Sentiment Analysis of Moroccan Tweets using Naive Bayes Algorithm

Abdeljalil el Abdouli
Ecole Nationale Supérieure d'Electricité et de Mécanique de Casablanca
7 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Larbi Hassouni
University of Hassan II of Casablanca
16 PUBLICATIONS **13** CITATIONS
SEE PROFILE

Houda Anoun
Instituto Politécnico de Lisboa
22 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Sentiment Analysis of Moroccan Tweets using Naive Bayes Algorithm

Abdeljalil EL ABDOULI, Larbi HASSOUNI, Houda ANOUN
RITM Laboratory, CED Engineering Sciences
Ecole Superieure de Technologie
Hassan II University of Casablanca, Morocco
elabdouli.abdeljalil@gmail.com, lhassouni@hotmail.com, houda.anoun@gmail.com

Abstract—**Twitter is a web-based communication platform, which allows its subscribers to disseminate messages called "tweets" of up to 140 characters where they can share thoughts, post links or images. Therefore, Twitter is a rich source of data for opinion mining and sentiment analysis. The simplicity of use and the services offered by the Twitter platform allow it to be widely used in the Arab world and especially in Morocco, this popularity leads to an accumulation of a large amount of raw data that can contain a lot of valuable information. In this paper, we address the problem of sentiment analysis in Twitter platform. First, we try to classify the Moroccan users' tweets according to the sentiment expressed in them: positive or negative. Second, we discover the subjects related to each category to determine what they concern, and finally, we locate these "tweets" on Moroccan map according to their categories to know the areas where the tweets come from. To accomplish this, we adopt a new practical approach that applies sentiment analysis to Moroccan "tweets" using a combination of tools and methods which are: (1) Apache Hadoop framework (2) Natural Language Processing (NLP) techniques (3) Supervised Machine Learning algorithm "Naive Bayes" (4) Topic Modeling using LDA (5) Plotting tool for interactive maps called "Folium". The first task of our proposed approach is to automatically extract the tweets with emotion symbols (e.g., emoticons and emoji characters) because they directly express emotions regardless of used language, hence they have become a prevalent signal for sentiment analysis on multilingual tweets. Then, we store the extracted tweets according to their categories (positive or negative) in a distributed file system using HDFS (Hadoop Distributed File System) of Apache Hadoop framework. The second task is to preprocess these tweets and analyze them by using a distributed program written in Python language, using MapReduce of Hadoop framework, and Natural Language Processing (NLP) techniques. This preprocessing is fundamental to clean tweets from #hashtags, URLs, abbreviations, spelling mistakes, reduced syntactic structures, and many; it also allows us to deal with the diversity of Moroccan society, because users use a variety of languages and dialects, such as Standard Arabic, Moroccan Arabic called "Darija", Moroccan Amazigh dialect called "Tamazight", French, English and more. Afterward, we classify tweets obtained in the previous step using Naive Bayes algorithm into two categories (positive or negative), then we use the Topic Modeling algorithm LDA to discover general topics behind these classified tweets. Finally, we graphically plot classified tweets on our Moroccan map by using the coordinates extracted from them.**

Keywords: Apache Hadoop framework; HDFS; MapReduce; Python Language; Natural Language Processing; Supervised Machine Learning algorithm "Naive Bayes"; Topic Modeling algorithm LDA; Plotting tool for interactive maps.

I. Introduction

The emergence of Web 2.0 has led to an accumulation of valuable information and sentimental content in the Web; such content is often found in the comments of users of Social Network Platforms, in messages posted in discussion forums and product review sites, etc. The Twitter platform is very popular, and its users post a lot of comments to express their opinions, sentiments, and other information. This transforms twitter platform into a rich source of data for data mining and sentiment analysis. In this paper, we are interested in the sentiment analysis of the Moroccan users, we provide, below, some statistics on their activities. According to the Arab Social Media Report [1], which started in 2011 and aims to understand the impact of social media on societies, development, and governance in the Arab region, the monthly number of active users of the platform Twitter nearly doubled between 2014 and 2017. It went from 5.8 million to about 11.1 million. Regarding Morocco, the number of active users of the Twitter platform has grown of 146,300 users, in the last three years, to reach the number of 200 thousand users. Morocco thus ranks 9th among the Arab countries registering the highest number of users. These statistics prompted us to lead a study that aims to analyze the sentiments expressed in the tweets published by Moroccan users, despite the difficulties quoted before.

The primary aim of this research is to identify the sentiments contained in the tweets posted from the Moroccan region by proposing a new practical approach for analyzing the Moroccan user-generated data on Twitter. Our approach is based on a system, which automatically handles the streaming of the most recent tweets from Twitter platform using the open and accessible API of Twitter that returns well-structured tweets in JSON (JavaScript Object Notation) format. These tweets shape the training set, and are classified into two categories (Positive or Negative) according to the emotion symbols (e.g., emoticons and emoji characters) which exist in each tweet, then they are stored in our distributed system using HDFS [2]. These tweets are preprocessed by a distributed program using MapReduce [3], which is written in Python language using Natural Language Processing (NLP) techniques [4], and it's launched on MapReduce using the Pig UDF [5] (User Defined Functions). This preprocessing is fundamental to clean the tweets which are very noisy and contain all kind of spelling, grammatical errors and also to handle the linguistic diversity used by Moroccan users in the tweets. The result of the previous step is a clean and filtered corpus of tweets that is divided into "Positive" (text with happy emoticons), and Negative" (text with sad and angry emoticons) samples. This corpus is used to form the training set for the Naive Bayes algorithm to identify the sentiment within the new collected tweets, then we apply topic modeling using LDA to discover the hidden topics within these tweets. Finally, we graphically plot the classified tweets using a tool called "Folium" on our Moroccan map by using the coordinates extracted from them, to discover the relationship between the areas of classified tweets and determined topics.

The remainder of the paper is organized as follows; we present some related work in Section II. In Section III, we introduce the tools and methods used to realize our system. In Section IV, we describe our system. Finally, in Section V; we end with a conclusion and work in perspective.

II. Related Work

Sentiment Analysis is receiving an increasingly growing interest from many researchers, which have begun to search various ways of automatically collecting training data and perform a sentiment analysis. [12] have relied on emoticons for defining their training data. [13] have used #hashtags for creating training data and they limit their experiments to sentiment/non-sentiment classification, rather than (positive-negative) classification. [14] have used emoticons such as ":-)" and ":-(" to form a training set for the sentiment classification, the author collected texts containing emoticons from Usenet newsgroups, and the dataset was divided into "positive" and "negative" samples. [15] have covered techniques and approaches that promise to enable the opinion-oriented information retrieval directly. [16] have used Twitter to collect training data and then to perform a sentiment search, they construct a corpus by using emoticons to obtain "positive" and "negative" samples and then use various classifiers, the best result was obtained by using Naïve Bayes Classifier.

III. Tools And Methods

A. Apache Hadoop

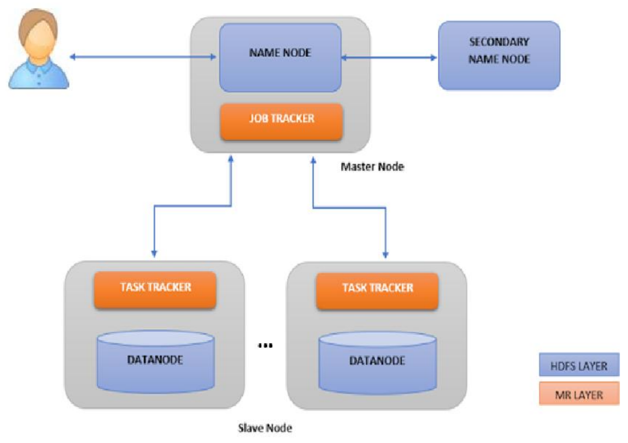


Figure 1. Apache Hadoop Architecture

Our approach is built using a specialized infrastructure, based on the Apache Hadoop Framework. The Apache Hadoop is an open-source software framework written in Java for processing, storing and analyzing large volumes of unstructured data on computer clusters built from commodity hardware.

The Hadoop Framework become a brand name, which contains two primary components. The first one is HDFS [5], which stands for Hadoop distributed file system; it is an open-source data storage, inspired by GFS (Google File System), it is a virtual file system that looks similar to any other file system, but the difference is that the file gets split into smaller files. The second one is MapReduce, which is an open-source programming model developed by Google Inc. Apache adopted the ideas of Google MapReduce and improved it. MapReduce provides a mechanism to break down every task into smaller tasks and the integration of results.

The HDFS (Hadoop Distributed File System) [2] system has many similarities with existing distributed file systems. However, the differences are significant, it is highly fault-tolerant and designed using low-cost hardware, also designed to be available and scalable. It provides high throughput access to stored data and can store massive files reaching the terabytes. By default, each stored file is divided into blocks of 64 MB, each block is replicated in three copies. The HDFS is based on Master and Slaves architecture in which the master is called the NameNode and slaves are called DataNodes, and it consists of:

a) Single NameNode: running as a daemon on the master node, it holds the metadata of HDFS by mapping data blocks to data nodes, and it is the responsible of managing the file system namespace operations.

b) Secondary NameNode: performs periodic checkpoints of the file system present in the NameNode and periodically joins the current NameNode image and the edits log files into a new image and uploads the new image back to the NameNode.

c) DataNodes: running as daemons on slave nodes, they manage the storing of blocks within the node (their default size is 128 MB). They perform all file system operations according to instructions received from the NameNode, and send a Heartbeat containing information about the total storage capacity of DataNode and Block report on every file and block they store to the NameNode.

The MapReduce [3] is the heart of Hadoop. It is a software framework that serves as the compute layer of Hadoop, it is modeled after Google's paper on MapReduce. It's characterized by fault tolerance, the simplicity of development, scalability, and automatic parallelization. It allows parallelizing the processing of massive stored data by decomposing the job submitted by the client into Map and Reduce tasks. The input of the Map task is a set of data as a key-value pair, and the output is another set of data as a key-value pair. The input of the reduce task is the output from a map task. Between the reduce input and the map output, MapReduce performs two essential operations, shuffle phase that covers the transformation of map outputs based on the output keys, and sort phase that includes the merge and sort of map outputs.

The MapReduce is also based on a master-slave architecture, and it consists of:

a) *JobTracker*: is running as a daemon on the master node, its primary role is accepting the job and assigning tasks to TaskTrackers running on slave nodes where the data is stored. If the TaskTracker fails to execute the task, the JobTracker assigns the task to another TaskTracker where the data are replicated.

b) *TaskTracker*: running as a daemon on slave nodes, it accepts tasks (Map, Reduce, and Shuffle) from JobTracker and executes program provided for processing. The TaskTrackers report the free slots within them to process data and also their status to the JobTracker by a heartbeat.

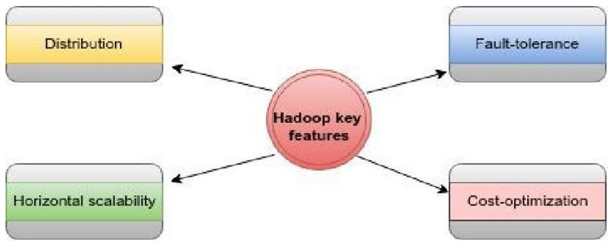


Figure 2. Key Features of Hadoop

The Key Features of Hadoop are:

Distribution: The storage and processing are spread across a cluster of smaller machines that work together.

Horizontal scalability: It is easy to extend a Hadoop cluster by adding new devices.

Fault-tolerance: Hadoop continues to operate even when a few hardware or software components fail to work correctly.

Cost-optimization: Hadoop runs on standard hardware; it does not require expensive servers.

Other Hadoop-related projects [7] at Apache that can be installed on top of or alongside Hadoop include:

- *Flume* [21]*:* is a framework for populating massive amounts of data into Hadoop.
- *Oozie* [22]*:* is a workflow processing system.
- *Mahout* [23]: Mahout is a data mining library.
- *Pig* [8]: a high-level data-flow language and execution framework for parallel computation.
- *Avro* [24]: a data serialization system.
- *HBase* [25]: a scalable and distributed database that supports structured data storage for large tables.
- *Hive* [26]: a data warehouse infrastructure that provides data summarization and ad hoc querying.
- *Spark* [27]*:* provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.
- And much more.

B. *Natural Language Processing (NLP)*

Natural Language Processing [4] is a part of computer science focused on developing systems that allow computers to recognize, understand, interpret and reproduce human language. NLP is considered as a subfield of artificial intelligence, and by using its algorithms, developers can perform tasks such as topic segmentation, translation, automatic summarization, named entity recognition, sentiment analysis, speech recognition, and much more.

There are two components of NLP. The first component is Natural Language Understanding (NLU) whose main function is to convert human language into representations that are easier for computer programs to manipulate. The other is Natural Language Generation (NLG) translate information from computer databases into readable human language. There are five steps in NLP:

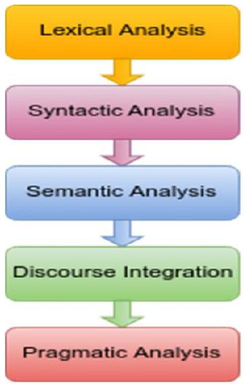


Figure 3. The steps of NLP

a) *Lexical Analysis:* identifying and analyzing the structure of words and dividing the whole text into paragraphs, sentences, and words.

b) *Syntactic Analysis:* analyzing and arranging words in a sentence in a structure that shows the relationship between them.

c) *Semantic Analysis:* extracting the exact meaning or the dictionary meaning of sentences from the text.

d) *Discourse Integration:* handles the meaning of current sentence depending on the sentence just before it.

e) *Pragmatic Analysis:* analyzing and extracting the meaning of the text in the context.

We use Natural Language Processing to perform tasks such as:

- Tokenization / segmentation
- Part of Speech (POS) Tagging: assign part-of-speech to each word.

- Parsing: create a syntactic tree for a given sentence.
- Named entity recognition: recognize places, people...
- Translation: Translate a sentence into another language.
- Sentiment analysis.
- Etc.

Using the NLP is necessary for our system because tweets are characterized by a noisy text containing many unwanted data; in addition, the language diversity used in Moroccan society adds many difficulties to the processing of tweets' content generated by Moroccan users.

C. *Scikit-learn and Naive Bayes algorithm*

Scikit-learn [18] is an open source library for machine learning that is simple and efficient for data mining and data analysis for the Python programming language. It is Built on *NumPy, SciPy,* and *Matplotlib* [10]; it includes many algorithms for classification, regression and clustering algorithm, and more. Because it is a robust library, we choose to Implement naive Bayes classifier in python with *scikit-learn*.

The Naive Bayes [19] is a supervised classification algorithm based on Bayes' Theorem with an assumption that the features of a class are unrelated, hence the word naive. The Naive Bayes classifier calculates the probabilities for every factor; then it selects the outcome with the highest probability.

Preprocessed tweets with NLP is given as input to train input set using Naïve Bayes classifier, then, trained model is applied to new collected tweets to generate either positive or negative sentiment.

The Bayes theorem is as follows:

$$P(H\,|E) = \frac{P(E\,|H) * P(H)}{P(E)}$$

Where:

- P(H): the probability of the hypothesis H being true. This is known as the prior probability.
- P(E): the probability of the evidence (regardless of the hypothesis).
- P(E|H) is the probability of the evidence given that hypothesis is true.
- P(H|E) is the probability of the hypothesis given that the evidence is there.

There are many applications of Naive Bayes Algorithms:

- Text classification/ Spam Filtering/ Sentiment Analysis
- Recommendation Systems.
- Real-time Prediction: Naive Bayes is a fast classifier, and it can be used for making predictions in real time
- Multi-class Prediction: more than two classes to be predicted.

D. *PIG UDF*

Apache Pig [8] is a popular data flow language; it is at the top of Hadoop and allows creating complex jobs to process large volumes of data quickly and efficiently. It will consume any data type: Structured, semi-structured or unstructured. Pig provides the standard data operations (filters, joins, ordering).

Pig provides a high-level language known as Pig Latin for programmers who are not so good at Java. It is a SQL-like language, which allows developers to perform MapReduce tasks efficiently and to develop their functions for processing data.

A Pig UDF [5] (User Defined Functions) is a function that is accessible to Pig but written in a language that is not PigLatin like Python, Jython or other programming languages; it is a function with a decorator that specifies the output schema.

We use Pig UDF to execute NLP program, written with Python language in a distributed manner using MapReduce. In consequence, the preprocessing became very fast and spread over the stored tweets.

E. *Topic Modeling Using Latent Dirichlet Allocation(LDA)*

Topic modeling allows us to organize, understand and summarize large collections of textual information. It helps to discover hidden topical patterns that are present in the collection; annotate documents according to these topics; and use these annotations to organize, search and summarize texts.

Topic models are unsupervised machine learning algorithms, which allow discovering hidden thematic structure in a collection of documents. These algorithms help us to develop new ways of text exploration. Many techniques are used to obtain topic models, but the most used one is Latent Dirichlet Allocation (LDA) [17].

LDA algorithm works as a statistical machine learning and text data mining; it allows discovering the different topics in a collection of documents. It consists of a Bayesian inference model that calculates the probability distribution over topics in each document, where each topic is characterized by a probabilistic distribution based on a set of words.

The LDA algorithm is used in our system, to discover the topics of classified tweets (positive and negative). For this reason, we implement a free python library for LDA called "Gensim" [20].

F. *Interactive maps using Folium*

Folium [11] is a powerful Python library that allows visualizing geospatial data onto interactive maps; it provides the facilities to transform coordinates to different map projections. The visualization happens "inline" or within the Python environment, using *IPython Notebook* and the results are interactive which makes this library very useful for dashboard building.

The Plotting of classified tweets in Moroccan map is necessary to discover the general mood in Moroccan regions as well as the dominant topics by using LDA.

IV. Architecture Of The System

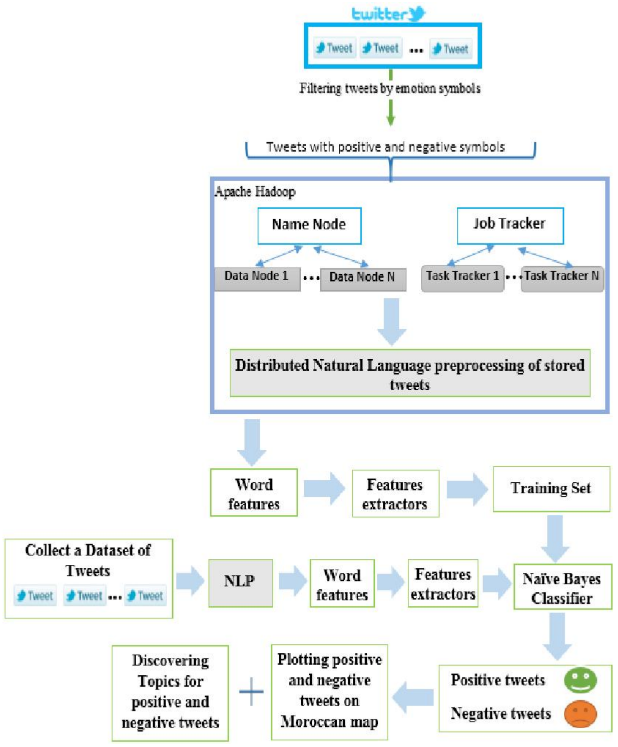

twitter
Tweet
Tweet
Tweet
Filtering tweets by emotion symbols
Tweets with positive and negative symbols
Apache Hadoop
Name Node
Job Tracker
Data Node 1
Data Node N
Task Tracker 1
Task Tracker N
Distributed Natural Language preprocessing of stored tweets
Word features
Features extractors
Training Set
Collect a Dataset of Tweets
NLP
Word features
Features extractors
Naïve Bayes Classifier
Discovering Topics for positive and negative tweets
Plotting positive and negative tweets on Moroccan map
Positive tweets
Negative tweets

Figure 4. Architecture of the System

The first part of our system involves the extraction of data from the social network Twitter, which is an essential task in the data analysis process. All these tweets contain coordinates from different locations in Morocco and stored in the HDFS.

A. Data Description, Collection, and Storage

1) Data Description:

Twitter has unique conventions that make it distinct from other social network platforms; indeed tweets have many unique attributes, which differentiates Twitter from other platforms in the field of research. The First convention is the maximum length of a tweet, which is 140 characters, an average length of 14 words per tweet. The second convention is the availability of data using Twitter API; it is much easier to collect a large volume of tweets for training.

In Twitter messages, we can find acronyms, emoticons and other characters that express special meanings. The Emoticons can be represented using punctuation and letters or pictures; they express the user's mood. Emoticons can be categorized as:

- Happy emoticons (positive) :-) :) :D, etc
- Sad emoticons (Negative) :-(:(:-c, etc.

To use the method based on the emotion symbols we need to make an assumption. This assumption is that the emotion symbols in the tweet represent the overall sentiment contained in that tweet. This assumption is reasonable because in the majority of cases the emotion symbols will correctly represent the overall sentiment of that tweet as the maximum length of a tweet is 140 characters.

2) Data Collection:

Twitter Platform allows developers to collect data via Twitter API, but first, they need to create an account on *https://apps.twitter.com*. For each created account, Twitter provides four secret information: consumer key, consumer secret key, access token and access secret token, then we are authorized to access the database and retrieve tweets using the streaming API.

To get tweets that contain emoticons which are generated in Moroccan region, we filter tweets by location coordinates using the Streaming API and by a list of positive and negative emotion symbols. We get the geographical coordinates (latitude and longitude) of Morocco that we utilized in this filter, by using the specialized website in geolocation *http://boundingbox.klokantech.com*.

To handle the streaming of data from Twitter, we used Python library *Tweepy* [9] as shown in the script below that allows accessing to Twitter API.

```
import tweepy
import json
import hadoopy

consumer_key= XXX
consumer_secret= XXX
access_token= XXX
access_token_secret= XXX

hdfs_path_Pos = 'hdfs://master:54310/tweets_Positive/'
hdfs_path_Neg = 'hdfs://master:54310/tweets_Negative/'

POSITIVE = ["*O", "*-*", "*O*", "*o*", "* *",":P", ":D", ":d",
":p", ";P", ";D", ";d", ";p",":-)", ";-)", ":=)", ";=)", ":<)", ":>)",
";>)", ";=)", "=}", ":)", "(:;)","(;", ":}", "{:", ";}", "{;:]","[;",
":')", ";')", ":-3","{;", ":]",";-3", ":-x", ";-x", ":-X",";-X", ":-}", ";-
=}", ":-]",";-]", ":-.)","^_^", "^-^","😀",, "😁","😃", "😄","😅","
😆","😇","😈","😉","😊","😋","😌","😍","😎","😏","😗","😘
","😙","😚","😛","😜","😝"]

NEGATIVE = ["":(", ";(", ":'(","=(", "={", "):", ");",")':", ")';", ")=",
"}=",";-{{", ";-{", ":-{{", ":-{",":-(", ";-(",":,)", ":'{","[:", ";]","😒
","😐","😑","😓","😔","😕","😖","😠","😡","😞","😟","😣","😢","
😦","😧","😨","😩","😪","😫","😭",,"😰","😱"]

class StdOutListener(tweepy.StreamListener):
   def on_data(self, data):

      decoded = json.loads(data)
      tweet_txt = decoded["text"]

     if decoded['place']['country_code'] == 'MA':
        ...
        for tok in tweet_txt.split(" ")
            for emoji in POSITIVE:
                if emoji.decode('utf-8') in tok:
                hadoopy.put(localeFile ,hdfs_path_Pos)
                ...
stream.filter (locations=[-
17.2122302,21.3365321,0.9984289,36.0027875],async='true',enc
oding='utf8')
```

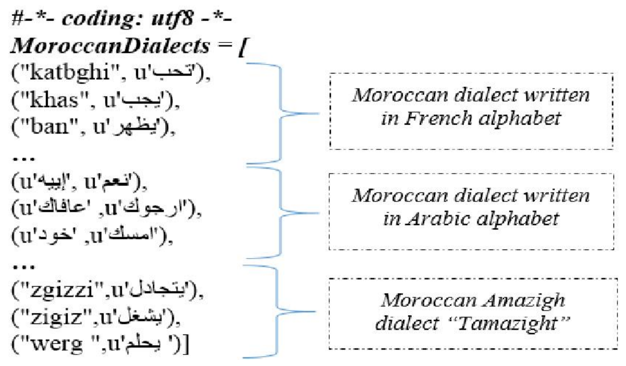
3) Data storage using HDFS

The storage of filtered tweets gathered from the Twitter API in HDFS is handled by using a Python wrapper for Hadoop called *Hadoopy* [6], which allows performing operations like reading and writing data from and to HDFS. We create Two folders in HDFS, one for the positive tweets (*'hdfs://master:54310/tweets_Positive/)'* and the other for the negative tweets (*'hdfs://master:54310/tweets_Negative/'*) as shown in the previous script.

B. Processing filtred tweets with NLP

A major issue which faces us when we are dealing with Twitter data is the informal style of the posts. Most tweets are written informally, contain many errors and abbreviations, and do not follow any grammatical rule. To minimize the effect of this informality on our classification, we will pre-process tweets, in order to clean them, before using them. We might find words misspelled, and therefore must be detected and corrected to evaluate sentiment more accurately.

Also, the linguistic diversity that characterizes the communication of Moroccan users on social network Twitter complicate the task of classification. To deal with this issue, we create a python file that contains a dictionary of words that we gathered manually, to transform words written in Moroccan dialect, or in a dialect of Berber Tamazight into Standard Arabic. These words could be written using the Arabic or French alphabet then we store it in each slave node of our cluster and imported inside the NLP script executed in these nodes. Below, a part of this file



```
#-*- coding: utf8 -*-
MoroccanDialects = [
("katbghi", u'تحب'),
("khas", u'يجب'),
("ban", u'يظهر'),
...
(u'إيييه', u'نعم'),
(u'عافاك' ,u'ارجوك'),
(u'خود' ,u'امسك'),
...
("zgizzi",u'يتجادل'),
("zigiz",u'يشغل'),
("werg ",u'يحلم ')]
```

Moroccan dialect written in French alphabet
Moroccan dialect written in Arabic alphabet
Moroccan Amazigh dialect "Tamazight"

The NLP step contains all the programs needed to preprocess the stored data, starting with parsing the tweets and extracting relevant information for our analysis, which are:

- *Text*: text of the tweet.
- *Lang*: language used by the user to write the tweet.
- *Coordinates:* location coordinates of a tweet.

The library used to preprocess tweets with NLP is the Natural language processing Toolkit (NLTK) [9], which is a set of open-source Python modules, allowing programs to work with the human language data. It involves capabilities for tokenizing, parsing, and identifying named entities as well as many more features; it also provides over 50 corpora and lexical resources such as WordNet and a set of text processing libraries.

We use the following steps for preprocessing the filtered tweets:

a) Delete unnecessary data: usernames, emails, hyperlinks, retweets, punctuation, possessives from a noun, duplicate characters, and special characters like smileys.

b) Shorten any elongated words (تكبير → تكبيــــــــــــــــــــــر)

c) Normalize whitespace (convert multiple consecutive whitespace characters into one whitespace character).

d) Convert hashtags into separate words, for example; the hashtag #SentimentAnalysis is converted into two words Sentiment and analysis.

e) Create a function to detect the language used to write the text of tweet (Standard Arab, French or English).

f) Create a function for automatic correction of spelling mistakes.

g) Create a list of contractions to normalize and expand words like (*What's=>What is)*

h) Delete the suffix of a word until we find the root. For example (*Stemming => stem)*

i) Remove tokens of part of speech that are not important to our analysis by using the Part-Of-Speech software of Stanford University. This software reads the text and assigns parts of speech (noun, verb, adjective) to each word.

j) Remove stopwords of standard Arabic *(أن, إن, بعد...)*, French (*alors, à, ainsi, ...*), and English *(about, above, almost, ...*).

These steps are assembled in a python file called *NLTK_Tweet.py*. This file is executed in a distributed manner by an Apache Pig file called *Pig_Tweet.pig*. The file *NLTK_Tweet.py* needs to be registered in the script of the Pig file using *Streaming_python* as follows:

```
REGISTER 'hdfs://master:54310/apps/NLTK_Tweet.py' USING
streaming_python AS nltk_udfs;
```

The launch of our file *NLTK_tweet.py* is defined as follows:

```
data = LOAD '/tweets_Positive /*  using TextLoader() AS
(line:chararray);

Result =  FOREACH data GENERATE
nltk_udfs.NLTK_Function(line));
```

C. Naïve Bayes Classifier

1) Data

Using Twitter API, we were able to collect experimentally a sample of 700 tweets (divided into positive and negative tweets) based on the emotion symbols and location filter, and 230 tweets as test set for accuracy evaluation of our classifier. All collected tweets are stored in a distributed manner using HDFS. The purpose of this paper, among others, is to be able to automatically classify a tweet as a positive or negative tweet. The classifier needs to be trained, that is why we use the stored tweets as training set after preprocessing step with NLP.

2) Implementation

For example, a fragment of the list of positive tweets looks like:

```
pos_tweets = [('I love this song, 'positive'),
           ('This picture is wonderful, 'positive'),
           ('I feel great this evening, 'positive'),
           ('This is my favorite food', 'positive')]
```

A fragment of the list of negative tweets looks like:

```
neg_tweets = [('I do not like this song, 'negative'),
           ('This picture is horrible', 'negative'),
           ('I feel sad this evening, 'negative'),
           (I hate this food, 'negative')]
```

We take these two lists and create a single list of tuples each containing two elements. The first element is an array containing the words and the second element is the type of sentiment. We ignore the words smaller than two characters, and we use lowercase for everything. The code is as follows:

```
tweets = []
for (words, sentiment) in pos_tweets + neg_tweets:
    words_filtered = [e.lower() for e in words.split() if len(e) >= 3]
    tweets.append((words_filtered, sentiment))
```

The tweets list now looks like this:

```
tweets = [
    (['love', 'this', 'song'], 'positive'),
    (['this', 'picture', 'wonderful'], 'positive'),
    (['feel', 'great', 'this', evening], 'positive'),
    (['this', 'favorite', 'food'], 'positive'),
    (['not', 'like', 'this', 'song'], 'negative'),
    (['this', picture, 'horrible'], 'negative'),
    (['feel', sad, 'this', evening], 'negative'),
   ([hate, 'this', 'food'], 'negative')])
]
```

3) Classifier

The list of word features needs to be extracted from the tweets. It is a list of every distinct word ordered by the frequency of occurrences. We use the following function and the two helper functions to get the list.

```
word_features = get_word_features(get_words_in_tweets(tweets))

def get_words_in_tweets(tweets):
    all_words = []
    for (words, sentiment) in tweets:
      all_words.extend(words)
    return all_words

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    word_features = wordlist.keys()
    return word_features
```

If we take a pick inside the function *get_word_features*, the variable 'wordlist' contains:

```
<FreqDist:
'this': 7,
'song': 2,
'feel': 2,
'evening': 2,
'picture': 2,
'wonderful': 1,
'favorite': 1,
'food':1
...
>
```

The list of word features is as follows:

```
word_features = [
'this',
'song',
'feel',
'evening',,
'picture',
'wonderful',
'favorite',
'food':1
   ...
]
```

The results show that 'this' is the most used word in our tweets, followed by 'song, then 'fell and so on …

We need to choose what features are pertinent to create our classifier. First, we need a feature extractor that returns a dictionary of words that are contained in the input passed. In our case, the input is the tweet. We use the word features list defined above along with the input to create the dictionary.

```
def extract_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains(%s)' % word] = (word in
document_words)
    return features
```

For example, let's call the feature extractor with the first positive tweet ['love', 'this', 'song']. We obtain the following dictionary which indicates that the document contains the words: 'love', 'this' and 'song'.

```
{'contains(love)': True,
 'contains(evening)': False,
 'contains(this)': True,
 'contains(picture)': False,
 'contains(wonderful)': False,
 'contains(song)': True,
 'contains(favorite)': False,
 'contains(food)': False,
 'contains(horrible)': True,
 'contains(hate)': False,
 'contains(sad)': False,}
```

We use the method *apply_features* to apply the features to our classifier, and we pass the list of tweets along with the feature extractor defined above.

```
training_set = nltk.classify.apply_features(extract_features,
tweets)
```

The variable called 'training_set' contains the labeled feature sets, it is a list of tuples, where each tuple containing the feature dictionary and the sentiment category for each tweet.

```
[({'contains(love)': True,
  ...
  'contains(this)': True,
  ...
  'contains(song)': True,
  ...
  'contains(hate)': False,
  'contains(sad)': False},
 'positive'),
 ({'contains(love)': False,
  'contains(picture)': True,
  ...
  'contains(this)': True,
  ...
  'contains(wonderful)': True,
  ...
  'contains(hate)': False,
  'contains(sad)': False},
 'positive'),
 ...]
```

Now we can train our classifier using the training set.

```
classifier = nltk.NaiveBayesClassifier.train(training_set)
```

4) Testing the Classifier

To check the quality of our classifier by using the test set, we use an accuracy method in nltk that computes the accuracy rate of our model. Our approach reaches an accuracy of 69% which is considerate as a good value in our case.The simplest way to improving the accuracy of our classifier would be to increase the size of the training set.

```
import nltk.classify.util
print 'accuracy:', nltk.classify.util.accuracy(classifier, testTweets)
```

5) Classification of new collected tweets

Now that we have our classifier initialized and ready, we can try to classify collected and preprocessed tweets using NLTK and see what is the sentiment category output (positive or negative). Our classifier can detect that tweets have positive or negative sentiments. We evaluate our approach by streaming new collected tweets from Twitter API estimated at 300 tweets. A sample of collected tweets is as follows:

الوداد نسي ماتش الأهلي و بدأ يفكر في ماتش باتشوكا المكسيكي في كأس العالم و المصريين لازالو يحللون الهدف هل شرعي أو... https://t.co/PibiSBFoms from @monsef_filali at 11/07/2017 14:33

@lescitoyensorg Et ça continue ... from @cramounim at 11/07/2017 19:59

@YouTube أداء مؤثر جدا و في قمة الخشوع لشيخي ماهر رعاه الله و حفظه مليكة from @khadimarrahmane at 11/07/2017 15:39

Watching winner slowly realise that they're being kidnapped is the funniest thing ever #WinnerOverFlowers from @winneroediya at 11/07/2017 15:29

...

The below code is used to classify these new collected tweets using the classifier.

```
import nltk
from nltk.probability import FreqDist, ELEProbDist
from nltk.classify.util import apply_features,accuracy
...

print classifier.classify(extract_features(tweet.split()))
```

The output of the classification is the sentiment category of each tweet which is positive or negative. Our approach show good result despite the difficulties of multilingual tweets. some tweets are misclassified but we can override this issue by increasing the number of tweets in training set.

D. Topic Modeling with LDA

LDA is a probabilistic model used to determine the covered topics using the word frequency in the text. We use LDA in our approach for the classified tweets for each category(positive and negative). The LDA step will explain the reasons for the Moroccan user's mood. To generate the LDA model, we need to construct a document-term matrix with a package called "Gensim", which allows us to determine the number of occurrences of each word in each sentiment category. The LDA program used to discover topics is as follows:

```
from gensim import corpora, models
import hadoopy

fname_in = '/home /corpusTweetsSeniment.csv'
documentsPos = ""
documentsNeg = ""

with open(fname_in, 'rb') as fin:
  reader = csv.reader(fin)
  for row in reader:
     if row[3] == "positive":
        documentsPos = documentsPos + row[2] + ","
    elif row[3] == "negative":
        documentsNeg = documentsNeg + row[2] + ","

documentsPos =  documentsPos[:-1]
documentsNeg =  documentsNeg[:-1]

print ( ---- Topics  for positive tweets -----)
texts = [word.split() for word in documentsPos.split(",")]
dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
lda = models.ldamodel.LdaModel(corpus, id2word=dictionary,
num_topics=2 , passes=10)
lda.show_topics()

print ( ---- Topics  for negative tweets -----)
texts = [word.split() for word in documentsNeg.split(",")]
dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
lda = models.ldamodel.LdaModel(corpus, id2word=dictionary,
num_topics=2 , passes=10)
lda.show_topics())
```

For instance, the topics detected by our LDA model are:

```
---- Topics  for positive tweets ------
Topic #1: Maroc, football, equipe, russie, qualification
Topic #2: كأس، افريقيا، الوداد، جماهير

---- Topics  for negative tweets -----
Topic #1:تلوث، تهديد، سكان، بيئة، إشكالية
Topic #2: accident, circulation, mort, blesses, route
```

E. Plotting the classified tweets on map using Folium

During the streaming of filtered tweets from the Twitter API, we extract the coordinates (longitude and latitude) of each tweet. We then use these coordinates in Folium to show locations of tweets on our Moroccan map. The tweets that belong to positive mood are in green color and the negative mood are in red color. The developed program is as follows:

```
import folium
import csv

filename = '/home /corpusTweetsCoordinatesSeniment.csv'

map = folium.Map(location=[36.0027875,-17.2122302],
zoom_start=6)

  with open(filename) as f:
      reader = csv.reader(f)
      for row in reader:
        if row[3] == "positive":
          folium.Marker(location=[row[1],row[0]],popup=row[
           2],icon = folium.Icon(color ='green')).add_to(map)

        elif row[3] == "negative":
          folium.Marker(location=[row[1],row[0]],popup=row[
           2],icon = folium.Icon(color ='red')).add_to(map)

map.save("/home/abdeljalil/map_tweets.html")
map
```

The Figure 4 below shows the result of plotting classified tweets on the Moroccan map:

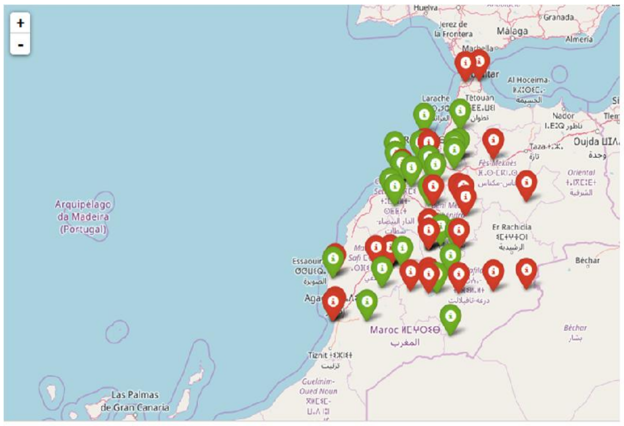


Figure 5. Locations of classified tweets on Moroccan map

This representation gives an idea about the locations of the Moroccan positive and negative tweets. This map and the topics generated by LDA are a good and perfect combination to study the mood of the Moroccan users, and more specifically to answer the two questions : Why this mood (LDA) and Where (Map).

V. Conclusion and Future Work

Twitter nowadays became one of the major tools and new types of the communication. People directly share their opinions through Twitter to the public. One of the very common analyses, which can be performed on a large number of tweets, is sentiment analysis. In the proposed work, we have presented a method for an automatic collection of a corpus that can be used to train a multilingual sentiment classifier so that it will be able to classify tweets into positive and negative. This classification is based on Naive Bayes classifier. Then we use methods to get insight from the classified tweets as the hidden topics and the locations of positive and negative tweets, which can conduct to better understanding of the Moroccan mood about different subjet and events. As future work, we plan to increase the accuracy of our classifier by increasing the number of filtered tweets, and by improving the preprocessing with NLP.

VI. References

[1] arabsocialmediareport, "Twitter in Arab Region". [Online]. Available: http://www.arabsocialmediareport.com/Twitter/LineChart.aspx. [Accessed: 01- Sep- 2017].

[2] Mrudula Varade and Vimla Jethani, "Distributed Metadata Management Scheme in HDFS", International Journal of Scientific and Research Publications,Volume 3, Issue 5, May 2013.

[3] M. Ghazi and D. Gangodkar, "Hadoop, MapReduce and HDFS: A Developers Perspective", Procedia Computer Science, vol. 48, 2015.

[4] M. Nagao, "Natural Language Processing and Knowledge", 2005 International Conference on Natural Language Processing and Knowledge Engineering.

[5] Pig.apache.org, "User Defined Functions". [Online]. Available: https://pig.apache.org/docs/r0.9.1/udf.html. [Accessed: 06- Sep -2017].

[6] "hadoopy", hadoopy.readthedocs.org. [Online]. Available: https://hadoopy.readthedocs.org/en/latest/. [Accessed: 01- Sep- 2017].

[7] "Apache Hadoop", [Online]. Available: https://en.wikipedia.org/wiki/Apache_Hadoop.html. [Accessed: 01- Mar- 2017]

[8] pig.apache.org, "Welcome To Apache Pig". [Online]. Available: https://pig.apache.org/. [Accessed: 01- Sep- 2017].

[9] nltk.org/, "nltk". [Online]. Available: http://www.nltk.org/. [Accessed: 01- Sep- 2017].

[10] matplotlib.org/basemap/, " Welcome to the Matplotlib Basemap Toolkit documentation". [Online]. Available: http://matplotlib.org/basemap/. [Accessed: 01- Sep- 2017].

[11] "Folium", [Online]. Available: https://github.com/python-visualization/folium. [Accessed: 01- Mar- 2017].

[12] A. Pak, and P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining", Proc. of LREC, 2010.

[13] D. Davidov, O. Tsur and A. Rappoport, "Enhanced sentiment learning using twitter hashtags and smileys", Proceedings of Coling, 2010

[14] Jonathon Read, "Using emoticons to reduce dependency in machine learning techniques for sentiment classification", In ACL. The Association for Computer Linguistics, 2005.

[15] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis", Foundations and Trends® in Information Retrieval, 2008.

[16] Alec Go, Lei Huang, and Richa Bhayani, "Twitter sentiment analysis", Final Projects from CS224N for Spring 2008/2009 at The Stanford Natural Language Processing Group, 2009

[17] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation", Journal of Machine Learning Research, 2003.

[18] scikit-learn.org/stable/, "scikit-learn". [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 01- Sep- 2017].

[19] Z. Zhang, "Naïve Bayes classification in R", Annals of Translational Medicine, vol. 4, no. 12, pp. 241-241, 2016.

[20] radimrehurek.com/gensim/, "Gensim topic modeling for humans". [Online]. Available: https://radimrehurek.com/gensim/. [Accessed: 01- Sep- 2017].

[21] flume.apache.org/, "Apache Flume". [Online]. Available: https://flume.apache.org/. [Accessed: 01- Sep- 2017].

[22] oozie.apache.org/, " Apache Oozie Workflow Scheduler for Hadoop". [Online]. Available: http://oozie.apache.org/. [Accessed: 01- Sep- 2017].

[23] mahout.apache.org/, "Apache Mahout: Scalable machine learning and data mining". [Online]. Available: http://mahout.apache.org/. [Accessed: 01- Sep- 2017].

[24] avro.apache.org/, "Welcome to Apache Avro!". [Online]. Available: https://avro.apache.org/. [Accessed: 01- Sep- 2017].

[25] hbase.apache.org/, "Apache HBase – Apache HBase™ Home". [Online]. Available: https://hbase.apache.org/. [Accessed: 01- Sep- 2017].

[26] hive.apache.org/, "APACHE HIVE TM". [Online]. Available: https://hive.apache.org/. [Accessed: 01- Sep- 2017].

[27] spark.apache.org/, " Apache Spark™ - Lightning-Fast Cluster Computing". [Online]. Available: https://spark.apache.org/. [Accessed: 01- Sep- 2017].