EMPIRIA. Revista de Metodología de las Ciencias Sociales
ISSN: 1139-5737
ISSN: 2174-0682
empiria@poli.uned.es
Universidad Nacional de Educación a Distancia
España

Arcila Calderón, Carlos; Ortega Mohedano, Félix; Álvarez, Mateo; Vicente Mariño, Miguel
Distributed Supervised Sentiment Analysis of Tweets: Integrating Machine Learning and Streaming Analytics for Big Data Challenges in Communication and Audience Research
EMPIRIA. Revista de Metodología de las Ciencias Sociales, núm. 42, 2019, -, pp. 113-136
Universidad Nacional de Educación a Distancia
España

DOI: https://doi.org/10.5944/empiria.42.2019.23254

Disponible en: https://www.redalyc.org/articulo.oa?id=297165961006

Distributed Supervised Sentiment Analysis of Tweets: Integrating Machine Learning and Streaming Analytics for Big Data Challenges in Communication and Audience Research

Análisis distribuido y supervisado de sentimientos en Twitter: Integrando aprendizaje automático y analítica en tiempo real para retos de dimensión big data en investigación de comunicación y audiencias

Carlos Arcila Calderón

University of Salamanca
carcila@usal.es (ESPAÑA)

Félix Ortega Mohedano

University of Salamanca
fortega@usal.es (ESPAÑA)

Mateo Álvarez

University Rey Juan Carlos
mateyoalvarez@gmail.com (ESPAÑA)

Miguel Vicente Mariño

University of Valladolid
miguelvm@soc.uva.es (ESPAÑA)

Recibido: 25.01 2018
Aceptado: 20.12.2018

RESUMEN

El análisis a gran escala de tweets en tiempo real utilizando el análisis de sentimiento supervisado representa una oportunidad única para la investigación de comunicación y audiencias. El poner juntos los enfoques de aprendizaje automático y de analítica en tiempo real en un entorno distribuido puede ayudar a los investigadores a obtener datos valiosos de Twitter con el fin de clasificar de forma inmediata mensajes en función de su contexto, sin restricciones de tiempo o almacenamiento, mejorando los diseños transversales, longitudinales y experimentales con nuevas fuentes de datos. A pesar de que los investigadores de comunicación y audiencias ya han comenzado a utilizar los métodos computacionales en sus rutinas, la mayoría desconocen el uso de las tecnologías de computo distribuido para afrontar retos de dimensión big data. Este artículo describe la implementación de métodos de aprendizaje automático paralelizados en Apache Spark para predecir sentimientos de tweets en tiempo real y explica cómo este proceso puede ser escalado usando computación distribuida tanto comercial como académica, cuando los ordenadores personales son insuficientes para almacenar y analizar los datos. Se discuten las limitaciones de estos métodos y sus implicaciones en los estudios de medios, comunicación y audiencias.

PALABRAS CLAVE

Análisis de sentimiento, Twitter, Big Data, Analítica en tiempo real, Investigación de comunicación y audiencias, Apache Spark.

ABSTRACT

The large-scale analysis of tweets in real-time using supervised sentiment analysis depicts a unique opportunity for communication and audience research. Bringing together machine learning and streaming analytics approaches in a distributed environment might help scholars to obtain valuable data from Twitter in order to immediately classify messages depending on the context with no restrictions of time or storage, empowering cross-sectional, longitudinal and experimental designs with new inputs. Even when communication and audience researchers begin to use computational methods, most of them remain unfamiliar with distributed technologies to face big data challenges. This paper describes the implementation of parallelized machine learning methods in Apache Spark to predict sentiments in real-time tweets and explains how this process can be scaled up using academic or commercial distributed computing when personal computers do not support computations and storage. We discuss the limitation of these methods and their implications in communication, audience and media studies.

KEY WORDS

Sentiment Analysis, Twitter, Big Data, Streaming, Machine Learning, Communication and Audience Research, Apache Spark.

1. INTRODUCCIÓN

There is a growing interest in surveying sentiments and opinions using large-scale data produced by social media (Cobb, 2015; O'Connor, 2010; Bollen, Mao & Pepe, 2011). However, most of these studies are based either on manual classification or automated content analysis using dictionaries that score words among others (e.g. giving an a priori negative or positive value to each word) (Leetaru, 2012; Feldman, 2013) and other approaches such as *supervised machine learning* (Vinodhini & Chandrasekaran, 2012) are scarce in communication and audience research (van Zoonen & Toni 2016) even when scholars in this field are aware of the advantages of computational methods to deal with large-scale textual analysis (Shahin, 2016). Moreover, the efforts to bring together automated sentiment analysis based on machine learning (*supervised sentiment analyses*) and streaming technologies that produce important amount of data in real time, are relatively new developments that private companies are adopting for several purposes but most social science scholars remain still unfamiliar. This paper intends to fulfill this methodological gap by describing and assessing the creation of machine learning models to predict sentiments in real-time tweets and explaining how this process can be scaled in communication and audience research using academic or commercial distributed computing when personal computers do not support computations and storage.

We explain what is *supervised sentiment analysis* and how useful are streaming analytics or real-time research in communication and audience studies. To joint these two approaches, this paper describes how communication scholars and social scientists can analyze the tone of big amount of tweets in real time using freely available resources such as Apache Spark, Python and the Application Program Interface (API) of Twitter. In a personal computer and based on NLTK and Scikit-Learn libraries, there is previous literature in communication and information sciences that show how to train a supervised machine learning model with different algorithms for classification and predict the sentiment of tweets in real time (Arcila et al., 2017). However, we state that local computing has serious limitations if communication scholars plan to scale this analysis with significantly higher amount of data, which requires scalable storage and distributed computing like the approach for Twitter detailed in Nodarakis et al. (2016). In fact, running streaming data analysis in distributed platforms has been a challenge in the complex and changing big data landscape (Turck & Hao, 2016). The incorporation of tools such as Apache Kafka has allowed the current most extended open software for distributed computing Apache Spark to achieve

this goal with Spark Streaming (Spark Kafka Integration, 2016), which can read code in Scala or also in Python (with the module PySpark).

In this paper, we show how communication scholars and social scientists can extend supervised sentiment analysis for big data and streaming challenges. This method is scalable using academic networks or commercial tools, such as the most popular Infrastructure as a Service (IaaS) Amazon Web Services (AWS), that offers Amazon S3 for massive storage and Amazon Elastic Computing Cloud (EC2) to create a flexible set of connected instances in the cloud in order to compute the analysis (reading and writing data directly from/to S3). The computational methods for big data problems in communication and audience research explained in this paper might help scholars to study large amounts of tweets in any language running sentiment analysis in real-time with few limitations. All these methods require programming skills, but exiting models allows short-time efforts and easy adaptations. They also require some discussion about an open-minded approach regarding the transition from conventional to computational research methods.

We provide scholars with all the commented code for Spark (written in Python and using PySpark) in an iNoteBook (ipynb)[1]. No mathematical background is needed to run the machine learning models, but a theoretical understanding of the algorithms will increase the quality of the research. Working with interdisciplinary teams (computer scientists, statisticians, computational linguistics, etc.) can also improve the results and save resources. The described procedure to monitor tweets in streaming might help testing traditional and emerging theoretical approaches in communication and audience research that require longitudinal data and might also contribute to experimental studies, which need real-time inputs to create or adapt stimuli. Additionally, this method brings social sciences closer to artificial intelligence, which is a field that is transforming many disciplines given its enormous potential to use mathematics and computers to model complexity and identify patterns and predict behaviors.

Predicting real time sentiment in Twitter during a long period also allows longitudinal analysis to detect changes over the time and compare these changes with day-to-day events, which in turn allow better interventions based on communication and audience research. In any case, we are also aware of the weaknesses of Twitter when it comes to reach depth of argumentation and the flaws that sometimes arise from this public opinion building-up processes.

2. SUPERVISED SENTIMENT ANALYSIS (SSA)

Sentiment analysis (hereinafter, also referred to as SA) is one of the main techniques used to study textual data at a small and large scale in social sciences and communication research. The main purpose of this methodology

[1] Scripts and documentation can be downloaded from: https://github.com/carlosarcila/autocop_en_distributed

is to recognize and evaluate the underlying emotions provided in the textual sources analyzed through their syntactic structure. These texts are subsequently classified, for example, into *positive*, *negative* or *neutral*. In the last years, this methodology has been mainly applied to the interpretation and analysis of social media texts, such as a given textual corpus from *Twitter*. These researches are usually achieved by examining the vocabulary of a text with a computer using a *lexicon*-based "dictionary", which processes, recognizes and evaluates the emotional tone behind the message. Sentiment analysis is often confused with *opinion mining*. Indeed, the latter is applied to detect polarity, and the identification of emotions is often used for the same purpose (Cambria, Schuller, Liu, Wang & Havasi, 2013); both techniques are different but complementary.

Identifying the prevailing sentiment in a written text is a complex task, even for a well-trained human brain. Therefore, automated sentiment analysis requires a constant and sharpened development that has been tackled from two main standpoints: semantic approaches (Turney, 2002), and machine learning techniques (Pang, Lee & Vaithyanathan, 2002). Semantic approaches are characterized by the use of sentiment dictionaries (*lexicons*) oriented towards polarity or opinion. These systems pre-process the text and divide it into words, subsequently verifying the presence of *lexicon* terms, in order to assign the text's polarity by adding the weighted polarity values of the terms ("sad" = -3; "happy" = +3). Additionally, these systems also include a fairly advanced treatment of modifiers (such as *very*, *little* or *extremely*), which increase or reduce polarity of the terms they are related to, and they include inverters or negators (such as *not* or *neither*), which invert the polarity of the word they are associated with. This was the method used by Turney (2002), who pioneered the application of automated sentiment analysis, which in this case focused on analyzing reviews on services and products.

As opposed to manual SA using human codifiers, or to computer-assisted automated analysis using dictionaries, *Supervised Sentiment Analysis (SSA)* applies *supervised machine learning* processes to generate models based on pre-annotated data. The aim is to predict, with a significant degree of reliability, the underlying sentiment in messages. This procedure uses classification algorithms, and their implementation in communicational processes allows for a fast analysis of communicational texts, thus avoiding any bias stemming from codifiers or lexicons with categories, which on an *a priori* basis are incapable of detecting subjects or topics, the context, or irony-sarcasm for example. SSA allows for transforming the classification model throughout the predictive process if such model is fed with additional and enriching annotated texts enhancing the adjustment process. One of the first approaches to this methodology was shown in the work of Pang et al. (2002), where supervised machine learning was applied to sentiment analysis of polarized reviews of movies, rating such reviews as positive or negative. Bermingham and Smeaton (2010) carried out a study on how short-length documents suggest that the sentiment they contain is more compact and explicit. Other researchers, such as Bakliwal, Arora, Madhappan, Kapre, Singh and Varma (2012) have focused on developing a function that allows for

tweet sentiment identification and classification, using a corpus of pre-annotated *tweets*.

SSA involves training a model with texts examples in order to summarize their syntactic structures using their words or sentences as features. Scholars that are familiar with Python can easily train a Naïve Bayes *classifier* for positive/negative previous labeled messages (training set) taking only adjectives as features for the models in few lines adapted from Kinsley (2017):

```
>>> import nltk
>>> import random
>>> from nltk.tokenize import word_tokenize
>>> from nltk.classify.scikitlearn import SklearnClassifier
>>> from sklearn.Naïve_bayes import MultinomialNB, BernoulliNB
>>> positive_messages = open("positive.txt","r").read()
>>> negative_messages = open("negative.txt","r").read()
>>> all_words = []
>>> documents = []
>>> allowed_word_types = ["J"]
>>> for p in positive_messages.split('\n'):
        documents.append( (p, "pos") )
        words = word_tokenize(p)
        pos = nltk.pos_tag(words)
        for w in pos:
        if w[1][0] in allowed_word_types:
        all_words.append(w[0].lower())
>>> for p in negative_messages.split('\n'):
        documents.append( (p, "neg") )
        words = word_tokenize(p)
        pos = nltk.pos_tag(words)
        for w in pos:
        if w[1][0] in allowed_word_types:
        all_words.append(w[0].lower())
>>> all_words = nltk.FreqDist(all_words)
>>> word_features = list(all_words.keys())[:5000]
>>> def find_features(document):
        words = word_tokenize(document)
        features = {}
        for w in word_features:
        features[w] = (w in words)
        return features
>>> featuresets = [(find_features(rev), category) for (rev, category) in
documents]
>>> random.shuffle(featuresets)
>>> testing_set = featuresets[10000:]
```

```
>>> training_set = featuresets[:10000]
>>> classifier = nltk.NaïveBayesClassifier.train(training_set)
```

In any case, even when SSA has not been completely accurate, nor has provided relevant or influential findings in *Twitter* analysis (Madlberger & Almansour, 2014), this technique has been recently applied to predict events in various domains (Kranjc et al, 2015; Preethi & Uma, 2015) such as finance (Smailović et al., 2013), social networks (Sluban et al., 2015) and election results (Smailović et al., 2015), with a more than promising prediction accuracy. In fact, in the last decade there has been an ever-increasing growth of SSA, particularly in the social and political science domain. Increasingly more experiments are based on this methodology-technique due to its comparative robustness. It is relevant to note that within communication and audience studies, there are early applications related to communicational analysis in social media. One of the few proven models is that of Bakliwal, Foster, Van der Puil, O'Brien, Tounsi and Hughes (2013), who put forward a sentiment analysis model with a political orientation based on supervised machine learning. The transition from and the dialogue with conventional Media Studies and Communication Research are still pending, as progress in these research approaches is not being leaded by Communication scholars yet. Our scientific discipline would benefit from this dialogue with other scientific and technical fields, as a way to increase our impact and efficacy.

A very good example of this dialogue is the work by Smailović et al. (2015) who monitored the sentiments in Twitter during the Bulgarian parliamentary elections in May 2013. The authors initially collected examples of general Bulgarian tweets (N=29,433) and political Bulgarian tweets (N=10,300), and manually classified the sentiment of these messages to build the training corpus and generate the machine learning models. Using real-time analysis during the elections, they found that negative sentiments about political parties prevailed before and after the elections. Moreover, they use sentiment analysis to determine how social media data can help to predict electoral results and found that the difference between the negative and positive tweets for political parties closely match the final of voting. The study clearly shows how this technique can be used to anticipate social phenomena and is indeed compatible with social, political and behavioral analysis.

3. REAL-TIME ANALYTICS IN COMMUNICATION AND AUDIENCE RESEARCH

As computational methods related to data mining evolve, and as automated and supervised sentiment analysis techniques focused on social media become more challenging, communication science and audience research are increasingly becoming more interested in applying these methods and techniques. Their huge potential lies in their ability to calibrate and build advanced real-time indicators of communicational content. Under this approach, researchers can imple-

ment *streaming analytics* by observing sentiment in sites like *Twitter* regarding specific users, current topics or *hashtags*, which allows for making predictions associated to social, political, economic, cultural processes related to relevant events. Previous studies have used historical Twitter data to make predictions, such as that of Choy, Cheong, Laik and Shung (2011) who applied dictionary-based sentiment analysis to predict the vote percentage that an individual candidate would receive in the Singapore presidential election of 2011. Bermingham and Smeaton (2011) also attempted to use *Twitter* to predict election results; they tried to predict the electoral outcome of the 2011 Irish general election using a supervised sentiment classifier. However, there are other attempts to make predictions using streaming analytics in Twitter. Whereas the historical sentiment analysis takes days, or even weeks, to be completed, real-time sentiment analysis in *Twitter* data delivers or returns results almost continuously and immediately. One of the pioneering works in this area was fostered by Wang, Can, Kazemzadeh, Bar and Narayanan (2012), who came up with a system for real-time twitter sentiment analysis of the 2012 US presidential election cycle. Their model instantly interprets results of how certain events can affect public opinion. Reducing the time gap between data collection and analysis is one of the pending challenges for Social Sciences, as one of the traditional critiques to our research points to a *decalage* between the point in time when things occur and the point in time when we, scholars, present our findings and concussions.

As we previously discussed, *SSA* is a central tool in order to shape positive and negative messages throughout their communication value chain. This method of analysis can also allow predicting sentiment in *real-time* communicational events, like political debates, or crisis communication management on various topics. However, there is a telling lack of real-time research and methodological applications in communication and audience studies, and even more so regarding sentiment analysis with *streaming* technologies based on supervised machine learning. Streaming analytics can be highly useful for audience and communication researchers (Pond, 2016; Bastos, Mercea & Charpentier, 2015; Driscoll, & Walker, 2014; Li & Xu, 2016; Kanejo & Tanai, 2016; Coletto et al., 2016), consulting firms and private enterprises in the fields of public opinion, marketing and political and governmental studies. Moreover, combined to SSA they can be helpful to study large amounts of *tweets*, by performing real-time sentiment analyses overcoming the limitations inherent to lexicon-based approaches. The aforementioned process to monitor *tweets* in streaming can improve communicational processes forecasts, and it can help in testing both the traditional and emerging approaches in public communicational opinion research requiring longitudinal data analysis in time series. This technique can also help in testing hypotheses in experimental studies requiring real-time entries to create or adapt stimuli, and thus validating exploratory theoretical models. These systems can bring comparative advantages for socio-political and communicational research. They can enable communication analysis teams to go a step ahead in terms of early detection and interpretation of the effectiveness of their communication strategies almost in real time.

In Python, communication and audience scholars can implement the library Tweepy and use few lines of code adapted from Kinsley (2017) to connect to the Twitter Stream and run SSA in real time to monitor sentiments through messages containing a specific hashtag (i.e. #Trump) with a specific level of confidence:

```
>>> from tweepy import Stream, OAuthHandler
>>> from tweepy.streaming import StreamListener
>>> import json
>>> import classifier as s
>>> ckey="", csecret="", atoken="", asecret=""
>>> class listener(StreamListener):
        def on_data(self, data):
            all_data = json.loads(data)
            tweet = all_data["text"]
            sentiment_value, confidence = s.sentiment(tweet)
            print(tweet, sentiment_value, confidence)
            if confidence*100 >= 80:
                output = open("twitter-out.txt","a")
                output.write(sentiment_value)
                output.write('\n')
                output.close()
            return True
        def on_error(self, status):
         print(status)
>>> auth = OAuthHandler(ckey, csecret)
>>> auth.set_access_token(atoken, asecret)
>>> twitterStream = Stream(auth, listener())
>>> twitterStream.filter(track=["#Trump"])
```

In sum, SSA can also be applied to monitor social media sites in real time. It takes advantage of the new computational methods, which simplify and streamline automatization processes, but as the same time has important implications for the elaboration of research designs in communication and audience studies.

4. DISTRIBUTED SSA FOR BIG DATA PROBLEMS

Bringing together large-scale supervised sentiment analysis and streaming analytics is a powerful approach to face big data challenges in communication research because it allows scholars to content analyze sentiments based on context with no restrictions of space (number of messages) and time (real-time and long periods with no interruption). In this section, we explain in detail how to overtake local computing limitations and use this computational method to face big data challenges using open source software with the data available from the Application Program Interface (API) of Twitter that nowadays provides freely

access to an important amount of semi-structured social data (Makice, 2009). We show how scholars can: (i) train machine learning models with parallelized algorithms using previously labeled tweets; (ii) connect to the streaming of Twitter and filter relevant messages; (iii) predict sentiments in real-time in a distributed context; and (iv) store the results in a *flexible* database and visualize the outcomes. This method requires a set of tools such as Python 3.5, Apache Spark 2.1, Apache Kafka 2.11, Apache Zookeeper 3.4.10 and MongoDB 3.4, and can be used in a local computer but is already prepared to scale up in the cloud in any academic or commercial service (such Amazon Web Services, AWS). A summary of the algorithms and tools to run supervised sentiment analysis in real time over big amount of tweets is shown in figure 1.

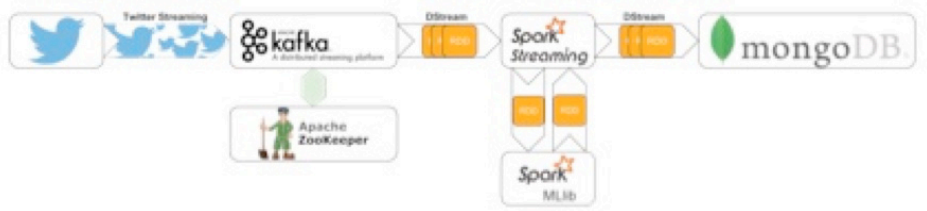

Figure 1. Tools and ML algorithms used in distributed supervised sentiment analysis of Twitter

Firstly, the most important step in supervised sentiment analysis is to obtain a good training dataset to train the models. Communication and audience researchers have a great experience in content analysis, which implies that labeling the initial dataset will be easy and labels will be reliable. To download raw tweets, scholars can use the Twitter API REST that allows the collection of recent messages in JSON format, filtering the search by any of the unstructured fields of the tweet (i.e. "text", "language", "location", etc.). Later, trained and independent coders must classify the messages (the "text" field) into negative or positive (or more specific labels such as "neutral", "strongly negative", etc.) according to traditional content analysis approach (Krippendorf, 2004; Neuendorf, 2016) to obtain adequate levels of reliability (>0,7 Krippendorf's Alpha or similar measures)[2]. There is not a minimum or maximum number of messages for the training dataset, but the algorithms will learn better if they have good

quality examples, or in other words, examples that really express that a message can be considered positive or negative. Each algorithm contains a specific bias (Kelleher, Mac Namee & D'Arcy, 2015), so the approach in supervised sentiment analysis is to use the one that is more predictive.

In a local computer, scholars can use the library Scikit-Learn in Python to train ML models (Raschka, 2015) using well known algorithms such as Original Naïve Bayes, Naïve Bayes for multimodal models, Naïve Bayes for multivariate Bernoulli models, Logistic Regression, Linear Support Vector Classification or Linear classifiers with stochastic gradient descent -SGD- training. However, if they have a huge training dataset that requires a lot of computational memory, they can train the ML models in a distributed context with Apache Spark (Pentreath, 2015) using the packages MLlib and PySpark. Not all algorithms are available in a parallelized format, but researchers can still implement Logistic Regression, Naïve Bayes and Support Vector Machines (SVM), and evaluated which gives the best accuracy. When working with unstructured data (like the text of a tweet) researchers must previously use natural language processing (NLP) techniques to tokenize words and convert messages into features vectors (Bird, Klein & Loper, 2009), which is the data that the algorithms will work with.

NLP includes extracting informative words based on a tag-of-speech approach, so researchers can decide which kind of word (verbs, adjectives, adverbs, etc.) will train the models. Once these words are selected, they must be grouped into an *array*, resulting in a sequence of words of the selected type, for example, if adjectives are selected, each tweet will be converted into an array of the adjectives of the tweet. This approach is more complex than the usual positive-or-negative word classification, as it considers sequences of words, which is useful, for example to identify degrees of positiveness and negativeness and more complex constructions such as the irony, that is impossible to identify with just the classification of words in negative or positive, as it usually has for example positive adjectives but in a specific sequence which really signifies a negative comment. This technique allows to not only considering the word but also the other ones that appears in the same tweet. The disadvantage is that much bigger training dataset is required. Going more into the detail of what is happening in the inside of the algorithm, every unique adjective present in every tweet must be registered in a dictionary of words. With this dictionary of unique words, each tweet can be transformed into an instance, an array of 1s and 0s, 1 when the word is present on the tweet and 0 otherwise. The result is a disperse array (an array with large amount of 0s and few 1s), and with as many independent variables as words in the dictionary. The next code shows how communication scholars can parallelize in Apache Spark the model training stage with the Naïve Bayes algorithm:

```
>>> import nltk, random, pyspark
>>> from nltk.tokenize import word_tokenize
>>> from nltk.classify.scikitlearn import SklearnClassifier
```

```
>>> import findspark
>>> findspark.init()
>>> sc = pyspark.SparkContext(appName="myAppName").
getOrCreate()
>>> spark = pyspark.sql.SparkSession(sc)
>>> sc._conf.getAll()
>>> allowed_word_types = ["JJ"]
>>> rdd_positive = sc.textFile("positive.txt")
>>> rdd_negative = sc.textFile("negative.txt")
>>> rdd_all_tokenized_words = rdd_positive.map(lambda tweet: (nltk.
pos_tag(word_tokenize(tweet)),1)).union(rdd_negative.map(lambda
tweet: (nltk.pos_tag(word_tokenize(tweet)),0)))
>>> rdd_selected_words = rdd_all_tokenized_words.map(lambda
review: \
([word[0] for word in review[0] if word[1] in allowed_word_
types],review[1]))
>>> rdd_all_words = rdd_selected_words.flatMap(lambda words:
words[0]).distinct()
>>> rdd_all_broadcast_words = sc.broadcast(rdd_all_words.collect())
>>> rdd_featured_instances = rdd_selected_words.map(lambda
instance: (find_features(instance[0]), instance[1]))
>>> def find_features(instance):
        features = []
        for word in rdd_all_broadcast_words.value:
          if word in instance:
            features.append(1)
          else:
             features.append(0)
        return features
>>> rdd_all_words.coalesce(1, True).saveAsTextFile("all_words")
>>> rdd_training_set = rdd_featured_instances.map(lambda instance:
LabeledPoint(label=instance[1], features=instance[0]))
>>> from pyspark.mllib.classification import NaïveBayes,
NaïveBayesModel
>>> from pyspark.mllib.util import MLUtils
>>> NB_model = NaïveBayes.train(rdd_training_set, 1.0)
```

Secondly, the Twitter API STREAMING allows instant access to 1% of the whole stream. This percentage might seem a small and an insignificant amount of tweets, but the true is that when we filter the tweets (i.e. only containing a specific hashtag, such as #Trump or #ClimateChange) we will not exceed this 1% limit in most of the cases. To locally connect to the stream, scholars can obtain the necessary code access (*API key, API secret, Access toke* and *Access token secret*) by registering to the Twitter API and use the library Tweepy for Pyhton (Roesslein, 2009) that simplifies the connection and works fine in a local compu-

ter. If communication and audience researchers are dealing with a huge amount of messages for a long period of time and decide to implement large-scale and real-time collection, this library still works but needs to be implemented in a remote machine in order to continuously produce the messages that will be analyzed afterwards. This process requires the execution of Apache Kafka (together with Zookeeper), so the researcher can remotely connect to the Twitter API and send the stream to a *kafka producer* in order to make data available from a *kafka broker*. This means that large-scale supervised sentiment analysis in real-time needs a virtual computer with Apache Kafka dedicated to be connected to the Twitter stream in order to continuously produce a new flux of parsed and filtered messages that can later be accessed from different nodes to execute the tone evaluation. In this stage, communication and audience researchers can monitor or even save the parsed tweets, which are a potential advantage if they decide to run any kind of exploratory, qualitative or on-the-way analysis for quick interventions. This code summarizes the distributed implementation of this *producer* in Spark for real-time tweets about #Trump:

```
>>> import json, tweepy, configparser
>>> from kafka import SimpleProducer, KafkaClient
>>> twitter_credentials = {“consumer_key”: “”, “consumer_secret”:
“”, “access_key”: “”, “access_secret”: “”}
twitter_parameters = {“hashtag”: [“#Trump”]}
>>> kafka_producer_parameters = {“batch_send_freq_t”:
1000,”batch_send_freq_n”: 10,”topic”: twitter_parameters[“hashtag”]
[0][1:],”connection_string”: “localhost:9092”}
>>> class TwitterStreamingListener(tweepy.StreamListener):
        def __init__(self, api, kafka_producer_parameters):
            self.api = api
            self.kafka_producer_parameters = kafka_producer_
    parameters
            super(tweepy.StreamListener, self).__init__()
            client = KafkaClient(kafka_producer_
    parameters[“connection_string”])
            self.producer = SimpleProducer(client, async = True,
                    batch_send_every_n = kafka_producer_
    parameters[“batch_send_freq_t”],
                    batch_send_every_t = kafka_producer_
    parameters[“batch_send_freq_n”])
        def on_status(self, status):
            msg =  status.text.encode(‘utf-8’)
            try:
                self.producer.send_messages(kafka_producer_
    parameters[“topic”], msg)
            except Exception as e:
                print(e)
```

```
                    return False
                return True
            def on_error(self, status):
                # Error in Kafka producer
                print(status)
                return True
            def on_timeout(self):
                print("Timeout on twitter API")
                return True
>>> twitter_parameters["hashtag"]
>>> auth = tweepy.OAuthHandler(twitter_credentials["consumer_
key"], twitter_credentials["consumer_secret"])
>>> auth.set_access_token(twitter_credentials["access_key"], twitter_
credentials["access_secret"])
>>> api = tweepy.API(auth)
>>> stream = tweepy.Stream(auth, listener =
TwitterStreamingListener(api, kafka_producer_parameters))
>>> stream.filter(track=twitter_parameters["hashtag"])
```

The third step for large-scale supervised sentiment analysis in real time is to predict the sentiments of the ongoing flux of messages we are producing; using the models we already trained with the parallelized algorithms. The models are kept as special files to save time, but they can be updated any time repeating step two and replacing the files. In this stage, researchers must run a set of virtual machines and implement an Apache Spark context if they really want to face big data challenges and keep the analysis fast and steady. Specifically, at this point the method consists in starting a streaming context in Spark to connect to the Kafka *producer* described above with a Kafka *consumer* to predict the score of tweet with the MLlib algorithms. To be able to classify the new tweets, these messages have to pass through the same process the training instances passed (tokenize words and select them based on tag of speech). To work in real time, all the tweets must also be gotten from the Kafka stream and save the texts to a DStream, a sequence of resilient distributed datasets (RDDs) for each period of time, which is the primary data structure of Apache Spark. All these phases allow executing the sentiment prediction in parallel instances, using the model and the word list produced with the ML algorithms during the training stage, and printing each prediction with standard output. Parallel computing and elastic computing capacities (most of academic and commercial services can be configured to add/remove virtual machines automatically based on demand) give communication and audience scholars a unique opportunity to overtake space and time limitations that traditional and "small data" computational methods have when dealing with sentiment analysis. This is a simplified code for the *consumer* with a call to three ML algorithms (Logistic Regression, Support Vector Machines and Naïve Bayes) and an order to choose which is the best to classify the tweets:

```
>>> import findspark, pyspark, os, datetime, nltk, random
>>> findspark.init()
>>> from pyspark.sql.functions import lit
>>> from pyspark import SparkConf
>>> SUBMIT_ARGS = "--packages org.apache.spark:spark-
streaming-kafka-0-8-assembly_2.11:2.1.0,org.mongodb.spark:mongo-
spark-connector_2.10:2.0.0 pyspark-shell"
>>> os.environ["PYSPARK_SUBMIT_ARGS"] = SUBMIT_ARGS
>>> conf = (SparkConf().set("spark.mongodb.input.uri", "mongodb://
localhost:27017/twitter.tests").set("spark.mongodb.output.uri",
"mongodb://localhost:27017/twitter.tests"))
>>> sc = pyspark.SparkContext(appName="streaming_app",
conf=conf).getOrCreate()
>>> spark = pyspark.sql.SparkSession(sc)
>>> from pyspark.streaming import StreamingContext
>>> ssc = StreamingContext(sc, 10)
>>> kafka_configuration_params = {"topic":
["BigData"],"connectionstring": "localhost:9092"}
>>> from pyspark.streaming.kafka import KafkaUtils
>>> directKafkaStream = KafkaUtils.createDirectStream(ssc,
kafka_configuration_params["topic"],{"metadata.broker.list": kafka_
configuration_params["connectionstring"]})
>>> from pyspark.mllib.classification import  NaïveBayesModel
>>> classif_LR_model = LogisticRegressionModel.load(sc, "LR_
model")
>>> classif_SVM_model = SVMModel.load(sc, "SVM_model")
>>> classif_NB_model = NaïveBayesModel.load(sc, "NB_model")
>>> LR_model = classif_LR_model
>>> SVM_model = classif_SVM_model
>>> NB_model = classif_NB_model
>>> from nltk.tokenize import word_tokenize
>>> allowed_word_types = ["JJ"]
>>> rdd_all_words = sc.textFile("all_words/part-00000")
>>> rdd_broadcast_all_words = sc.broadcast(rdd_all_words.collect())
>>> def convert_tweet_to_instance(tweets):
        rdd_tweets = tweets.map(lambda tweet: [word[0] for word in
        nltk.pos_tag(word_tokenize(tweet)) if word[1] in allowed_
        word_types])
        rdd_instances = rdd_tweets.map(lambda instance: find_
    features(instance))
        return rdd_instances
>>> def find_features(instance):
            features = []
            for word in rdd_broadcast_all_words.value:
              if word in instance:
```

```
                    features.append(1)
                else:
                    features.append(0)
            return features
>>> rdd_input = directKafkaStream.map(lambda output: output[1])
>>> classification = convert_tweet_to_instance(rdd_input).map(lambda
instance: 1 if (LR_model.predict(instance) + int(NB_model.
predict(instance)) + SVM_model.predict(instance))>=1 else -1)
>>> classification_each = convert_tweet_to_instance(rdd_input).
map(lambda instance: [LR_model.predict(instance), NB_model.
predict(instance), SVM_model.predict(instance)])
>>> def save_to_db(rdd, collection):
        df = rdd.zipWithUniqueId().toDF().withColumn('timestamp'
        ,lit(datetime.datetime.utcnow())).toDF('label','in_batch_id',
        'timestamp')
        df.write.format("com.mongodb.spark.sql.DefaultSource").
        mode("append").option("database","twitter").
        option("collection",collection).save()
>>> classification.foreachRDD(lambda rdd: save_to_db(rdd,"labels"))
>>> rdd_input.foreachRDD(lambda rdd: save_to_db(rdd, "tweets"))
>>> ssc.start()
```

Finally, the results of distributed supervised sentiment analysis must be stored and visualized. When working with streaming analytics, all results are stored by periods of time, printing each *batch* separately. The first way to store the tweets and their predicted value (and any other extra data scholars wish to keep for further research) is to save the results in plain text files (tab/comma and line separated). However, even when social scientists might be still unfamiliar with non-relational databases, the best way to store Twitter messages and their predicted sentiments is through a NoSQL database, either SQL or NoSQL, depending on the data, such as PostgreSQL or MongoDB, for example. In contrast with relational databases, these NoSQL databases are more flexible and less restrictive, which allows scholars to scale up their analysis when dealing with big data. The above code described for the *consumer* includes commands for distributed storage and the flow of all this process is described in figure 2.

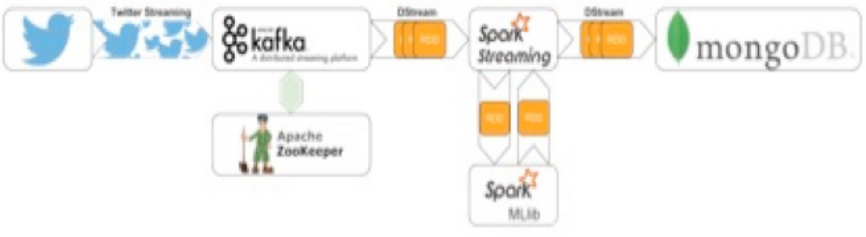

Figure 2. Flow of data during distributed and real-time supervised sentiment analysis of Twitter

On the other hand, scholars can use different Pyhton libraries such as Matplotlib or Bokeh, to visualize the ongoing sentiments. In particular, Bokeh is a powerful tool to visualize the trends if we have stored the results in MongoDB. All scripts will visualize batches of tweets, but scholars can control parameters such as the update frequency (time to wait to poll the database for the next batch) or the number of points that can be simultaneously shown on the screen (to produce averages of minutes, hours, days, months, etc.). Control of parameters and flexible visualization of large-scale sentiments on Twitter can enhance social research designs especially when scholars need to monitor big amounts of tweets as fast as they are produced. Figure 3 shows a visualization of distributed sentiment analysis of tweets containing the hashtag "#Trump" in real time, and the next code show scholars how to execute this task with Bokeh reading from MongoDB:

```
>>> import time, pymongo, pprint
>>> import numpy as np
>>> from bokeh.models.sources import ColumnDataSource
>>> from bokeh.plotting import figure
>>> from bokeh.io import output_notebook, show, push_notebook
>>> from datetime import datetime, timedelta
>>> from bson.objectid import ObjectId
>>> SERVER_URL = "mongodb://localhost:27017"
>>> client = pymongo.MongoClient(SERVER_URL)
>>> db = client.twitter
>>> coll = db.labels
>>> def compute_batch_score(batch):
        score = 0
        for result in batch:
            score = score + result["label"]
        return score
>>> source = ColumnDataSource(dict(x=[], y=[]))
>>> my_figure = figure(plot_width=800, plot_height=400)
>>> my_figure.line(source=source, x="x", y="y", line_width=2,
alpha=.85, color='blue')
>>> handle = show(my_figure, notebook_handle=True)
>>> new_data = dict(x=[0], y=[0])
>>> x = []
>>> y = []
>>> step = 0
>>> period = 2  # in seconds
>>> n_show = 300  # number of points to keep and show
>>> timenow = datetime.utcnow() - timedelta(hours=2, seconds=10)
>>> while True:
        batch = coll.find({'timestamp':{'$gt': timenow}}).
```

```
sort([("timestamp", -1)])
    latest_value = new_data['y'][0]
    new_data = dict(x=[step], y=[latest_value + compute_batch_
score(batch)])
    source.stream(new_data, n_show)
    push_notebook(handle=handle)
    step += 1
    timenow = datetime.utcnow() - timedelta(hours=2,
seconds=10)
    time.sleep(period)
```

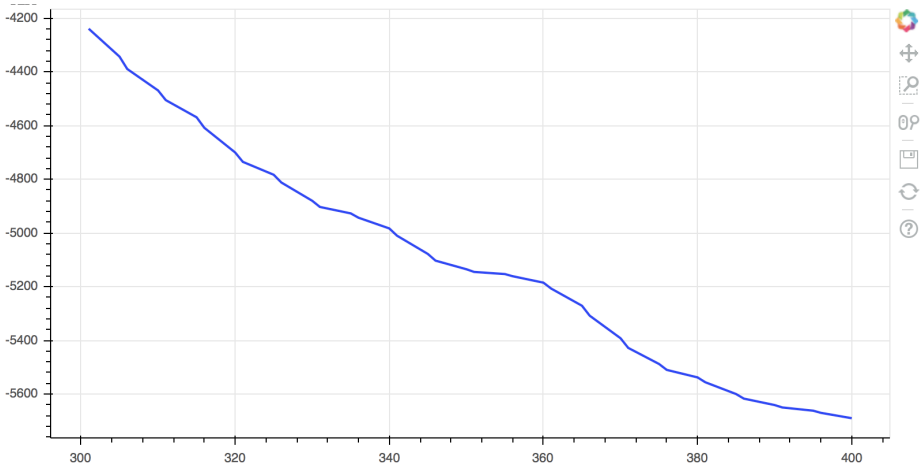

Figure 3. Streaming visualization of supervised sentiment analysis to tweets containing "#Trump"

5. DISCUSSION AND CONCLUSIONS

In this paper we have explained the relevance of *supervised sentiment analysis* and *streaming analytics* in communication and audience research, and have described the implementation of *distributed supervised sentiment analysis*, a computational method that allow communication scholars and social scientists to face the *big data* challenge of Twitter contents. This method overtakes the disadvantages of other approaches orientated to computationally manage *small data* in social media and gives communication and audience researchers a fair overview of how cutting-edge technologies can be adapted to computational social science.

The described method and its implementation is an open source free of charge solution (except for the financial costs of the cloud computing service such as AWS), giving the scholars the absolute power to adapt and modify all parameters. However, there are some commercial products of companies such

as Microsoft Azure or IBM Bluemix, that might also help scholars to scale their *distributed supervised sentiment analysis* of tweets with more friendly interfaces. Microsoft Azure nowadays offers a sentiment analysis service built on the MPQA Subjectivity Lexicon (Wilson, Wiebe & Hoffmann, 2005), but can also run scalable sentiment analysis with a machine learning algorithm (Two-Class Support Vector Machine) written in R to predict opinion polarity in tweets. Microsoft Azure runs real time sentiment analysis of tweets with Azure Stream Analytics, but uses Sentiment140, an external classification method using machine learning algorithms (Go, Bhayani & Huang, 2009). On the other hand, IBM Bluemix can run distributed sentiment analysis based on Spark (and Kafka), and can as well connect to their own machine learning algorithm Watson Tone Analyzer, built on neural networks to predict emotions (i.e. anger, fear, joy, sadness, disgust), social tendencies (i.e. openness, conscientiousness, extraversion, agreeableness, emotional range) and writing style (i.e. confident, analytical, tentative) on the texts. These services on Microsoft Azure and IBM Bluemix are mostly available for tweets written only in English. All these services are great for unexperienced users that want to use already implemented solutions with the restrictions each solution provides, the algorithms available and assuming the cost of them. Even more friendly, they are still away from most of Audience and Communication scholars research tooolkits, pointing out the need to combine conventional research methods with the innovations being developed in the computational social sciences and digital humanities. This window is open and a wider instruction of social scientists in these fields of knowledge is still required, as a way to increase our research effectiveness. Analyzing tweets is an open door to explore what a given audience daily expresses, so getting closer to these messages is a way to reduce the distance between what people think and do in the public domain.

However, the approach we present on this paper is to create a custom environment that can be deployed on an academic cloud computing service or in-house cluster, as well as in any other commercial provider such as Amazon Web Services, Microsoft Azure or IBM Blue Mix, using just the virtual machines they provide. The main advantage of this approach is that the scholars can deploy their own code and algorithms, and also scale and control the amount of resources they need, which will have an impact on the cost of the study. Nevertheless, some knowledge on how to deploy the virtual machines, and configure them in order to use the whole system is required, which can be a difficult task for beginners in deployment operations.

Regarding the limitations of this paper, we stress that only three algorithms were used for the distributed supervised sentiment analysis in Apache Spark, and there was no probability classification, which, compared to the local mode classification with Scikit-Learn, is a disadvantage, as the user cannot control the minimum degree of certainty to classify each tweet. Some future development can solve some of the previously explained limitation. For example, it is necessary to introduce more degrees for classification or probabilistic results on predictions. On the other hand, a major challenge is to simplify the deployment

of the whole system for the distributed analysis, which is another main limitation of the present study. Other interesting points to consider can be the deployment of the prediction models as microservices, as the data amount does not usually require a Spark Streaming system and can be solved with a REST API in which the tweet stream will be classified. This approach is interesting as it can simplify the deployment and reduce the amount of resources used for the analysis.

However, *distributed supervised sentiment analysis* represents an innovative computational method in communication science that can be adopted by different actors such as:

Communication and audience researchers: scholars can use this technique to carry out descriptive, cross-sectional, longitudinal and experimental studies.

Firms engaged in public opinion studies: these companies may use the classifier to monitor opinion trends on communicational topics. In addition, they can use this methodology to come up with advanced indicators of socioeconomic and political state of the art within the digital society, notoriety of brands´, messages´, candidates' "popularity and acceptance", among others.

Communication consulting firms: these firms can use the classifier to supplement their supervision and study of communicational facts that take place in the country, in order to plan communication and political persuasion strategies (brand, corporate or political marketing), particularly when sentiment analysis and sentiment interpretation arising from communicational campaigns associated to investment opportunities, political actions, social campaigns and initiatives associated may require proactive real-time analysis and conclusions for increased effectiveness.

Political parties: political parties can use this technique to plan their actions according to the existing sentiment on certain public topics on investment opportunities and political candidates suitability.

The challenges and opportunities seized by *big data* solutions, and particularly by *distributed supervised sentiment analysis* in Audience or Communication Research, using the abovementioned techniques, leads to the conclusion that we are witnessing a Data-Analysis-Revolution (DAR) regarding the analysis, interpretation and management of communication initiatives run by stakeholders in the digital communication value chain of fully connected society. The weaknesses and threats, and more importantly the strengths and opportunities, provided by these methodologies and techniques to communication and audience researchers anticipate a deeper revolution in the scientific analysis of communication processes with the establishing of a *computational communication science*, similar to other fields such as computational biology, where computational methods have transformed the whole discipline. Under this idea, it is highly relevant that *computational communication scholars* incorporate to their routines the conceptual and technical skills explained in this article.

Distributed supervised sentiment analysis of Twitter messages is a supplementary but necessary computational method to test and predict communication patterns. In a context where the quality of communication, election or branding research is subject to constant scrutiny and re-assessment by experts and public

opinion based on their "inaccuracy," this approach may appears as a "3D scanner in real time" that can complement traditional analyses of representative samples to predict communicational patterns. Additionally, real time supervised sentiment analysis of messages for long time periods in *Twitter* and other digital social networks will enable longitudinal analyses to detect changes in topics such as "communicational incidences" and the various variables examined throughout the relevant time period for a given subject-topic. Accordingly, one could test the relationship between these changes and specific events, which would also allow implementing policies based on research and data analysis on a more proactive basis and with greater time reactivity. Large-scale algorithm patterns and the understanding of their internal operation is a key issue for the development and implementation of these methodologies within the big data landscape. The interpretation and grasping of the computational method workflow by the researcher may be of vital importance for the further understanding of the complex decision taken process which may arise. *Distributed supervised sentiment analysis* may provide for the time being that extra luminescence for the understanding of complex communicational processes which in our today technological scenario require still the ability and expertise of human-brained supervision.

6. ACKNOWLEDGEMENTS

The authors wish to express their gratitude to *Fundación Universidad de Salamanca* and *Plan TCUE* [2015-2017 Fase 2] for funding the project "Clasificador en tiempo real de opiniones políticas en español con técnicas de aprendizaje automático (Auto-cop)", in which we the developed all the methods and codes explained in this paper. We also thank Javier Ramírez, Javier Jiménez Amores and Sofía Trullenque for their valuable support.

7. REFERENCES

Arcila, C.; Ortega, F.; Jiménez, J. & Trulleque, S. (2017). Análisis supervisado de sentimientos políticos en español: Clasificación en tiempo real de tweets basada en aprendizaje automático. *El Profesional de la Información*, 26 (5), 978-987.

Bakliwal, A., Foster, J., van der Puil, J., O'Brien, R., Tounsi, L., & Hughes, M. (2013, June). Sentiment analysis of political tweets: Towards an accurate classifier. pp. 49-58. *Association for Computational Linguistics*.

Bakliwal, A., Arora, P., Madhappan, S., Kapre, N., Singh, M., & Varma, V. (2012). Mining sentiments from tweets. *Proceedings of the WASSA*, *12*. pp. 11-18.

Bastos, M. T., Mercea, D., & Charpentier, A. (2015). Tents, tweets, and events: The interplay between ongoing protests and social media. *Journal of Communication*, 65(2), 320-350. doi: 10.1111/jcom.12145

Bermingham, A., & Smeaton, A. F. (2011). On using Twitter to monitor political sentiment and predict election results. In: *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP), IJCNLLP 2011*, pp. 2-10.

Bermingham, A., & Smeaton, A. F. (2010, October). Classifying sentiment in microblogs: is brevity an advantage?. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 1833-1836). ACM.

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

Bollen, J., Mao, H., & Pepe, A. (2011). Modelling public mood and emotion: Twitter sentiment and socio-economic phenomena. *ICWSM, 11*, 450-453.

Cambria, E., Schuller, B., Liu, B., Wang, H., & Havasi, C. (2013). Knowledge-based approaches to concept-level sentiment analysis. *IEEE Intelligent Systems*, 28(2), 12-14. doi: 10.1109/MIS.2013.45

Choy, M., Cheong, M. L., Laik, M. N., & Shung, K. P. (2011). A sentiment analysis of Singapore Presidential Election 2011 using Twitter data with census correction. Report in *arXiv preprint arXiv:1108.5520*.

Cobb, W. N. W. (2015). Trending now: using big data to examine public opinion of space policy. *Space Policy*, 32, 11-16. doi: 10.1016/j.spacepol.2015.02.008

Coletto, M., Esuli, A., Lucchese, C., Muntean, C. I., Nardini, F. M., Perego, R., & Renso, C. (2016, August). Sentiment-enhanced multidimensional analysis of online social networks: perception of the mediterranean refugees crisis. In *Advances in Social Networks Analysis and Mining (ASONAM)*, 2016 IEEE/ACM International Conference on (pp. 1270-1277). IEEE.

Driscoll, K., & Walker, S. (2014). Big data, big questions| working within a black box: Transparency in the collection and production of big twitter data. *International Journal of Communication*, 8, 20. doi: 1932–8036/20140005

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4), 82-89.

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

Kaneko, T., & Yanai, K. (2016). Event photo mining from twitter using keyword bursts and image clustering. *Neurocomputing*, 172, 143-158. doi: 10.1016/j.neucom.2015.02.081

Kelleher, J. D., Mac Namee, B., & D'Arcy, A. (2015*). Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. Cambridge, MA: MIT Press.

Kinsley, H. (2017). PythonProgramming. https://pythonprogramming.net/

Kranjc, J., Smailović, J., Podpečan, V., Grčar, M., Žnidaršič, M., & Lavrač, N. (2015). Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the ClowdFlows platform. *Information Processing & Management*, *51*(2), 187-203. doi: 10.1016/j.ipm.2014.04.001

Krippendorff, K. (2004). *Content analysis: An introduction to its methodology.* Thousand Oaks, CA: Sage.

Leetaru, K. (2012). *Data mining methods for the content analyst: An introduction to the computational analysis of content*. New York: Routledge.

Li, J., & Xu, H. (2016). Suggest what to tag: Recommending more precise hashtags based on users' dynamic interests and streaming tweet content. *Knowledge-Based Systems*, 106, 196-205. doi: 10.1016/j.knosys.2016.05.047

Madlberger, L., & Almansour, A. (2014, November). Predictions based on Twitter—A critical view on the research process. In *Data and Software Engineering (ICODSE),*

2014 International Conference on (pp. 1-6). IEEE. pp. 1-6. doi: 10.1109/ICOD-SE.2014.7062667

Makice, K. (2009). *Twitter API: Up and running: Learn how to build applications with the Twitter API*. Sebastopol, CA: O'Reilly Media, Inc.

Neuendorf, K. A. (2016). *The content analysis guidebook*. Thousand Oaks, CA: Sage.

Nodarakis, N., Sioutas, S., Tsakalidis, A. K., & Tzimas, G. (2016, March). Large Scale Sentiment Analysis on Twitter with Spark. In *EDBT/ICDT Workshops* (pp. 1-8).

O'Connor, B., Balasubramanyan, R., Routledge, B. R., & Smith, N. A. (2010). From tweets to polls: Linking text sentiment to public opinion time series. *ICWSM, 11*, pp. 122-129.

Pang, B., Lee, L., & Vaithyanathan, S. (2002, July). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing,* Volume 10, pp. 79-86. Association for Computational Linguistics.

Pond, P. (2016). Twitter time: A temporal analysis of tweet streams during televised political debate. *Television & New Media*, 17(2), 142-158. doi: 10.1177/1527476415616190

Preethi, P. G., & Uma, V. (2015). Temporal sentiment analysis and causal rules extraction from tweets for event prediction. *Procedia Computer Science*, *48*, 84-89. doi: 10.1016/j.procs.2015.04.154

Raschka, S. (2015). *Python machine learning*. Birmingham: Packt Publishing Ltd.

Pentreath, N. (2015). *Machine Learning with Spark*. Birmingham: Packt Publishing Ltd.

Roesslein, J. (2009*). Tweepy. An easy-to-use Python library for accessing the Twitter API*. Retrieved from http://www.tweepy.org/

Shahin, S. (2016) When Scale Meets Depth: Integrating Natural Language

Processing and Textual Analysis for Studying Digital Corpora. *Communication Methods and Measures*, 10(1), 28-50, doi: 10.1080/19312458.2015.1118447

Sluban, B., Smailović, J., Battiston, S., & Mozetič, I. (2015). Sentiment leaning of influential communities in social networks. *Computational Social Networks*, *2*(1), 1-21. doi: 10.1186/s40649-015-0016-5

Smailović, J., Kranjc, J., Grčar, M., Žnidaršič, M., & Mozetič, I. (2015, October). Monitoring the Twitter sentiment during the Bulgarian elections. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on* (pp. 1-10). IEEE. doi: 10.1109/DSAA.2015.7344886

Smailović, J., Grčar, M., Lavrač, N., & Žnidaršič, M. (2013). Predictive sentiment analysis of tweets: A stock market application. In *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data* (pp. 77-88). Springer Berlin Heidelberg.

Spark Kafka Integration (2016). Spark Streaming + Kafka Integration Guide. Available at: http://spark.apache.org/docs/latest/streaming-kafka-integration.html

Turck, M. & Hao, J. (2016). The Chart of the Big Data Landscape 2016 (Version 3.0). Available at: http://mattturck.com/big-data-landscape-2016-v18-final/

Turney, P. D. (2002, July). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 417-424). Association for Computational Linguistics.

van Zoonen, W., & Toni, G. L. A. (2016). Social media research: The application of supervised machine learning in organizational communication research. *Computers in Human Behavior*, 63, 132-141. doi: 10.1016/j.chb.2016.05.028

Vinodhini, G., & Chandrasekaran, R. M. (2012). Sentiment analysis and opinion mining: a survey. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(6), 282-292.

Wang, H., Can, D., Kazemzadeh, A., Bar, F., & Narayanan, S. (2012, July). A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations* (pp. 115-120). Association for Computational Linguistics.

Wilson, T., Wiebe, J. & Hoffmann, P. (2005). Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 347-354). Association for Computational Linguistics.

Endnotes

2 For trial, the models can be trained with positive/negative movie reviews in English of IMDB provided by NLTK or freely available in the Internet.