International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 4, Issue 2, pp 31-40, February-2020

Arabic Sentiment Analysis using Apache Spark

Mohamed A. Ahmed

Computer Science Department, College of Computer and Information Systems, Umm Al-Qura University,
Makkah Al Mukarramah, Kingdom of Saudi Arabia

Abstract - **People express their feelings and emotions on social media platforms including Twitter. Twitter blogging system currently includes huge Arabic user-generated contents. These Arabic data are rapidly increasing in volume. Arabic language has special characteristics, syntax, grammar and morphological rules. In this research, we are investigating the development of a big data system to analyze the emotion of Arabic-related contents. This application could be useful for many monitoring, marketing, recommendation and decision support systems. We make use of machine learning algorithm to achieve this goal. We particularly incorporate and compare the supervised-based Naïve-Bayes and Logistic Regression algorithms as the main machine learning engines that processes the Arabic-language tweets in order to classify them if the new tweets as either positive, negative or neutral. Before applying the Naïve-Bayes and logistic regression processing algorithms, we apply a pre-processing phase of the input tweets to generate numerical features instead of input text and emojis to make it suitable for the processing algorithms. We have developed a customized pre-processing pipeline that includes several Arabic NLP (ANLP) preparation steps and that suits Arabic language and the contents of real-life tweets. In addition, these two machine learning algorithms have several hyper-parameters that could affect the performance and accuracy of the algorithm. Therefore, we have utilized cross-validation techniques to evaluate and detect the best possible hyper-parameters combination that results in the best accuracy results of classification outputs. Experiments show promising results using the designed system. We will present some experiment results that show the accuracy of the system against real-life Arabic tweets data in terms of f1-score, weighted precision and weighted recall evaluation metrics.**

Keywords: Big Data, Machine Learning, Arabic NLP, Emojis, Cross-Validation.

I. Introduction

Sentiment analysis, sometimes called opinion mining, mainly exists to analyze and classify emotions and opinions of end-users input in social media. These inputs are in the form of informal textual messages. However, most of the research in this field, while utilizing big data techniques and optimizations, were performed for the English language. In our research, we target Arabic language. Meanwhile, sentiment analysis usually involves text processing, Natural Language Processing (NLP) and computational linguistics techniques.

We are utilizing a big data framework called Apache Spark to perform sentiment analysis for Arabic tweets on the Twitter platform. In 2017, number of Twitter Arab users was 11.1 million users [1]. In only two years, the number of tweets has increased by 59% up to 850 million tweets each month. This represents a huge volume of streaming contents that would require a robust big data platform to process. Therefore, we recommend utilizing a powerful big data framework such as Apache Spark [2] and its ecosystem to acquire, process and store this huge volume of useful Arabic contents. Arabic Language is an RTL (Right-To-Left) language. Its alphabet consists of 28 basic letters. Arabic orthographic system uses small diacritical vowel markings to represent the three short vowels (pronounced like the letters a, i, o). Arabic sound phonetics include total of 13 different diacritics. When written, these markings are placed either above or below the letter to indicate the phonetic information associated with each letter to clarify meaning of the word. For example, the un-diacritized word ع لم in Arabic could mean several different things if diacritical vowel markers (harakat) are used. Although this word could mean either: َ عَلَم (i.e. flag) or عَلَّمَ (i.e. taught) or عِلْ م (i.e. science), which may sound confusing, the Arabic natives successfully disambiguate the meaning through understanding the context surrounding the word. In real-world situations, most Modern Standard Arabic (MSA) documents are typically written without diacritics. The main difficulty when performing sentiment analysis in Arabic social media lies in the fact that communication in the social media context is carried out using vernacular or dialectical Arabic rather than the more formal Modern Standard Arabic (MSA) [3]. The vocabulary of dialectical Arabic is different from that of MSA. There are different Arabic dialects in the different Arab countries such as Egyptian ammiyah, Gulf khaleeji, Levantine shaami and North African maghribi. In addition, sentences in social media are much more random, unstructured and noisier than MSA rules. This makes the task of automated parsing and understanding Arabic contents that are available in social media is a major challenge [4].

Meanwhile, machine learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying instead on historical patterns and inference. It is seen as a subset of artificial intelligence algorithms. In this paper, we especially use supervised machine learning algorithms. This type of algorithms builds a mathematical model based on sample labeled observations, known as "training data", in order to make predictions and decisions about future unseen samples without being explicitly programmed to perform the task. In our work, we mainly utilize a reliable machine algorithm namely the Logistic Regression [5] algorithm. We will also study the effect of changing some of the hyper parameters of this algorithm on the accuracy results.

Apache Spark is an open source computing framework and it is proven to be 100 times faster than Map Reduce [6]. Apache Spark is an alternative form of distributed data processing for both batch and streaming processing that could utilize a computer cluster which consists of hundreds and thousands of server nodes. Apache Spark is a high-level Java-based API solution that is mainly built using Scala programming language which is a derived programming language from the popular Java programming language. Spark platform could run Spark-based distributed applications. Each Spark application could launch multiple jobs [7]. Each job could be executed through multiple stages. Each stage involves many distributed tasks which could run in parallel throughout the cluster nodes. Apache Spark relies on a certain type of data structure called Resilient Distributed Dataset (RDD). RDD is the basic abstraction dataset in Spark that is distributed, fault tolerant and scalable if we add more hardware (CPU, memory and storage). Each RDD is automatically segmented into partitions which are distributed on the cluster nodes. Each partition could be processed using a separate task. RDD is considered resilient because Apache Spark monitors when any node fails and thus automatically restart its jobs and reconstruct specific partitions of RDDs as needed. A main advantage of using RDDs in Apache Spark over other big data frameworks is that RDDs are cached and processed in-memory and not kept in disk (unless there is not enough runtime space in main memory). That is why Apache spark could be 100 times faster than Hadoop platform. RDDs are immutable and manipulated through a diverse set of parallel transformations and actions. In Spark, a partition is the atom of data. Each partition contains a collection of records. Apache Spark provides lazy execution of transformation operations (i.e. certain operations are not actually realized until data in needed for an execution action). Spark supports two types of operations namely; transformations and actions. Transformation operators (such as: map(), filter(), join(), …etc) are lazy and are not immediately executed or causes the creation of a new RDD out of the old RDD. A physical execution plan is usually generated after performing a series of automatic optimization steps inside the Job Scheduler. In addition, Spark is considered polyglot because it supports multiple programming languages such as Java, Scala, Python, R and even run queries using SQL language. Spark software also provides interactive shells for Scala and Python to execute commands and statements step-by-step.

The remainder of this paper is organized as following: we will present some literature review in the domain of sentiment analysis especially for Arabic language contents in section 2. Then in section 3, we describe the overall big data architecture that we have built to run our Arabic-based sentiment analysis application. We will explain some details about the Arabic Natural Language Processing (ANLP) steps which are used for pre-processing of Arabic text and emojis within the tweets and we will describe how we implemented these steps using the Apache Spark core and higher-level ML API libraries. In section 4, we will provide the results of our system evaluation as well as a discussion and some insights based on various experiments results in order to elect the best machine learning model and hyper parameters tuning. In section 5, we will provide concluding remarks along with our future directions.

II. Literature Review

The research work in [8] has presented a complete framework and its components for mining Arabic Tweets in order to evaluate customer feedback about a certain Telecom company in the Kingdom of Saudi Arabia. The authors have used certain preprocessing steps such as: normalization, generation of n-gram features and certain Arabic-based contextual rules to improve the accuracy of the classification results of the system. The research work in in [9] has applied semantic techniques as well as Arabic linguistic methods to raw Arabic tweets. They applied semantic classification steps after the preprocessing of the tweets. In addition, the authors have combined English and Arabic language lexicons to calculate the sentiment polarity of the Arabic tweets.

The authors in [10] used a deep learning modeling [11] approach to detect the sentiment of Arabic tweets. Their approach relies on the utilization of pre-trained word vector representation and not having any feature engineering preprocessing steps. The reasoning engine includes an ensemble model that combines both Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) [12] models. LSTM is a special type of Recurrent Neural Networks

(RNN) which can learn long-term dependencies. LSTM was developed to prevent the long-term dependency problem that exists in standard RNNs. The results show good improvements in the accuracy and f1-score measurements over some other flat approaches.

The research in [13] utilized supervised machine learning to resolve the issues of aspect-based sentiment analysis (ABSA) for the domain of Arabic Hotels reviews. An aspect [14] represents a certain component or attribute within an object. A tweet may talk about several independent aspects with different sentiments. The study compared Support Vector Machines (SVM) algorithm against deep Recurrent Neural Network (RNN) algorithm. The training phase made use of lexical, word, morphological, syntactic and semantic features of the text. Evaluation results showed that SVM approach was more accurate than deep RNN approach. Nevertheless, the RNN approach resulted in better timing performance than SVM approach.

The authors in [15] have utilized Arabic WordNet (AWN) [16] to create Arabic sentiment lexicon using a semi-supervised learning method. The researchers have started with short list of seed lexicon that consisted of only 10 positive and 10 negative words. They used this short list to assign polarity scores to different AWN node objects. Using the AWN relationships, these scores were iteratively propagated throughout the AWN node objects. The final lexicon contained 23,481 nodes. The authors then manually trimmed this lexicon into 7,576 entries. The evaluation experiments proved that the Arabic sentiment analysis accuracy has improved using this new AWN lexicon.

As described, although these research methods performed sentiment analysis for Arabic contents but they didn’t run and were not optimized for big data platforms.

III. Big Data Architecture for Arabic Sentiment Analysis

In this section, we will describe the high-level Apache Spark architecture and its components that we will utilize in our system. We targeted the PySpark API because we adopted the Python programming language in this research. We will then provide some details about the Spark ML (Machine Learning) API library which is available within the Apache Spark ecosystem and could run on top of the Spark Core layer.

3.1 Apache Spark Big Data Architecture

Utilization of big data frameworks could be a great benefit for the domain of machine learning problems because they naturally require a lot of processing power, time and storage resources especially if we are dealing with large volumes of data. We will use Apache Spark framework for processing large scale datasets on a cluster of computing nodes. Figure 1 illustrates the high-level Apache Spark Architecture and its main components and interactions. Our application will have a driver program which initiates and manages the whole Spark application life cycle. When the driver program needs resources to run jobs or tasks, it first asks the cluster manager process for some resources. The cluster manager is responsible of scheduling and allocating resources across the nodes of the cluster. The cluster manager assigns the required resources such as number of executors, number of cores per driver and per executor, memory space for the driver and executors. The worker (or slave) nodes are allocated throughout the cluster to instantiate and run executor processes in order to run distributed job tasks. The executors will run for the whole life-time of the Spark application to execute these tasks in different Java Virtual Machines (JVMs) and using multiple threads. This guarantees that there will be isolation among the different Spark applications running on the cluster. The executors, shown in figure 1, should be able to communicate among themselves to perform shuffling operations when required. These executors should also be able to contact the driver program. The driver contains a low-level task scheduler that distributes the physical tasks to the executors on the worker in the form of task sets. This means that the actual machine learning algorithms, that we will implement, will run as different distributed jobs, stages and tasks on the worker nodes in a distributed and parallel manner. At the end, the executor nodes will return their results to the driver process which usually aggregates these partial results to construct the final output results.

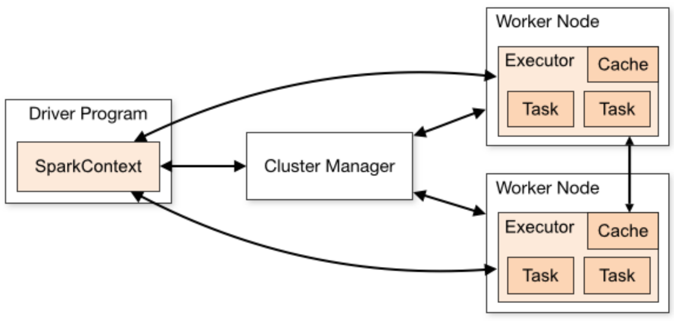


Figure 1: Apache Spark high-level architecture and main components [17]

3.2 Spark Machine Learning Library

In order to fully utilize machine learning algorithms, these algorithms should be compatible with the components of the big data platform, its data structures and logic flow. Thus, we will utilize the Machine Learning (ML) API [18] which is a high-level and scalable API that is offered by the Apache Spark ecosystem. Spark ML API provides a set of useful tools related to different aspects of machine learning. These tools include tools for performing popular classification, clustering, collaborative filtering and dimensionality reduction algorithms. In this research, we will mainly use the Naïve Bayes and Logistic Regression algorithms.

An important construct that is available in Spark ML API is Pipelines. Pipeline constructs, in Apache Spark, is borrowed from the Scikit-Learn [19] Machine Language API. Spark ML pipeline is a uniform high-level API built on top of Spark Data Frames API. Pipelines module offers mechanisms to construct, evaluate and tune pipelines. Spark ML also allows us to save and load machine learning algorithms, trained models and pipelines so that we could reuse the results of previous steps to generate new results without the need to start the whole process from scratch.

3.3 Arabic Text Analytics

The life-time scenario of sentiment analysis of Arabic tweets is presented in figure 2. It shows the main steps for both the training and runtime phases. In training phase, we attempt to generate an optimal machine learning model out of training labeled data samples. In our experiments, we train the machine learning algorithms using an available training dataset and compare the performance and accuracy of these models using a special evaluation criterion (e.g. using the f1-score metric) on a hold-out labeled testing dataset. In section 4, we will provide more details about building these models using Apache Spark big data architecture and how we could compare different ML algorithms.

In addition, these algorithms utilize hyper parameters which control the behavior of the algorithms in terms of bias and variance. We aim to attain the lowest possible bias and variance for these models. The chosen model should avoid both under fitting and over fitting to the training datasets as much as possible. Thus, a separate process of tuning the hyper parameters of the chosen model is performed. We will utilize the cross-validation approach [20] for the tuning process till we reach the optimal values for the hyper parameters. Once we find an optimal ML model, we will utilize this model at runtime to predict the label of future never-seen Arabic tweets.

As we have mentioned, we are analyzing Arabic tweets to get the sentiment value of each tweet; positive, negative or neutral. Unfortunately, machine learning algorithms cannot handle input textual data as is. Therefore, we must perform some pre-processing, cleansing and data wrangling steps of the input text for each observation (tweet) before we would be able to run the various machine learning algorithms. The objective of this phase is to generate a dataset in an optimal format. This result format consists of derived numerical features in a vector form. As described next, we will apply some generic pre-processing rules as well as other specific Arabic-language rules. The derived dataset is then segmented into two separate datasets. The larger dataset is usually used for the training phase using the supervised machine learning algorithms (Naïve Bayes and logistic regression).

In Apache Spark ML API terminology, we are performing a "fit" operation for the ML algorithm using the training dataset. The smaller remaining dataset is preserved for the testing phase of each model. In Apache Spark ML API terminology, we are doing a "predict" operation using the testing dataset where we could obtain several ML evaluation metrics in order to compare the different ML models.

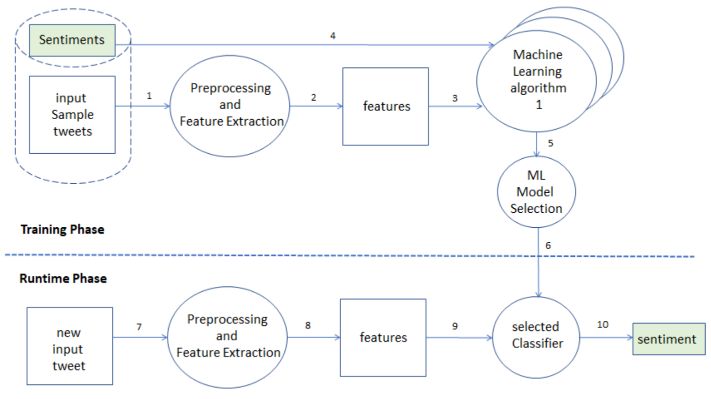


Figure 2: Lifetime scenario for supervised machine learning

3.3.1 Pre-processing Phase

The most difficult task is possibly the cleansing and preprocessing of the raw text data. We attempt to get rid of noisy and non-useful tokens within each tweet. We then utilize the results to perform feature extraction techniques to represent each observation (tweet) as a vector of numerical features so that it could be ready for the machine learning training and testing processes. We utilize some rules that are related to Arabic morphological and syntax laws in order to refine the input text stream and generate the most useful normalized tokens. Spark ML API facilitates the whole process of performing these tasks as a one streamlined operation through the pipeline construct. We could summarize the steps of the cleansing and preprocessing phase as following:

3.3.1.1 Remove the elongation and diacritics letters

Some Arabic text editors could add extra features to the Arabic text such as elongation ("Tatweel") and diactrics ("Tashkeel") letters. "Tatweel" [21] is a dual-joint letter with the Unicode value of U+0640 which looks like a hyphen letter. It is used to connect two joined letters to give a wider shape to the connection between these two letters. Meanwhile "Tashkeel" [22] means utilization of some diacritical marks after each letter to affect the way that we pronounce this letter according to some syntax and grammatical rules. Both "Tashkeel" and "Tatweel" could result in different shapes for words of the same origin. Thus, to perform some sort of normalization of these various formats of similar words, we get rid of both "Tatweel" and "Tashkeel" effects during the pre-processing steps.

3.3.1.2 Emoji Processing

Tweets normally include a lot of Emoji characters. Emojis are icons and smileys that could be placed inline within the text of the tweets. They express some emotional opinions about the contents of the tweet. Nowadays, the use of Emojis is pervasive in all sources of social media. Examples of Emojis are: 😀 , 😪 , ✌ , …etc. They are sources of very useful features during the feature extraction steps. Therefore, we keep them for next processing steps. Each Emoji has a Unicode representation [23].

As presented, all Emojis lie within specific Unicode ranges. Thus, to extract Emojis as separate features, we apply regular expressions (Regex) on the input text to identify Emojis within the text and add white space characters around each Emoji within the text to deal with the possibility of having multiple consecutive Emojis within the text. Then, we handle each Emoji as a separate token during the following tokenization step.

3.3.1.3 Arabic Text Cleaning and Normalization

At this step, the text of the tweet is ready for some cleaning and normalization. We apply text cleanup and normalization steps for each Arabic tweet. In this task, we execute a series of regular expressions to remove unnecessary tokens in preparation for the feature extraction and sentiment analysis processing afterwards. For example, we remove URLs, email addresses, RT, cc, hashtags, mentions and numbers.

Then, we run several steps to tokenize and normalize the tweet into multiple tokens. For the tokenization task, we make use of the Word Punct tokenizer from the NLTK [24] natural language processing (NLP) library. This step generates a group of raw tokens that are extracted from the tweet as an array of strings and Emoji tokens.

These set of tokens are introduced next to the step of removing Arabic and English stop words. These stop words are certain common Arabic and English words that would not influence the identification of the sentiment of the tweet. Examples of Arabic stop words are: من , في , هذا , الي , …etc. Examples of English stop words are: in, the, a, to, …etc.

Then, we perform a final step using the available tokens of each tweet. This step involves the normalization of the input tokens through a stemming operation. Using this process, we attempt to deduce the same stem for different morphological forms of the same word. We utilize an Arabic stemmer called ISRI Stemmer [25] from the NLTK library API. During features extraction, we utilize the stem of each token instead of the original token.

3.3.1.4 Building an Apache Spark ML Custom Transformer

In order to apply the different steps of text preprocessing and normalization, we have built a custom Spark ML transformer. This custom transformer becomes a part of the ML pipelines that we designed for all our experiments. In Apache Spark ML terminology, there are two main types of operators: transformers and estimators.

Examples of estimators are logistic regression, Naïve Bayes, SVM and decision trees machine learning algorithms. Both transformers and estimators are of a type called as Pipeline Stage in Apache Spark ML library. Thus, both transformers and estimators could be parts of a spark ML pipeline in any sequence and grouping. In our design, we have built the pipeline in a linear form where the output of each step represents the input to the next step within the pipeline till we receive the final output of the pipeline.

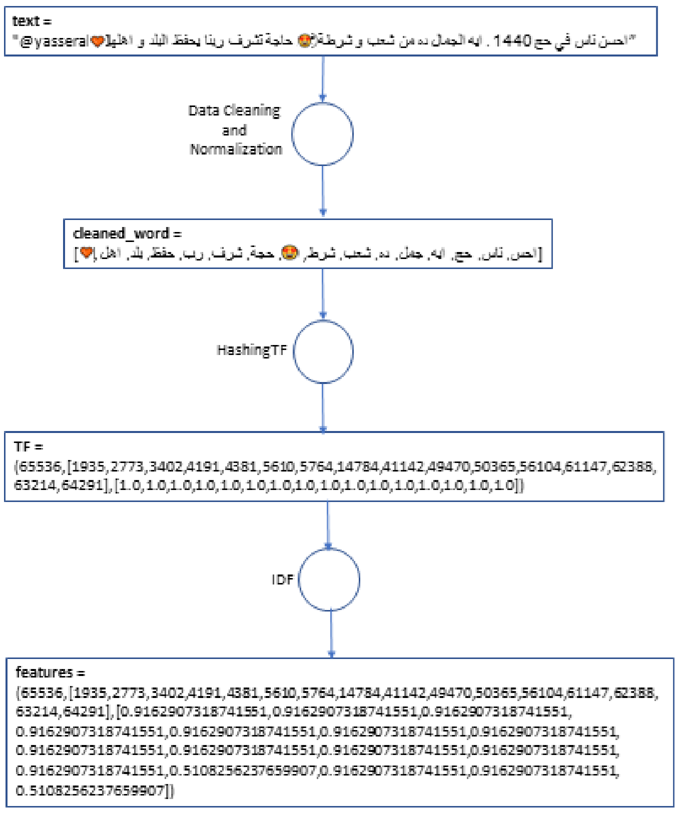
3.3.2 Features Extraction

The next logical step after all previous features engineering and preprocessing is to perform features extraction using the result tokens. We mainly incorporate TF.IDF (Term Frequency Inverse Document Frequency) approach to extract the features vector out of the input bag of words (BOW). TF.IDF involves two parts that are mathematically multiplied: TF and IDF processes. We chose the total size of terms to be 2^16 (i.e. 65,536 features) which means that the tweets-features matrix would be a very sparse matrix.

After these previous steps, we obtain a numerical vector of features which is usually sparse along with the given labels (classes) in a numerical format that is ready to be fed to any machine learning algorithm that we use in our experiments. Figure 3 depicts an example of the feature extraction process that generates numerical features out of an Arabic tweet example using TF.IDF and the hashing trick [26].

ISSN (online): 2581-3048

Volume 4, Issue 2, pp 31-40, February-2020



```
text =
"احسن ناس في حج 1440 . ايه الجمال ده من شعب و شرطة😍 حاجة تشرف ربنا يحفظ البلد و اهليها❤ @yasseral"

cleaned_word =
[احسن, ناس, حج, ايه, جمل, ده, شعب, شرط, 😍, حجة, شرف, رب, حفظ, بلد, اهل, ❤]

TF =
(65536,[1935,2773,3402,4191,4381,5610,5764,14784,41142,49470,50365,56104,61147,62388,
63214,64291],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])

features =
(65536,[1935,2773,3402,4191,4381,5610,5764,14784,41142,49470,50365,56104,61147,62388,
63214,64291],[0.9162907318741551,0.9162907318741551,0.9162907318741551,
0.9162907318741551,0.9162907318741551,0.9162907318741551,0.9162907318741551,
0.9162907318741551,0.9162907318741551,0.9162907318741551,0.9162907318741551,
0.9162907318741551,0.5108256237659907,0.9162907318741551,0.9162907318741551,
0.5108256237659907])
```

Figure 3: An example of feature extraction process of an Arabic tweet

3.3.3 Training Phase

We are implementing and comparing the performance of certain supervised classification algorithms while running on the Apache Spark platform. We will make use of available implementations of machine learning algorithms which are available in the Spark machine learning library. These implementations are optimized to run in a distributed and parallel manner over a cluster of computing and storage resources. These algorithms are: Naïve Bayes and Logistic Regression. We partition the input labeled dataset into two parts: training dataset and testing dataset. The training dataset will be used during the training phase. The testing dataset will be utilized later by the trained model so that we could compare the prediction labels versus the real input labels. We utilize a separate testing dataset to mitigate the problem of model over fitting to the training dataset.

IV. Experiments Results and Discussion

In this section, we will present the empirical results of the different experiments that we have conducted to perform sentiments analysis of Arabic tweets. As a proof of concept, we have built a cluster prototype that consists of commodity PC computers. The cluster test bed consists of one master node and two worker nodes. The architecture looks like figure 1. Each machine has an i7 64-bit quad-core 3.4 GHz processor, 8 GB RAM and 500 GB hard drive. The cluster runs Windows 10 operating system, Java JDK 1.8.0 and Python 3.7. The cluster was built using Apache Spark 2.4.4 big data platform which runs in a standalone [7] cluster mode. Upon developing our Spark distributed application in Python, we launch it using the spark-submit command utility. The storage system is built using the Hadoop Distributed File System (HDFS) [27]. HDFS supports fault tolerance through replication. We have chosen a replication factor of the value 3 (i.e. the blocks of all stored files are replicated 3 times throughout the cluster). Spark utilizes HDFS block distribution to support data locality of computation. This means that Spark attempts as much as possible to run application's tasks close to the data blocks of the stored files. The input labeled tweet files which are used for training and testing are CSV (Comma Separated Values) files.

We have utilized a total dataset of 9,730 Arabic tweets that come from the ASTD dataset [28]. We randomly split this dataset into two datasets; training and testing. The training dataset constitutes 66% of the total dataset. Meanwhile, the testing dataset contains the remaining 34% of the total dataset with totally distinct observations that are never seen by the various ML algorithms during training. Both training and testing datasets are balanced datasets with almost same number of samples for the three classes; positive, negative and neutral.

The problem is considered as an optimization problem that involves multiple machine learning algorithms and several hyper parameters per each algorithm. Thus, we made use of cross-validation techniques to train the system using different machine learning algorithms and several hyper parameters settings for each algorithm. We utilized Apache Spark ML API to construct different pipelines which included cleansing, preprocessing, stemming and feature extraction steps. Each pipeline is built for a different machine learning algorithm of the two machine learning algorithms; Naïve Bayes and Logistic Regression. We adopted K-fold cross-validation. In K-fold cross-validation, the input training dataset is randomly divided into K number of folds. We have chosen the K value of 5.

For comparing the different models, we need to use an evaluation process. Thus, we calculate the weighted average for the different measures; recall, accuracy and f1-score based on the calculations of the metric for each individual class of the tress classes; positive, negative and neutral. The weighted average is then calculated as the average while being weighted by the class support (i.e. the number of true samples for each class). This will also consider the possibility of class imbalance within the dataset. We chose the f1-score metric to be the decisive measurement among the different models because f1-Score is the harmonic mean of both precision and recall metrics. The model with the highest f1-score value represents the best model. Equations 1 to 7 illustrate how we calculate the different intermediate and final evaluation metrics.

$$\text{Accuracy} = [TP + TN] / [TP + FP + TN + FN] \ \dots \ \text{(equation 1)}$$

$$\text{Recall}_i = TP_i / [TP_i + FN_i] \ \dots \ \text{(equation 2)}$$

$$\text{Weighted Recall} = \sum_{i=0}^{i=N-1} [\text{Recall}_i * \text{Support}_i] / \sum_{i=0}^{i=N-1} \text{Support}_i \ .. \ \text{(equation 3)}$$

$$\text{Precision}_i = TP_i / (TP_i + FP_i) \ \dots. \ \text{(equation 4)}$$

$$\text{Weighted Precision} = \sum_{i=0}^{i=N-1} [\text{Precision}_i * \text{Support}_i] / \sum_{i=0}^{i=N-1} \text{Support}_i .. \ \text{(equation 5)}$$

$$\text{F1-score}_i = 2 * [(\text{Precision}_i * \text{Recall}_i) / (\text{Precision}_i + \text{Recall}_i)] .. \ \text{(equation 6)}$$

$$\text{F1-score} = \sum_{i=0}^{i=N-1} [\text{F1-score}_i * \text{Support}_i] / \sum_{i=0}^{i=N-1} \text{Support}_i \ \dots\dots \ \text{(equation 7)}$$

Where:

TP means True Positives, FP means False Positives,
TN means True Negatives, FN means False Negatives,
N = number of class labels = 3, i is the index of class i where i = 0, 1, …., N-1
Support_i means the number of true samples for class i,

Table 1 lists all the different hyper parameters values that were used for training and testing the two machine learning algorithms; Naïve Bayes and Logistic Regression.

TABLE 1
Hyper parameter values for the two machine learning algorithms

ML algorithm	Hyperparameter	Options
Naïve Bayes	smoothing	0.0, 0.5, 1.0
Logistic Regression	maxIter	10, 15, 20
	regParam	0.0, 0.05, 0.1

Where,

- *smoothing* means the Laplace smoothing parameter when we have unknown words in the testing dataset,
- *maxIter* means maximum number of iterations,
- *regParam* means regularization parameter

Figure 4 depicts a comparison of the performance results of the different trained ML models using the f1-score metric. As shown, the logistic regression algorithm with hyper parameters *maxIter* = 20 and *regParam* = 0.05 slightly leads to the best performance results for the given training and datasets. The best associated *f1-score* result is 72.89 % which is a promising result for our problem. Figure 5 shows similar comparisons in terms of *weighted precision* metric while figure 6 presents the comparisons in terms of *weighted recall* metric.

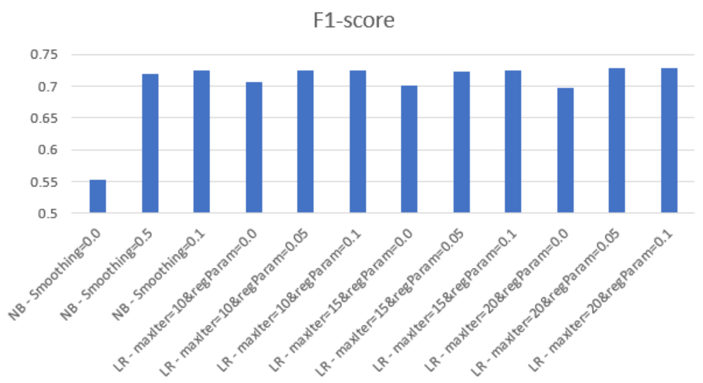


Figure 4: Comparison in terms of *f1-score* metric

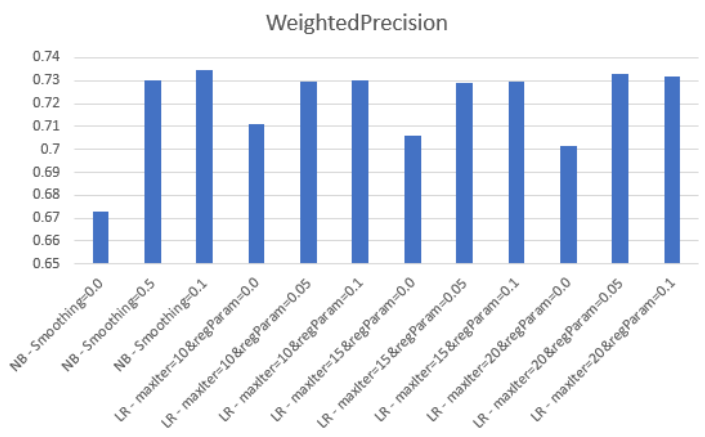


Figure 5: Comparison in terms of *weighted precision* metric

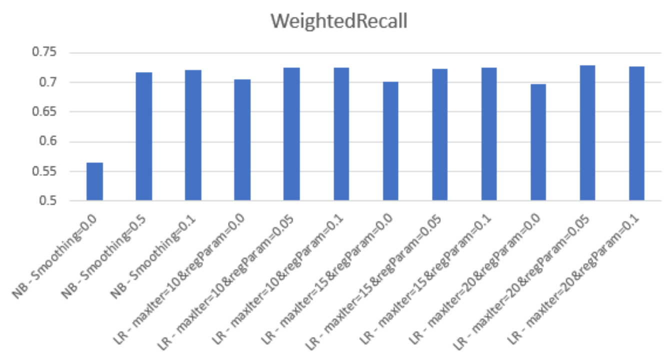


Figure 6: Comparison in terms of *weighted recall* metric

However, we have noticed that logistic regression is more computationally expensive. One justification for that is that by looking into the jobs and tasks history, we have realized that logistic regression results in much greater number of jobs and stages [7] than Naïve Bayes for this multiclass training problem.

These results are calculated for a given training and testing datasets and for a certain list of suggested hyper parameters. It is possible that we find a different optimal model for a different training and testing datasets and hyper parameter grid search values.

We think that there is a chance to improve the accuracy results in our future work. First, increasing the size of training dataset with no or few outliers could result in a better ML model. Another way of improvement is to use a different Arabic stemmer algorithm than ISRI stemmer. A better stemmer would take into consideration the peculiar features of social media contents with respect to the different dialects and non MSA (Modern Standard Arabic) rules. A better stemmer may include a domain-based Arabic lexicon that is especially created for social media data sources. A third area of improvement is adding a more comprehensive feature engineering steps that would results in better derived features set such as: the number of Emojis, the number of special characters (e.g. exclamation and question letters), user's historical influence level, number of re tweets, number of comments, number of likes, …etc. The features set could also be augmented with some word2vec features in addition to the Bag Of Words (BOW) features. We expect that this would add extra power to the ML model and thus lead to better accuracy results.

Another future direction is to investigate other ML algorithms such as SVM, XGBoost, Random Forest, Ensembles and Deep Learning.

V. Conclusion and Future Work

In this paper, we have presented a distributed big data system that performs sentiment analysis for Arabic tweets. To make use of the system, we run a training phase that learns from labeled Arabic tweets. Arabic tweets don't usually follow the formal Modern Standard Arabic (MSA) rules that are used in official documentation. Therefore, we have first designed a customized ANLP pre-processing pipeline of actions in order to prepare the input unstructured Arabic tweets into a more structured and cleaner data structure that suits the following training, validation and testing phases. For training phase, we have incorporated the Naïve Bayes and Logistic Regression algorithms which are supervised machine learning algorithms that could learn from existing classified Arabic tweets dataset. A validation stage is added to fine-tune the hyper-parameters of the machine learning algorithm using a separate validation dataset through the utilization of cross-validation and K-fold techniques. After the training phase, we have tested the accuracy and performance of the training results using a separate testing dataset. The system is based on Apache Spark big data framework. Apache spark version 2.4.4 and the high-level Spark ML (Machine Learning) software library provide fault-tolerance, powerful caching, near real-time processing and polyglot language support. Apache Spark utilize in-memory caching and several optimization techniques. Therefore, its performance is much better than many other platforms such as Apache Hadoop. The Apache Spark components are responsible for distributing the processing tasks among the different slave nodes. Experiments concluded that the average overall accuracy of the Logistic Regression training algorithm outperformed the Naïve Bayes machine learning algorithm for the given Arabic tweets datasets in terms of f1-score, weighted precision and weighted recall metrics.

In future, we plan to investigate other machine learning algorithms and compare them in terms of accuracy and processing performance. In addition, we think that the pre-processing stage could be improved further by eliminating spam and not-useful input tweets which could result in better accuracy during the training and validation phases.

ACKNOWLEDGMENTS

The author would like to thank the Deanship of Scientific Research at Umm Al-Qura University for the continuous support. This work was supported financially by the Deanship of Scientific Research at Umm Al-Qura University (Grant Code: 15-COM-3-1-0015).

REFERENCES

[1] "Prominent Arabic Social Media", https://www.extradigital.co.uk/articles/arabic/social-media.html, *Last accessed on 15* July 2019.

[2] "Apache Spark - Unified Analytics Engine for Big Data", https://spark.apache.org/, *Last accessed on 15* July 2019.

[3] M. Al-Ayyoub, A. Khamaiseh, Y. Jararweh and M. Al-Kabi, "A comprehensive survey of Arabic sentiment analysis", *Journal of Information Processing & Management*, Volume 56 Issue 2, March 2019, pp. 320-342.

[4] K. Haifa and A. Azmi, "Arabic tweets sentiment analysis – a hybrid scheme", *Journal of Information Science*, Volume 42, 2016, pp. 782 –797.

[5] T. Tjur, "Coefficients of determination in logistic regression models", *American Statistician*, doi:10.1198/tast.2009.08210, 2009, pp. 366–372.

[6] "Apache Hadoop 2.8.0 – MapReduce", https://hadoop.apache.org/docs/r2.8.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html, *Last accessed on* 15 July 2019.

[7] J. Perrin, "Spark in Action", *second edition, book.* March 2018, ISBN: 9781617295522.

[8] M. Heikal, M. Torki and N. El-Makky, "Sentiment Analysis of Arabic Tweets using Deep Learning", *The 4th International Conference on Arabic Computational Linguistics (ACLing 2018), Dubai, United Arab Emirates*, November 17-19 2018, pp. 114–122.

[9] J. Schmidhuber, "Deep learning in neural networks", *Journal of Neural Networks,* Volume 61 Issue C, January 2015, pp. 85-117.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation,* Volume 9 Issue 8, November 1997, pp. 1735–1780.

[11] L. Almuqren and A. Cristea, "Framework for sentiment analysis of Arabic text", *Proceedings of the 27th ACM Conference on Hypertext and Social Media, Halifax*, Nova Scotia, Canada, July 10-13 2016, pp. 315-317.

[12] L. Al-Horaibi and M. Khan, "Sentiment Analysis of Arabic Tweets Using Semantic Resources", *International Journal of Computing and Information Sciences,* Volume 13 Issue 1, January 2017, pp. 9-18.

[13] M. AL-Smadi, O. Qawasmeh, M. Al-Ayyoub and Y. Jararweh, "Deep Recurrent Neural Network vs. Support Vector Machine for Aspect-Based Sentiment Analysis of Arabic Hotels' Reviews", *Journal of Computational Science,* doi: 10.1016/j.jocs.2017.11.006, Volume 27, July 2018, Pages 386-393.

[14] B. Wang and M. Liu, "Deep Learning for Aspect-Based Sentiment Analysis", *Stanford University report, [online]* Available at: https://cs224d.stanford.edu/reports/WangBo.pdf, 2015.

[15] F. Mahyoub, M. Siddiqui and M. Dahab, "Building an Arabic sentiment lexicon using semi-supervised learning", *Journal of King Saud University-Computer and Information Sciences,* doi: 10.1016/j.jksuci.2014.06.003, Volume 26 Issue 4, December 2014, pp. 417–424.

[16] Y. Regragui, L. Abouenour, F. Krieche and K. Bouzoubaa, "Arabic WordNet: New Content and New Applications", *Proceedings of the 8th Global Wordnet Conference (GWN 2016),* Bucharest, Romania, January 2016, pp. 330-338.

[17] M. Mike Frampton, "Mastering Apache Spark", *book. Packt Publishing,* September 2015, ISBN: 9781783987146

[18] R. Dua, N. Pentreath and M. Ghotra, "Machine Learning with Spark", *second edition, book.* April 2017, ISBN: 9781785889936

[19] R. Garreta, G. Moncecchi, T. Hauck and G. Hackeling, "*Scikit-learn: Machine Learning Simplified*", book. 2017, ISBN: 9781847197528

[20] M. Markatou, H. Tian, S. Biswas and G. Hripcsak, "Analysis of Variance of Cross-Validation Estimators of the Generalization Error", *Journal of Machine Learning Research*, July 2005, pp. 1127-1168.

[21] I.Hamed, M. Elmahdy and S. Abdennadher, "Building a First Language Model for Code-switch Arabic-English", *Procedia Computer Science Volume* 117, April 2017, pp. 208-216.

[22] T. Zerrouki and A. Balla, "Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems", *Data in Brief,* February 2017, pp. 147-151.

[23] "Full Emoji List, v12.0", https://unicode.org/emoji/charts/full-emoji-list.html, *Last accessed on* 15 July 2019.

[24] F. Al Omran and C. Treude, "Choosing an NLP Library for Analyzing Software Documentation: A Systematic Literature Review and a Series of Experiments", *IEEE/ACM 14th International Conference on Mining Software Repositories (MSR),* May 20-28 2017, pp. 187-197.

[25] M. Dahab, R. Al-Mutawa and A. Ibrahim, "A Comparative Study on Arabic Stemmers", *International Journal of Computer Applications,* Volume 125 Issue 8, September 2015, pp. 38-47.

[26] C. Freksen, L. Kamma and K. Larsen, "Fully understanding the hashing trick", *32nd International Conference on Neural Information Processing Systems, Montréal,* Canada, December 3-8 2018, pp. 5394-5404.

[27] T. White, "Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale", *fourth edition, book.* April 2015, ISBN: 9781491901632

[28] M. Nabil, M. Aly and A. Atiya, "ASTD: Arabic Sentiment Tweets Dataset", *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal,* September 17–21 2015, pp. 2515–2519.

Citation of this Article:

Mohamed A. Ahmed, “Arabic Sentiment Analysis using Apache Spark” Published in *International Research Journal of Innovations in Engineering and Technology* - *IRJIET*, Volume 4, Issue 2, pp 31-40, February 2020.

****