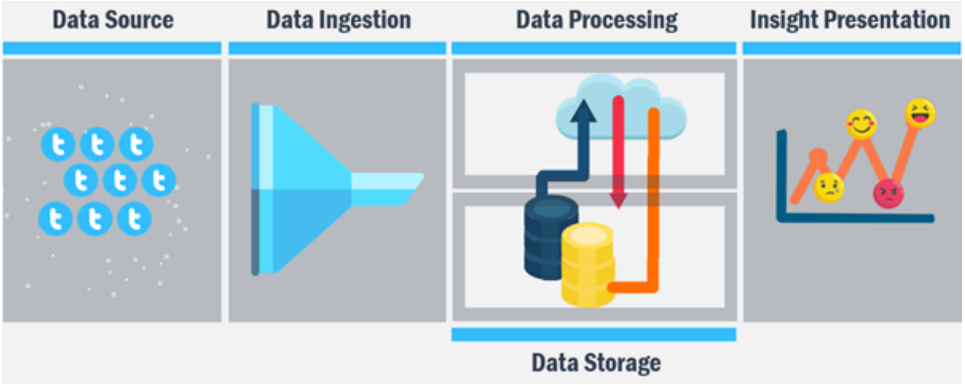

Figure 1. System Model for Sentiment Analysis

A. Data Ingestion

Aziz, Zaidouni, & Bellafkih (2018) and Rodrigues & Chiplunkar1 (2019) employed Apache Flume, a distributed system for efficiently collecting, aggregating, and moving large amounts of event data, for their works. On the other hand, Yadranjiaghdam, Yasrobi, & Tabrizi (2017) utilized Apache Kafka, a message brokering system with publish-subscribe approach. Both Flume and Kafka could be configured to communicate with twitter API (either search or streaming API), fetch the tweets in json format and sink those data into data storage platform, Hadoop Distributed File System (HDFS).

B. Data Storage

In general, there are three approaches in data processing, whether it is batch, micro-batch, or single stream at once. Regarding the first one, Hadoop MapReduce has been early platform that widely used for batch processing as being applied by Parveen & Pandey (2016), Sheela (2016), and Kanavos, Nodarakis, Sioutas, Tsakalidis, Tsolis, & Tzimas (2017). The batch processing only works after the overall data available in data storage. Therefore, stream data that are flow from data ingestion platform couldn't be processed immediately as they are available, instead they must be sunk into data storage first. On the other hand, micro-batch and single stream processing are quite flexible to process streaming data from the data ingestion platform directly and then store the analysis result in the data storage platform. Apache Spark typically treats the streaming data by accumulating them into a micro batch abstraction called Discretized Stream (DStream). Some researches utilizing Apache Spark conducted by Yan, Yang, Ren, Tan, & Liu (2017). As Apache Spark, Apache Storm also has micro batch abstraction called Trident. However, Apache Storm has true streaming processing capability through core storm. For this reason, Khumoyun, Cui, & Lee (2016) used Apache Storm in their works. Although those three approaches considered as independent one to another, Karanasou, Ampla, Doulkeridis, & Halkidi (2016) combined batch processing as offline process and stream processing as online process in their work. They applied batch processing to train classifier and develop classification model. Furthermore, this model was employed for streaming processing to predict the sentiment of streamed Twitter data.

C. Data Processing

Most of the reviewed works conducted by Rodrigues et al. (2019), Parveen et al. (2016), and Yadranjiaghdam et al. (2017) exploited HDFS for storing the tweet data or the analysis result. The

data are split into several chunks and distributed in HDFS cluster data nodes so that the data chunks could be processed in parallel. HDFS also have fault tolerant capability by replicating the data chunks on the data nodes. However, some researches also employed NoSQL Databases or SQL Engines that was coexisted with HDFS such as MongoDB, a document-based database (Sheela, 2018), HBase, a columnar oriented database (Karanasou et al., 2016), Hive, a data warehouse software that provide SQL-like interface (Rodrigues et al., 2019). These systems could be considered as an additional layer to structure the twitter data (Rodrigues et al., 2019) or store the analysis results (Sheela, 2018).

D. Insight Presentation

Not many reviewed literatures mention about the tools used for visualizing the analysis results. Only that elaborate how they use javascript library, D3, and web application framework, Flask, to present their findings. However, the other researchers such as Sheela (2016) describe how she used different visualization techniques like heatmap, tag clouds, timeline, and affinity on tweet features to present her analysis results.

III. Data Processing Platform Comparison

Social media posts are astonishingly large in size and comes from enormous growing number of users. This streaming data has all the properties to be called big data, it is high in volume, velocity, and variety as unstructured data, and it cannot be analysed in a traditional way using a single node with static configuration of main memory and disk capacity. It is simply because large RAM cannot guarantee to be able to handle a complex analytical process on large datasets with high I/O operations, instead, lead to an unfinished task (Ranjan, 2014). Furthermore, Gaber and Krishnaswamy (2005) highlights some fundamental issues in performing information extraction with interminable streams of data. Primarily those issues are related to scalability, data mining techniques, dynamic stream result, and integration of storage, analysis method, and the data itself. As the problem arise, many open source applications appear as big data engine to address those issues and, even more, to fulfil the need for fast data processing. The following sections compare three applications, Hadoop MapReduce, Apache Spark, and Apache Storm, that have been widely used to perform streaming data processing.

A. Hadoop MapReduce

1. Description

Hadoop MapReduce is a software framework used for developing applications which compute vast amounts of data (typically in terabyte size) in-parallel on large clusters (up to thousands of nodes) in a reliable, fault-tolerant manner. A MapReduce job splits the input dataset into independent chunks which are processed by the map tasks parallely. The framework shuffles the outputs of the maps which are then assigned to the corresponding reduce tasks (Apache Software Foundation, n.d.).

To manage the MapReduce job on the cluster, Hadoop follows master and slave architecture, where there is one master node, called JobTracker, and many slave nodes, called TaskTracker. The former one has responsibility to distribute the job to available TaskTrackers in the cluster and monitor the results. Meanwhile, the latter perform the actual work of the job and communicate back to the JobTracker. Typically, the JobTracker is at the same master node as

the HDFS's name node and so do the TaskTracker with the name node. Thus, TaskTracker run the job by consuming the files on the corresponding name node and write the results on this name node.

2. Application

Kanavos et al. (2017) utilized Hadoop MapReduce for implementing four consecutive jobs. Firstly, tweet features were extracted and transformed into key-value pair outputs where the key was each of the features and the value was the list of tweets that contain it. Following this, feature vectors for training set and test set were built where the key and the value of the outputs were the tweet id and corresponding features vectors. Those vectors subsequently became inputs for the third step where Euclidean distance between training set and test set feature vectors was computed. This step resulted key-value pairs where the key was the id of a vector in test set and the value was the id of a corresponding vector in training set and also the Euclidean distance value between them. Finally, on the last step, an estimated sentiment label was assigned for each test set vector based on its lowest distance to the training set vector.

Parveen et al. (2016) implemented map function in Hadoop that perform two major tasks; creating a hash map for retrieval of polarity of each word for each tweet and processing the overall polarity of each tweet by applying Naive Bayes algorithm. On the other hand, reduce function categorized the overall polarity into five different categories; extreme positive, positive, extreme negative, negative, and neutral.

Sheela (2016) implemented two map-reduce passes. In the first pass, the mapper consumed labelled tweets from training data and produced key value pair of category and word. The corresponding reducer summed up each word occurrences and outputs category and word-count pair as key-value. On the next pass, after the mapper had classified the category based on conditional probability calculation of each word, the reducer calculated the final probability and outputted the predicted category and its probability as key-value pair.

B. Apache Spark

1. Description

Apache Spark (Spark) is a fast and general-purpose framework for cluster computing system. It is claimed that Spark run workloads 100 times faster than Hadoop (validated while running logistic regression). Spark provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming for processing data streams. (Apache Software Foundation, n.d.).

The data stream abstraction of Spark is called Discretized Stream (DStream) which is produced by a data ingestion platform or an operation of other DStreams. Internally, a DStream comprises of Resilient Distributed Datasets (RDD) sequence that is considered as a micro-batch. Thus, Spark could be considered as a framework for performing batch process on a continuous basis. The processing operation is categorized as transformation and action operation. Some instances of transformation operation are map, filter, and join. This type of operation transforms the elements of incoming RDDs into new RDDs (Yadranjiaghdam et al.,

2017). On the other hand, action operation such as count, collect, and reduce, do not result any RDDs. Instead, they produce non-RDD values which will be sent to master node or be stored to external data store.

As a distributed computing framework, Spark applies master and slave architecture on runtime where there is one central coordinator, known as driver program, which manages multiple distributed workers, known as executors, in a cluster. Inside the driver program, there is SparkContext, a master application which responsible to establish connection to the worker nodes (with help from cluster manager) and assign an executor application to each of those workers. Furthermore, the driver program executes the job by splitting it into multiple tasks and distributing the tasks via SparkContext to be run by executors over the worker nodes. Finally, the executors return the results to the driver program (DataFlair, 2017).

2. Application

Yan et al. (2017) utilized Apache Spark to parallelize Support Vector Machine (SVM) classification along with Term Frequency - Inverse Document Frequency (TF-IDF) as the weighting method. They represented microblog text data from Twitter and Weibo as RDDs. These RDDs then flew through TF-IDF calculation process that involved three Apache Spark's operations (HashingTF, IDF and IDFModel) implemented in Scala. The map and reduce operations were also used to calculate the kernel matrix of Radial Basis Function along with SVM.

Kanavos et al. (2017) utilized Apache Spark's operations of Map and Reduce functions to adopt their first implementation in Hadoop MapReduce. They also employed Machine Learning Library (MLlib) in the operations to perform sentiment analysis based on determined machine learning methodologies.

C. Apache Storm

1. Description

Apache Storm is a free and open source distributed real-time computation system. Storm makes it easy to reliably process unbounded streams of data, doing for real-time processing what Hadoop did for batch processing. Storm can be used with any programming language. Storm supports many use cases, such as real-time analytics, online machine learning, continuous computation, distributed RPC, and ETL. It is claimed that Storm can process over a million tuples per second per node. A Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed (Apache Software Foundation, n.d.).

The basic Storm processing framework involves streams of tuples that flows through topologies (Tonshiwal et al., 2014). A topology is an arrangement of vertices that represent computation and directed edges that represent data flow between the computation elements. There are two types of vertices; Spout and Bolt. Spouts are the source of stream data that are being pulled by Bolts to be processed. Bolts can perform the operations of filtering, aggregation, joining, interacting with data sources and databases, and emitting results to the other Bolts. The topology is submitted to Nimbus as master node which allocates the computation among worker nodes in a cluster. Each worker node has one or more worker

processes that each of them runs several executors in a Java Virtual Machine (JVM). Executors are made of one or more tasks that perform the actual works of Spouts and Bolts. The coordination between master node and worker nodes is maintained by Zookeepers as centralized service provider of distributed systems.

2. Application

Khumoyun et al. (2016) defined a Storm topology in three phases; Pre-processing, Feature Extraction, and Classification. The first phase began with retrieving twitter streaming data that was performed by a spout, CrawlerSpout. The tweets data then were pulled from the spout by DataCleanerBolt to be cleaned. Subsequently, PreprocessingBolt pulled those data and performed pre-processing steps such as text normalization and stemming. Turning to Feature Extraction phase, there were three bolts involved, FeatureExtractorTFIDFBolt, FeatureExtractorBOWBolt, and FeatureExtractorSTATBolt, which had responsibilities in extracting Term Frequency – Inverse Document Frequency and Bag-of-Words, and also applying $\chi 2$ statistical method. Finally, in the last phase, three bolts, NBClassifierBolt, SVMClassifierBolt, and SMOClassifierBolt, classified tweets based on several machine learning classification algorithms such as Naive Bayes, SVM (Support Vector Machines), and SMO (Sequential Minimal Optimization). The classification results then were written to the filesystem for further analysis.

D. Overall Comparison

Overall comparison between above mentioned three data processing platform can be seen on Table 1. However, the available support of machine learning library and the concern of performance and latency are the focus of this subsection as most of the time they are the main factors when deciding which platforms should be used in sentiment analysis work.

	Hadoop MapReduce	**Apache Spark**	**Apache Storm**
Processing Model	Batch	Micro-Batch	Micro-Batch (Trident), and Single Stream (Core Storm)
Workload	CPU, Memory, and I/O intensive	CPU and Memory intensive	CPU and Memory intensive
Latency Level	High (impact of I/O operation)	Low (benefit of applying in-memory processing)	Very Low (advantage of its design as true streaming processing)
Machine Learning Support	Apache Mahout (third party)	MLlib (spark module)	Apache Samoa (third party)
Stream Source	Data Ingestion Platform	Data Ingestion Platform	Data Ingestion Platform and Spout

	Hadoop MapReduce	Apache Spark	Apache Storm
Stream Primitive	Files	DStream (a continuous sequence of RDDs)	Tuple
Computation Operation	Mapper, Reducer	Transformation, Action	Bolts
Master Daemon	JobTracker	DriverProgram	Nimbus
Worker Daemon	TaskTracker	Executor	Supervisor (managed by Zookeeper)

Table 1. Overall Data Processing Platform Comparison

1. Machine Learning Library Support

When researchers decide to apply machine learning approach for performing sentiment analysis, the availability of machine learning package on data processing platform has been of great concern to them. It is believed that the availability of MLlib (machine learning library) has been the advantage of Apache Spark than Hadoop MapReduce and Apache Storm. MLlib includes popular machine learning algorithms for performing sentiment analysis such as Naive Bayes, Logistic Regression, Support Vector Machine, and Decision Tree. MLlib offers API that is seamlessly compatible with Apache Spark, while Hadoop MapReduce and Apache Storm should struggle to utilize third party machine learning packages such as Apache Mahout and Apache Samoa.

2. Performance and Latency Benchmark

It should be recalled that these discussed data processing platforms have different design one to another. While Hadoop MapReduce is designed for distributed batch processing, Apache Spark is implemented to process micro-batch and Apache Storm has capability to continuously compute single stream at once. Thus, it is difficult to have apple-to-apples comparison between them. However, some researches succeeded to find benchmark method for comparing Hadoop MapReduce to Apache Spark and Apache Spark to Apache Storm.

Harazika, Ram, & Jain (2017) found that Spark outperformed Hadoop with performance ratio between them were 2.8207 for word count and 2.1745 for logistic regression analysis scenario while using single node cluster. They mentioned that Spark performance experienced deterioration while doing iterative queries like logistic regression as the cache memory was limited in size.

Meanwhile, after implementing the same approach of sentiment classification on Hadoop MapReduce to Apache Spark, Kanavos et al. (2017) discovered that the running time in Spark was two up to four times faster than Hadoop MapReduce depended on different setups. One of the reasons that they believed was the different approach of these two platforms on handling intermediate data. Hadoop does not cache intermediate data, instead it flushes the data to the disk between each step. On the other hand, Spark maintains the data in the workers' memory.

In this way, Spark exceeds Hadoop performances, especially for algorithms with multiple operations.

Chintapali et al. (2016) implemented benchmark testing between three of the most popular platforms (Strom, Flink, and Spark). The result shows that Storm had linear response time increase while processing the test dataset due to its design to process an incoming event as it becomes available. On the other hand, the Spark had a stepwise manner behaviour, as a result of its micro-batching design. Further, they suggested that Spark performance still could be improved by configuring its batch interval appropriately.

IV. Data Preprocessing

The data that has been collected need to be pre-processed before it can be used for the analysis. The form of language in Social Media is often different from the standard form found in a dictionary and mainstream media. Furthermore, there are "slang" like informal language with abbreviations, emoticons, tags and other characteristics that could only be found in the Social Media platform. These aspects must be considered during data processing. Meaningless terms that do not contribute to the analysis should be removed to save storage and increase the efficiency of the analysis. This section will discuss the common data preprocessing approaches and suggest a structured step by step guideline for preprocessing tweets from Twitter. Typically, several techniques need to be followed in language processing, namely normalization, tokenization, and part of speech (POS) tagging.

A. Normalization

As mentioned above, the words and expressions used in Social Media are often not in their standard form. The ill-formatted and noisy text will be normalized to increase the performance of language processing and information retrieval for the sentiment analysis. Some normalization processes that need to be done include:

1. Lower casing words in the text

All the words in the text need to be lower-cased so that they will not be registered as multiple features because of the different placement of capital letter on the same word. It is expected to have "Ball","ball", and "bAll" as the same feature because their meanings are the same.

2. Extracting and keeping specific properties

In Twitter tweets data, there are some unique properties which make it different from other social media platform's data. There are some of the properties that could be useful for sentiment analysis and the other properties are not. Useful properties should be kept in the training data and normalized to the uniform format while the not useful one should be removed.

a. URL Links

Twitter users often post links in their tweets which are then automatically altered to 23 characters (Twitter, 2019). The URL link in tweets will not be useful for sentiment analysis. Therefore, the URL link should be removed or replaced with "weblink" token.

b. Usernames

One of the most popular features of Twitter is mention. When people want to refer other users, they can use the @ sign followed by the username they want to mention in their tweets, e.g., "Hello @Elon!". The choice to remove the username from the text or not depends on the research's goal. Chatzakou et al. (2017) used both user-based and text-based approaches in their study to detect aggressor and bullies on Twitter. In a user-based approach, in order to reduce the feature space, it is suggested to replace all the words which start with the @ sign with the equivalent token "username". On the other hand, if the research is text-based, the username should be filtered and removed from the text as the username will not be useful in the analysis.

c. Hashtags

Twitter promoted the popularity of using the hashtag symbol # before a relevant keyword to emphasize the topic of the tweets, e.g., "people are excited about the #election". Again, the choice to remove or keep this feature depends on the research's approach. Chatzakou et al. 's study observed the number of hashtags used in aggressors and bullies' tweets. Therefore hashtags were kept as features in the analysis. However, there are also other researches which removed hashtags from tweets to minimize bias in classification because of the existence of hashtags.

d. Emoticons

In recent years, people are getting used to the use of graphical icons called emoticons that mimic facial expression to convey emotions in computer-mediated communication (Hogenboom, 2013). There has been some sentimental analysis based on emoticons research conducted. Emoticons are usually incorporated in the analysis by assigning them with their polarity score. Jonathon Read's study suggested that emoticon-trained classifiers could perform well and achieved up to 70% accuracy (Read, 2005). However, sentiment analysis based on emoticons still has a problem dealing with texts which contain mixed sentiment and sarcasm. Emoticons could also be filtered in order to minimize bias in positive and negative classification.

e. Retweets

Another distinctive feature of Twitter is retweet. People can republish another user's statements or reply to that particular tweet using this feature. Retweets are marked with "RT" in front of the text that is being quoted. This "RT" token can be used as a mark to remove all texts that follow that token as those are redundancy from another tweet. Only the text in front of "RT" that should be kept as they are the unique reaction to that particular tweet.

3. Handling repeated punctuation signs

Multiple consecutive punctuation signs can be replaced with the equivalent token "multistops" for full stops, "multiexclamation" for exclamation signs, and "multiquestion" for question marks. There are already some studies which generate the polarity for punctuations like: "…",

"?!?", "!!!", etc. There is also another approach which obliterates punctuations from the text as they are considered as meaningless characters for the sentiment analysis.

4. Removing stopwords

Opposite to the natural instinct to believe that a word will become more significant if it appears more frequently in a document, the fact is it does not carry any significance as that word would most surely be the common words like "the" or and" (Rajaraman, 2011). In computing, these commonly used words are called stop words, and they need to be filtered out as they are meaningless for sentiment analysis and removing them will decrease the number of features and storage space. The most popular way to collect a list of stop words is by using the pre-existing stop word lists in the tool.

5. Replacing negation

Negation is a grammatical construction that contradicts the meaning of a sentence. It should be taken into consideration in sentiment analysis as it affects polarity (Wiegand, 2010). In the English language, negation is marked by a negating word "not". Therefore, contractions like: don't, shouldn't, won't should be replaced with "not" token to indicate that the tweets contain negations.

6. Replacing out-of-vocabulary words

The language in social media is informal and includes unstandardized abbreviations, spelling errors, etc. It will be hard to directly find matches for these words in the dictionary, therefore, further processing needs to be done.

a. Correcting spelling errors.

   Edit distance and N-Gram are some of the popular methods to get candidates for correct words given a wrongly spelled word. Each word in the text will be compared to entries in the corpus. Spelling errors like "perrrrfeeeeect" and "beautyful" could easily be detected and replaced by the intended words "perfect" and "beautiful".

b. Replacing acronyms and slang

   Informal abbreviations and terms in Social Media conversation would not be found in the dictionary list. Some research had been done to map common slang used in Social Media. This list could be used to include the semantics of the slang in the sentiment analysis.

Words that cannot be found in the dictionary after these processes being done would be removed from the text.

B. Tokenization

The sentiment analysis will be conducted by analysing each word in the text. Therefore, the tweet text will need to be split into a series of tokens based on whitespaces, like space and line break, and punctuation signs to indicate a separation of different words.

C. Part-of-Speech (POS) Tagging

A Part-Of-Speech Tagger is a piece of software that reads the text and put a label to each word in the text which corresponds to a particular part of speech, such as noun, verb, adjective, etc., based on both the word's definition and context. Word-sense disambiguation is a challenge to identify which sense of a word is used and which part of speech it is placed in a sentence, when the word has multiple meanings, for example:

The bird flies in the sky

Rotten meat attract flies

The word ‘flies' in the above sentences has different sense and word-category, one is a verb; the other one is a noun. Therefore, it is essential to understand what specific meaning is being conveyed by the given sentence to make sure that the analysis uses the right sense for the word. There are two distinctive approaches in POS Tagging:

1. Rule-Based Tagging

Rule-based approaches use contextual information to assign tags to unknown or ambiguous words. Solving disambiguation is done by analysing the linguistic features of the word, its preceding word, its following word, and other aspects that are coded inside the rules. For example: "Nouns can appear after determiners and adjectives, and can be the subject or object of the verb" (Bird, 2009).

2. Stochastic Tagging

Any model which incorporates frequency or probability could be addressed as stochastic tagging approach. The simplest stochastic tagging approach calculates the probability that a word occurs with a particular tag and will assign the most frequent tag each time it encounters the instance of the word. Another approach calculates the probability of a given sequence of tags occurring. This approach is called n-gram approach as it calculates the probability of a tag occurs together with the n previous tags. A more advanced approach called Hidden Markov Model (HMM) incorporates both tag sequence probabilities and word frequency measurements for tagging a label to each word in a sentence.

Most commonly used part-of-speech tagging sets includes Penn Treebank (48 tags), Brown (87 tags), and Universal (12 tags). Table 2 shows the list of Penn Treebank tagging sets.

1.	CC	Coordinating conjunction	25.	TO	*to*
2.	CD	Cardinal number	26.	UH	Interjection
3.	DT	Determiner	27.	VB	Verb, base form
4.	EX	Existential *there*	28.	VBD	Verb, past tense
5.	FW	Foreign word	29.	VBG	Verb, gerund or present participle
6.	IN	Preposition or subordinating conjunction	30.	VBN	Verb, past participle

7.	JJ	Adjective	31.	VBP	Verb, non-3rd person singular present
8.	JJR	Adjective, comparative	32.	VBZ	Verb, 3rd person singular present
9.	JJS	Adjective, superlative	33.	WDT	Wh-determiner
10.	LS	List item marker	34.	WP	Wh-pronoun
11.	MD	Modal	35.	WP$	Possessive wh-pronoun
12.	NN	Noun, singular or mass	36.	WRB	Wh-adverb
13.	NNS	Noun, plural	37.	#	Pound sign
14.	NNP	Proper noun, singular	38.	$	Dollar sign
15.	NNPS	Proper noun, plural	39.	.	Sentence-final punctuation
16.	PDT	Predeterminer	40.	,	Comma
17.	POS	Possessive ending	41.	:	Colon, semi-colon
18.	PRP	Personal pronoun	42.	(	Left bracket character
19.	PRP$	Possessive pronoun	43.	)	Right bracket character
20.	RB	Adverb	44.	"	Straight double quote
21.	RBR	Adverb, comparative	45.	‘	Left open single quote
22.	RBS	Adverb, superlative	46.	‘ ‘	Left open double quote
23.	RP	Particle	47.	’	Right close single quote
24.	SYM	Symbol	48.	’ ’	Right close double quote

Table 2. List of Penn Treebank Tagging set (Taylor et al., 2003)

D. Stemming and Lemmatisation

Stemming and lemmatization are two approaches to obtain the roots of words by removing inflections from the word. The main difference between stemming and lemmatization is stem might not be an actual word whereas, a lemma is an actual language word. Stemming follows an algorithm with steps to strip recognize prefixes and suffixes, whereas lemmatisation takes the algorithm further by looking up the words' part-of-speech tags before determining the lemmas for the given words. In order to compare stemming and lemmatisation, one can try applying both approaches to the word "laziness". The stem for "laziness" would be "lazi" which is not an actual word, while the lemma is "lazy" which exists in the dictionary. These techniques are more useful in tasks like information retrieval where the user wants to retrieve the same result for "dog" and "dogs", for example. Stemming and lemmatisation could also be used in sentiment analysis tasks if the researcher does not want to distinguish the polarity of the word based on its form.

There is no definite steps to do preprocessing. The researchers can choose which processes they want to include or exclude depending on the goal of their research. Suppose that there is a synthesized tweet:

"@Elon this isn’t a great day for playing SOCCER. #windy day… :("

Figure 2 could be followed to help to capture the polarity of the tweet easier in the analysis later.

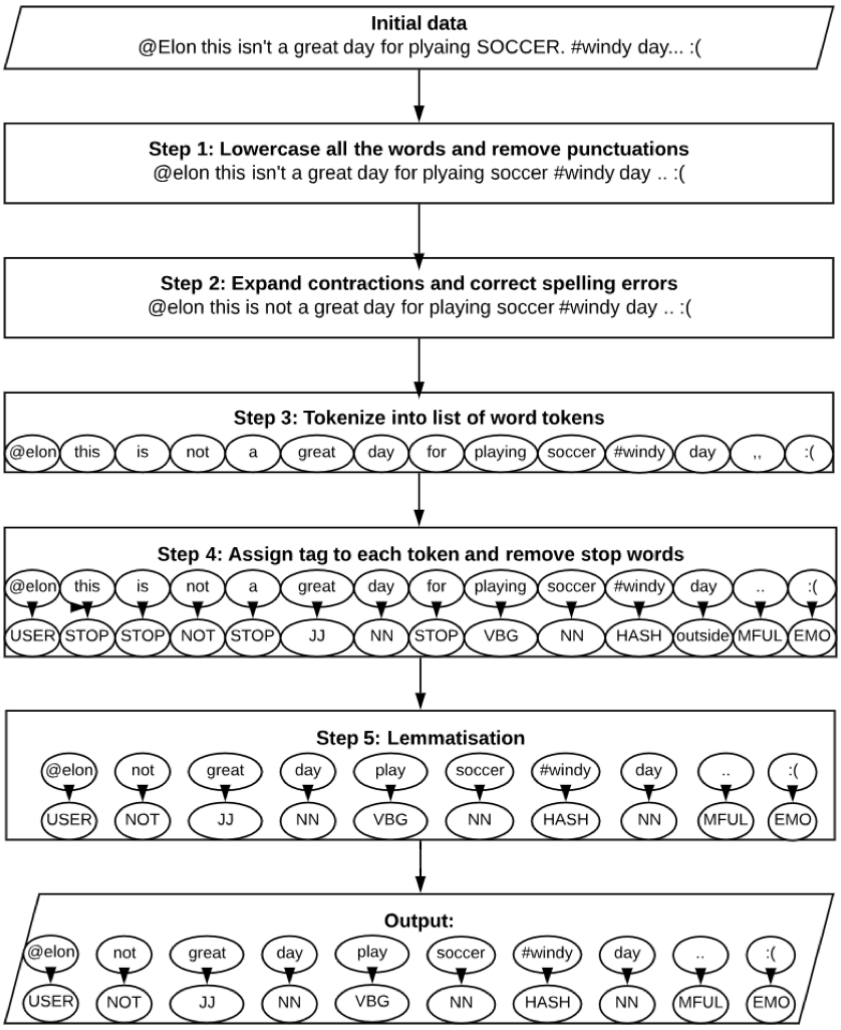

Figure 2. Suggested Data Pre-processing Steps

The first normalisation technique that can be applied is lowercasing all the characters in the tweet to prevent registering the same feature as different ones because of the use of the capital letter. Punctuations could also be removed as it will not help in the analysis later. One can decide to keep multiple punctuations as a feature if it will be used as part of the research. Then, the text of tweets is normalised by expanding contractions to find negation in the text and also correcting spelling errors so that every word in the text can hopefully be found in the dictionary. After that, the text is tokenized based on whitespaces so that there is a list of word tokens. Finally, the tags for each token can be assigned:

1. if the token is a Twitter's feature

   give "USER" tag for mentions, or "LINK" for URL link, or "HASH" for hashtags, or "EMO" for emoticons

2. if the token is a negation word

   give “NOT” tag to the word

3. if the token exists in the list of stop words

   remove the token

4. if the token exists in the dictionary

   refer to Penn Treebank tagging set for the right tag

5. if the token is repeated punctuation signs

   give “MFUL” for multiple full stops, or “MQUE” for multiple question marks, or “MEXC” for multiple exclamation marks.

The last step is doing lemmatisation to get the basic form of each word based on WordNet’s lexical knowledge.

These processes are the most common steps in preprocessing tweets. However, it can also be adjusted depending on the case. Since tweets usually use casual language, some processes such as spelling correction and slang replacement can be complex tasks. Moreover, tweets can also contain other languages which ideally need to be translated to the main language in the tweet. Therefore, these constraints should be taken into account before performing sentiment classification.

V. Sentiment Analysis Approaches

After text are normalized and all important features that can describe the syntactic and semantic distribution of the text such as POS tag are extracted, the next step is to perform the sentiment classification. The end result of sentiment analysis is the polarity of the text, whether positive, negative, or neutral.

A. Lexicon Based Approach

This approach assumes that overall polarity of a collection of texts is defined by the polarity of individual text or phrase. Kaushik and Mishra (2014) explain that every word can be associated with a polarity strength. They illustrate using *‘good’* and *‘great’*, both are positive words but the polarity strength of the later outweigh the former. Furthermore, they elaborate that negation words (*such as ‘not’*), blind negation words (such as *‘needed’*), and negative emotions (such as L) on a text contain positive word can reverse the polarity into negative sentiment.

A more complicated approach using syntactical tree and Word Sense Disambiguation algorithm introduced by Neri et al. (2012). Basically, this approach is the extended version of the above approach by taking into account all important information contained in the text including entities, places, time markers, noun phrases, part-of-speech tags (such as noun, verb, adjective), and semantic role (such as agent, object, cause) to find the polarity of a word related to a given context. Accordingly, by comparing text with those set of rules, the sentiment is easily classified in one process. It even can be performed without a pre-labeled dataset. This lexicon-based approach is

fast and suitable for real time sentiment analysis in big streaming data. However, if using the pre-label dataset, in most cases, the accuracy hardly outweighs the machine learning approaches' (Kaushik and Mishra 2014).

B. Machine Learning Approach

Predictive model is built upon the training set and evaluated by the validation set. The main concern before performing any approach is to get the pre-labeled data as the training set. Thus, in this part, the first discussion is about getting the training set, followed by various ways to perform the sentiment analysis.

1. Training Set Collection

The basic way to get the training set is by manually label the data. Neethu and Rajasree (2013) use human annotated label on their Twitter data, 600 tweets for each positive and negative class that is later divided into training set and test set. Their Naive Bayes classifier return nearly 90% accuracy. Stanford Twitter Sentiment test set (STS-test) is another human annotated Twitter data which only contains 177, 182 and 139 of positive, negative and neutral sentiment respectively but has been used widely in various evaluation scenarios (Saif, Fernandez, He, & Alani 2013). These are examples of qualified human annotated datasets, which small in size but contribute to return high precision. The main concern here is this kind of annotation is given by experts which can be a lengthy and costly task. Crowdsourcing could be a way to have fast and cheap human annotated dataset. However, since it is done through online recruitments, the annotators highly possible to be non-experts. In order to reach the same quality as the expert annotators, there should be some adjustment such as noise eliminations which, becomes a further tedious process (Hsueh, Melville, & Sindhwani 2009).

An automatic labelling using emotion icons to the rescue. It offers fast, costless and simple way to automatically label the dataset. Some emoticons are categorized into positive and negative. Text with such emoticons are labelled according to number of pure emoticons contained in the text. If a text purely contains positive emoticons, it is labelled as positive sentiment, and vice versa (Chikersal, Poria, Cambria, Gelbukh, & Siong 2015). The Stanford Twitter Sentiment (STS), one of the most commonly used pre-labeled Twitter dataset, using this approach. It has 1.6 million tweets with negative, positive, and neutral sentiment (Go, Bhayani, & Huang, 2009). It is obvious that automatic labelling is more efficient than human annotated in terms of processing time. However, this type of automatic labelling may lack in accuracy since not all tweets' actual sentiments can be represented through emoticons (Saif et al. 2013). Therefore, before using pre-labeled dataset or conduct automatic labelling using emotions, one should make sure that the label is matched.

Orientation	List of Emoticons
Positive	(-: , (: , =) , :) , :-) , =') , :') , :'-) , =-d , =d , ;d , :d , :-d , ^_^ , ^_^ , :] , ^_- , ^_* , ^^
Negative	)-: ,): , =(,]: , :[, :(, :-(, >;(, >:(, :_(, d'x , :'(, :"(, ='[, :'(, :'-(, \: , :/ , (~_~) , >__> , <('-')> , </3

Table 3. Human Annotated Emoticons Classification List for Automatic Labelling by Chikersal et al. (2015)

A major breakthrough in sentiment analysis is to detect polarity not only in text-level but also in entity-level. As mentioned by Saif et al. (2013), a tweet text "*iPhone 5 is awesome, but I can't upgrade*" should have negative sentiment as a whole, but the entity "iPhone 5" should be tagged with positive sentiment. Based on this concern, they introduce STS-Gold dataset to be used for that requirement. It begins with STS dataset and ends with human annotators agreement on given labels, thus this dataset is a mixture of automatic labelling and human annotation approach. It contains 2,206 tweets tagged with negative, positive, neutral, mixed, and others sentiment; and 59 entities with the same sentiment label. It is worth noted there are massive pre-labeled datasets available online with various labelling approaches.

2. Feature Selection

Beside the terms in the text, it is suggested to add more features into the model such as n-gram (unigram and bigram), term frequency, part-of-speech tag, negations flag, number of special keywords, emoticons, hashtags based on its polarity group (positive or negative), and retweet status (Neethu and Rajasree, 2013). All the features are represented in a feature vector. The polarity class is determined according to the highest probability given the feature vector for each class. The model is trained several times by adjusting the hyperparameter until it returns the highest evaluation metrics including accuracy, precision, recall and F-score. This explains why this exhaustive task can return better accuracy than the lexicon-based approach but considered impractical for big streaming data unless the analytical process run in a high performing application model.

3. Sentiment Classification Approaches Determination

The most common supervised learning approaches for sentiment prediction are Naive-Bayes, Max-Entropy and Support Vector Machine. These approaches are different in algorithms but share the same steps and evaluation metrics. This section discusses the key differences among the most common sentiment classification approaches.

a. Naive Bayes

$$P(class_j|features) = \frac{P(feature_i \dots feature_m|\, class_j)P(class_j)}{P(feature_i \dots feature_m)}$$

$$P(class_j|features) \approx P(feature_i \dots feature_m|\, class_j)$$

$$P(class_j|features) \approx \prod_{i=1}^{m} P\,(feature_i|class_j)$$

In the above equations, 'features' stand for feature vector and '$class_j$' for particular sentiment polarity, it could be positive, negative, and neutral. Naive-Bayes Classifier

computes the probability of each class given features and return class with the highest probability. It is notified in the formula that NB utilize all the features but not the relationship between them (Neethu and Rajasree, 2013). This is due to independent assumption between features, the reason behind 'naive'. Even though this assumption generally wrong, the popularity of ND remains the top classifier for Natural Language Processing (NLP) tasks because of its simplicity and efficiency (Kaushik and Mishra, 2014).

b. Maximum Entropy

$$P_\lambda(class|\ features) = \frac{1}{Z(features)} e^{\sum_i \lambda_i f_i(features|class)}$$

$$f_i(features|class) = \begin{cases} 1, & if\ features = feature_i\ and\ class = class_j \\ 0, & otherwise \end{cases}$$

In the above equations, 'features' stand for feature vector, 'class' for sentiment label, '$Z(features)$' for the normalization factor, 'λ_i' for the weight coefficient, and '$f_i(features|class)$' for the feature function. As its name, Max-Entropy Classifier is mainly work on maximising entropy through conditional distribution of the class given features. Unlike NB, this approach does not have any assumption regarding features relationship, but it accounts the overlap between features effectively (Neethu and Rajasree, 2013).

c. Support Vector Machine (SVM)

$$g(features) = w^T \phi(features) + bias$$

Support Vector Machine originally comes from linearly separable idea in which there are two parallel lines (hyperplanes) with maximum margin that split data into two classes. In practice, SVM allows biases and adopts kernel trick to plot features in high dimensional space thus dataset generally separated by a non-linear kernel function. In the above equation, '$\phi(features)$' is the kernel function, 'w' is the weight vector given to the active points in support vectors, and 'bias' is the bias vector. These parameters are selected in the training process to maximize the margin between classes (Neethu and Rajasree, 2013).

By comparing the complexity of the algorithms among the three machine learning approaches, NB tends to be the simplest one which may end up in faster processing time. However, it may suffer from lower accuracy score compared to the other two because of its naive assumption. ME accounts the overlapping feature thus it should return better performance than NB. Similarly, SVM maps the feature vector in high dimension scope which account the distribution between features. Thus, theoretically its performance is comparable with ME. The main drawback of SVM is it is intended to binary classifier. Performing sentiment analysis for three classes require three different models which is time consuming, especially for big data. Even for one model, it can consume more time compared to the others because of the high dimensionality mapping.

The above hypothesis is supported by several related researches. According to Neethu and Rajasree (2013), the top performers in their research are SVM and ME which return similar accuracy score (90%) followed by NB (89.5%). While Hamad, Alqahtani, & Torres (2017) find that SVM accuracy exceed Naive Bayes by 9.9% but the training time went slower than NB training time by more than 100 times in average. Although these two researches are not comparable, the logic behind the trade-offs between time and accuracy match the hypothesis. Therefore, one should consider these factors before applying the algorithm in the big data processing engine.

The different performance between these approaches is comparable within the same feature vector. However, increasing the accuracy score can be done by modifying the feature vector itself. Both Balahur (2013) and Chikersal et al. (2015) conclude that classification result can be improved through feature modification and filtering. They both work mainly in n-gram modification by using SVM as the classifier. Balahur (2013) try different combination of unigram and bigram with some set of rules. He finds that putting both unigram and bigram together can improve the model performance. On the other hand, Chikersal et al. (2015) have a unique strategy in modifying n-gram. They modify based on conditional words such as "*but*", "*if*", "*unless*", "*until*" and "*in case*" which turns out this modification significantly improve the model evaluation metrics. These feature selection techniques can be adopted in any machine learning approach. Therefore, the best approach depends on the resources, with more sophisticated infrastructure more complicated approach can be applied to return the intended performance.

C. Deep Learning

Most works employing this approach are inspired by Sentiment-Specific Word Embedding (SSWE) (Tang, et al. 2014) to represent word into a dense and meaningful vector. While the general word embedding is good enough and (some) are available in ready-to-use model, those are likely misleading in the field of sentiment analysis (e.g. word "good" and "bad" have close representation). In the initial experiment of Twitter sentiment classification using SSWE, tweets are represented in vectors composed by *min*, *max* and *average* convolutional layers using continuous representation of words and phrases as the features. The results show superiority compared to some sophisticated feature-engineering machine learning classification (Table 4). Additionally, the quality of word representation is benchmarked against some existing neural network model (i.e. C&W and word2vec). Performance in popular sentiment lexicons dataset are showing better results (Table 5).

Method	Macro-F1
DistSuper + unigram	61.74
DistSuper + uni/bi/tri-gram	63.84
SVM + unigram	74.50
SVM + uni/bi/tri-gram	75.06
NBSVM	75.28
RAE	75.12
NRC (**Top System** in SemEval)	**84.73**
NRC - ngram	84.17
$SSWE_u$	**84.98**
$SSWE_u$+NRC	**86.58**
$SSWE_u$+NRC-ngram	**86.48**

Table 4. Macro-F1 on positive/negative classification of tweets (Tang, et al. 2014)

Embedding	HL	MPQA	Joint
Random	50.00	50.00	50.00
C&W	63.10	58.13	62.58
Word2vec	66.22	60.72	65.59
ReEmb(C&W)	64.81	59.76	64.09
ReEmb(w2v)	67.16	61.81	66.39
WVSA	68.14	64.07	67.12
$SSWE_h$	74.17	68.36	74.03
$SSWE_r$	73.65	68.02	73.14
$SSWE_u$	**77.30**	**71.74**	**77.33**

Table 5. Accuracy of the polarity consistency of words in different sentiment lexicons (Tang, et al. 2014)

Annual competition of Semantic Evaluation (SemEval) shows outstanding works on performing sentiment analysis on Twitter data of various levels. While the challenges are different on each year, the approach of using deep learning is converged from some top-scorers in 2015 into the most popular approach in 2017 (Rosenthal et al. 2015; Rosenthal, Farra & Nakov. 2017). Two common methods used are Long Short Term Memory (LSTM) Networks and Convolutional Neural Networks (CNN). The following descriptions showing architecture of the early implementation.

1. CNN

CNN consists of several layers with multiple convolving filters that are applied in local features. A work of sentiment analysis using this approach shows the architecture of CNN as Figure 3 (Severyn & Moschitti. 2015).

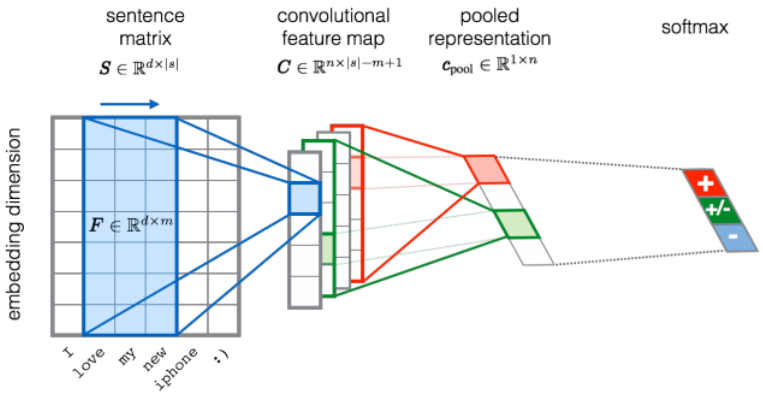

Figure 3. Deep learning architecture model for sentiment classification (Severyn & Moschitti. 2015)

The sentence matrix contains the vector representations of tweets based on word embedding matrix. Convolutional feature map works to recognize discriminative patterns found in the inputs. To gain a better representation, the model applies more than one feature map in parallel. Each of those layers are followed by activation function such as sigmoid, tanh or ReLU. Pooling layer manage to accumulate the results and reduce the representation with regard of bias provided from convolutional layer as well. Some options for pooling operations are k-max and average pooling. Lastly, softmax layer computes the probability of the label respect to their weight vector and bias (Severyn & Moschitti. 2015).

Aforementioned components in CNN are subject to experiments and task specific. An early work by Kim, Y. (2014) shows superiority in the case of sentiment analysis on MPQA dataset compared to some sophisticated machine learning approaches.

2. LSTM

Motivated by continuous features representations as well as avoiding feature engineering, Tang et al., (2015) propose a basic LSTM approach making use of sentence representation as the feature to predict the sentiment. Sentence is translated into word embedding, stacked in word embedding matrix. The architecture (Figure 4) is similar to standard Recurrent Neural Network (RNN), however, LSTM cell has three extra gates (input, output and forget) to remember input, forget history and generate output vector respectively. Softmax layer computes the probability of given sentiment class.

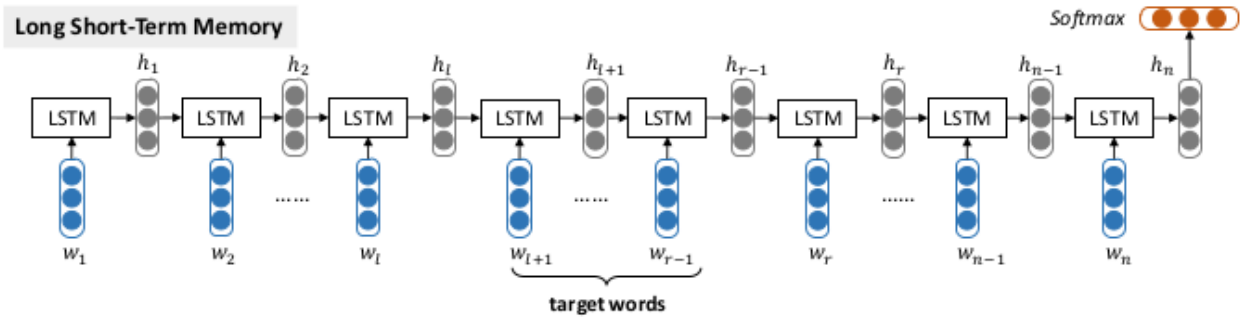

Figure 4. Basic LSTM architecture (Tang et al., 2015)

The network is trained in a supervised learning framework with cross-entropy-error as the loss function. Tang et al., (2015) compares the performance against different methods and found this model is about on par (Table 6). Therefore, enhancements are made in the LSTM model to consider target words and target dependencies.

Method	Accuracy	Macro-F1
SVM-indep	0.627	0.602
SVM-dep	0.634	0.633
Recursive NN	0.630	0.628
AdaRNN-w/oE	0.649	0.644
AdaRNN-w/E	0.658	0.655
AdaRNN-comb	0.663	0.659
Target-dep	0.697	0.680
Target-dep$^+$	0.711	**0.699**
LSTM	0.665	0.647
TD-LSTM	0.708	0.690
TC-LSTM	**0.715**	0.695

Table 6. Comparison of different methods on target-dependent sentiment classification (Tang et al., 2015)

SemEval competition on Twitter sentiment analysis shows loads of improvement done based on the two approaches. Common ways are found by pre-trained the (CNN) network by using various embeddings (Severyn, A., & Moschitti, A., 2015), furthermore various parameter tuning and embeddings are used in ensemble method (Yin, Y., Song, Y., & Zhang, M., 2017). One of the top performers in this competition (Cliche, M., 2017) ensembles 10 CNNs and 10 LSTMs prior to applying different hyper-parameters and pre-training strategies.

D. Comparison Between Sentiment Analysis Approaches

According to the discussion above, choosing the best sentiment analysis approach over the others comes with several constraints. It depends on users' resources, both human and computing power. Trend shows deep learning offers the most promising result performance however it has high data demand, complex program, and longer processing time. Deep learning works better with many

feature extractions and it does not require specific feature selection as it will find the relation within the features. The features-hungry nature of deep learning will likely to replace manual features-hand-picking approach in machine learning.

Machine learning could offer high performance result too but with extra care in feature selection and choosing the right methods. Whilst, lexicon-based approach could be the simplest among the others as it can be applied for a single twitter without extra efforts to train a model. However, it might be not applicable for real business case with high accuracy demand as it only rely on word sentiment and not capture the semantic distribution of the text. There is a case when businesses want to show the sentiment over their products based on buyers comments. In this case, deep learning and machine learning are the foremost options since showing wrong sentiment can lead to further issue when dealing with sensitive people. One way to perform complex methods such deep learning and machine learning in a reduced processing time is by utilizing the big data platform as mentioned in the second and third section.

Table 7 shows general comparison of the three approaches. There is no specific range of Low, Moderate, and High, these words stand as ordinal rank between one approach over the others in general cases.

	Annotated Corpus	Data Demand	Complexity	Processing Time	Accuracy
Lexicon Based Approach	Not Necessary	Low	Low	Low	Low/ Moderate
Machine Learning	Needed	Moderate	Moderate	Moderate	Moderate/ High
Deep Learning	Needed	High	High	High	High

Table 7. General comparison between Sentiment Analysis Approaches

VI. Conclusions

Performing sentiment analysis on Twitter is unique compared to other NLP tasks. Tweet does not always follow grammar rules as well as the availability of annotated corpus is limited. Therefore, preprocessing step has to be done carefully to prevent loss of important features to be used in sentiment classification. There are three main approaches in classification task which each has its own benefits and drawbacks. Lexicon based gives a good start by providing heuristic method, but unfortunately does not work for sarcasm and unseen word. Machine learning success relies heavily on large annotated data and a set of efficient features to gain the best performance. While exploring deterministic features for sentiment analysis is labour-intensive, there comes the deep learning approach which exploit the word sequence as the main feature. The later approach is proven favorable in the SemEval task of Twitter sentiment analysis and it is highly recommended for task with high accuracy demand.

Further, desired model should be applied to the tweets data. As these data have high-velocity and high-volume characteristics, it is obvious that distributed and parallel technologies should be utilized to support the analytics application. Among the available big data technologies, Hadoop MapReduce, Apache Spark, and Apache Storm have been widely exploited as data processing platforms for this sentiment analysis work. According to overall comparison between these

platforms, Apache Storm comes as the top recommendation to handle streaming data processing in fast amount of time which provides flexibility to integrate with deep learning approach for sentiment classification. Ultimately, this research has intensively evaluate imperative aspects that should be considered before performing sentiment analysis on streaming twitter data, the trade-offs between one approach over the others, and user-specific recommendations.

VII. Future works

Described approaches on Twitter sentiment analysis so far only consider features from the tweet. However, some valuable information might lie beyond the text itself. For example, user profile (gender, following relations, social-network structure, etc.) and Twitter's specific feature (conversations, replies, etc.). Also, recent trend in deep learning for NLP task shows the implementation Transformer outperform previous systems developed based on LSTM or CNN. Thus, the gold standard of doing this task might shift soon.

On the other hand, the utilization of Apache Storm will be more likely to increase as it has true streaming processing capability and the lowest latency. As one of the most widely used distributed stream processing engines, it would revamp the sentiment analysis works on streaming data and fill the gap that cannot be addressed by Hadoop MapReduce all this time by enabling parallel computation on streams and combining the scalability of distributed processing.

References

Apache Software Foundation. (n.d.). *Apache Storm*. Retrieved May 23, 2019, from https://storm.apache.org/

Apache Software Foundation. (n.d.). *MapReduce Tutorial*. Retrieved May 23, 2019, from http://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

Apache Software Foundation. (n.d.). *Spark Overview*. Retrieved May 23, 2019, from https://spark.apache.org/docs/latest/

Aziz, K., Zaidouni, D., & Bellafkih, M. 2018. *Real-Time Data Analysis Using Spark and Hadoop*.

Balahur, A. (2013). Sentiment analysis in social media texts. In *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 120-128).

Bifet, A., & Frank, E. (2010, October). Sentiment knowledge discovery in twitter streaming data. In *International conference on discovery science* (pp. 1-15). Springer, Berlin, Heidelberg.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Chatzakou, D., Kourtellis, N., Blackburn, J., Cristofaro, E. D., Stringhini, G., & Vakali, A. (2017). Detecting aggressors and bullies on Twitter. In WWW (*Companion Volume*) (pp. 767-768). ACM

Chikersal, P., Poria, S., Cambria, E., Gelbukh, A., & Siong, C. E. (2015, April). Modelling public sentiment in Twitter: using linguistic patterns to enhance supervised learning. In International Conference on Intelligent Text Processing and Computational Linguistics (pp. 49-65). Springer, Cham.

Chintapalli, S., Dagit, D., Evans, B., Farivar, R., Graves, T., Holderbaugh, M., Liu, Z., Nusbaum, K., Patil, K., Peng, B.J., & Poulosky, P. 2016. Benchmarking Streaming Computation Engines: Storm, Flink and Spark Streaming. *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 1789 - 1792. Chicago, USA: IEEE.

Cliche, M. (2017). BB_twtr at SemEval-2017 task 4: twitter sentiment analysis with CNNs and LSTMs. *arXiv preprint arXiv:1704.06125*.

DataFlair. (2017). *How Apache Spark Works – Run-time Spark Architecture*. Retrieved May 23, 2019, from https://data-flair.training/blogs/how-apache-spark-works/

Gaber, M. M., Zaslavsky, A., & Krishnaswamy, S. (2005). Mining data streams: a review. ACM Sigmod Record, 34(2), 18-26.

Go, A., Bhayani, R. and Huang, L., 2009. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(2009), p.12.

Hamad, R. A., Alqahtani, S. M., & Torres, M. T. (2017, July). Emotion and polarity prediction from Twitter. In 2017 Computing Conference (pp. 297-306). IEEE.

Harazika, A.V., Ram, G.J., & Jain, E. 2017. Performance Comparison of Hadoop and Spark Engine. *International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud).*

Hogenboom, A., Bal, D., Frasincar, F., Bal, M., de Jong, F., & Kaymak, U. (2013, March). Exploiting emoticons in sentiment analysis. In *Proceedings of the 28th annual ACM symposium on applied computing*(pp. 703-710). ACM.

Hsueh, P. Y., Melville, P., & Sindhwani, V. (2009, June). Data quality from crowdsourcing: a study of annotation selection criteria. In Proceedings of the NAACL HLT 2009 workshop on active learning for natural language processing (pp. 27-35). Association for Computational Linguistics.

Kanavos, A., Nodarakis, N., Sioutas, S., Tsakalidis, A., Tsolis, D., & Tzimas, G. 2017. *Large Scale Implementations for Twitter Sentiment Classification*. In B.Carpentieri (Ed.). *Algorithms*, 10-33.

Karanasou, M., Ampla, A., Doulkeridis, C., & Halkidi, M. 2016. Scalable and Real-time Sentiment Analysis of Twitter Data. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 944-951. Barcelona, Spain: IEEE.

Kaushik, C., & Mishra, A. (2014). A scalable, lexicon based technique for sentiment analysis. *arXiv preprint arXiv:1410.2265*.

Khumoyun, A. Cui, Y., & Lee, H. 2016. Real time information classification in Twitter using Storm. *Full Paper Proceeding ECBA -2016*, 49 (6), 1-4.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Murthy, J.S., Siddesh, G.M., & Srinivasa, K.G. 2018. A Distributed Framework for Real-Time Twitter Sentiment Analysis and Visualization. In P.K.Sa, S.Bakshi, I.K. Hatzilygeroudis, M.N. Sahoo. *Recent Findings in Intelligent Computing Techniques*, 3, 55-61. Singapore: Springer.

Neethu, M. S., & Rajasree, R. (2013, July). Sentiment analysis in twitter using machine learning techniques. In 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT) (pp. 1-5). IEEE.

Neri, F., Aliprandi, C., Capeci, F., Cuadros, M., & By, T. (2012, August). Sentiment analysis on social media. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 919-926). IEEE.

Parven, H., & Pandey, S. 2016. Sentiment Analysis on Twitter Data-set using Naïve Bayes Algorithm. *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)* (pp 416-419). Bangalore, India: IEEE. Retrieved May 20, 2019, from https://ieeexplore-ieee-org.ezp.lib.unimelb.edu.au/abstract/document/7912034.

Rajaraman, A.; Ullman, J. D. (2011). "Data Mining" . *Mining of Massive Datasets*. pp. 1–17.

Ranjan, R. (2014). Streaming big data processing in datacenter clouds. *IEEE Cloud Computing*, *1*(1), 78-83.

Read, J. (2005, June). Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL student research workshop*(pp. 43-48). Association for Computational Linguistics.

Rodrigues, A.P., & Chiplunkar, N.N. 2019. A new big data approach for topic classification and sentiment analysis of Twitter data. *Evolutionary Intelligence* (pp 1-11). Springer Berlin Heidelberg Retrieved May 20, 2019, from https://doi.org/10.1007/s12065-019-00236-3.

Rosenthal, S., Farra, N., & Nakov, P. (2017, August). SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)* (pp. 502-518).

Rosenthal, S., Nakov, P., Kiritchenko, S., Mohammad, S., Ritter, A., & Stoyanov, V. (2015). Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)* (pp. 451-463).

Saif, H., Fernandez, M., He, Y., & Alani, H. (2013). Evaluation datasets for Twitter sentiment analysis: a survey and a new dataset, the STS-Gold.

Severyn, A., & Moschitti, A. (2015, August). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 959-962). ACM.

Severyn, A., & Moschitti, A. (2015, August). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 959-962). ACM.

Sheela, L.J. 2016. A Review of Sentiment Analysis in Twitter Data Using Hadoop. *International Journal of Database Theory and Application,* 9 (1), 77-86. Retrieved May 20, 2019, from https://www.researchgate.net/publication/299423553.

Tang, D., Qin, B., Feng, X., & Liu, T. (2015). Effective LSTMs for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100*.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Vol. 1, pp. 1555-1565).

Taylor, A., Marcus, M., & Santorini, B. (2003). The Penn treebank: an overview. In *Treebanks*(pp. 5-22). Springer, Dordrecht.

Toshniwal, A. Taneja, S. Shukla, A., Ramasamy, K., Patel, J.M., Kulkarni, S., Et al. 2014. Storm at Twitter. *SIGMOD '14 Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 147-156. New York, USA: ACM.

Twitter.com. How to post links in a Tweet. (2019). Retrieved from https://help.twitter.com/en/using-twitter/how-to-tweet-a-link. Accessed 24 April 2019'

Wiegand, M., Balahur, A., Roth, B., Klakow, D., & Montoyo, A. (2010). A survey on the role of negation in sentiment analysis. In *Proceedings of the workshop on negation and speculation in natural language processing*(pp. 60-68).

Yadranjiaghdam, B., Yasrobi, S., Tabrizi, N. 2017. *Developing a Real-time Data Analytics Framework for Twitter Streaming Data*. 2017 IEEE International Congress on Big Data (BigData Congress). Honolulu, USA: IEEE.

Yan, B., Yang, Z., Ren, Y., Tan, X., & Liu, E. 2017. Microblog Sentiment Classification using Parallel SVM in Apache Spark. *2017 IEEE International Congress on Big Data (BigData Congress)*, 282-288. Honolulu, USA: IEEE.

Yin, Y., Song, Y., & Zhang, M. (2017, August). Nnembs at semeval-2017 task 4: Neural twitter sentiment classification: a simple ensemble method with different embeddings. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 621-625).

Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), e1253.