Noname manuscript No.
(will be inserted by the editor)

An Evaluation of Sentiment Analysis for Mobile Devices

Johnnatan Messias · João P. Diniz · Elias Soares · Miller Ferreira · Matheus Araújo · Lucas Bastos · Manoel Miranda · Fabrício Benevenuto

Received: date / Accepted: date

Johnnatan Messias
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
E-mail: johnnatan@dcc.ufmg.br

João P. Diniz
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
E-mail: jpaulo@dcc.ufmg.br

Elias Soares
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
E-mail: eliassoares@dcc.ufmg.br

Miller Ferreira
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
E-mail: millermarques@dcc.ufmg.br

Matheus Araújo
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
E-mail: matheus.araujo@dcc.ufmg.br

Lucas Bastos
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
E-mail: lucasmbastos@dcc.ufmg.br

Manoel Miranda
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
E-mail: manoelrmj@dcc.ufmg.br

Fabrício Benevenuto
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brazil
E-mail: fabricio@dcc.ufmg.br

Abstract Sentiment Analysis has become a key tool to extract knowledge from data containing opinions and sentiments, particularly, data from online social systems. With the increasing use of smartphones to access social media platforms, a new wave of applications that explore sentiment analysis in the mobile environment is beginning to emerge. However, there are various existing sentiment analysis methods and it is unclear which of them are deployable in the mobile environment. In this paper, we provide the first of a kind study in which we compare the performance of 14 sentence-level sentiment analysis methods in the mobile environment. To do that, we adapted these methods to run on Android OS and then we measure their performance in terms of memory, CPU, and battery consumption. Our findings unveil methods that require almost no adaptations and run relatively fast as well as methods that could not be deployed due to excessive use of memory. We hope our effort provides a guide to developers and researchers interested in exploring sentiment analysis as part of a mobile application and can help new applications to be executed without the dependency of a server-side API. We also share the Android API that implements all the 14 sentiment analysis used in this paper.

Keywords Sentiment Analysis · Performance Evaluation · Mobile · Android

1 Introduction

The adoption of smartphones and online social networks has experienced a dramatic growth in the last years. On one hand, the number of people that own a smartphone increased from 49 million to 189.7 million from May 2010 to May 2015, according to ComScore [7]. Furthermore, a few online social systems such as Facebook and YouTube have more than 1 billion active users, which corresponds to one-third of all people on the Internet [3]. Particularly, Facebook registered in one month more than 1.25 billion mobile active users, which represents an increase of 24 percent year-over-year [2].

A large amount of information shared on mobile devices has opened space for a new wave of mobile applications. Particularly, sentiment analysis has become an extremely popular tool employed in various analytic domains, especially in the Web and social media. More than $7,000$ articles have been written about sentiment analysis on the last decade, and various startups are developing tools and strategies to extract sentiments from text [17]. Most applications of sentiment analysis are focused on monitoring the reputation or opinion of a company or a brand with the analysis of reviews of consumer products or services [21]. Sentiment analysis can also provide analytic perspectives for financial investors that want to discover and respond to market opinions [6, 33]. Another important application is in politics, where marketing campaigns are interested in tracking opinions expressed by voters associated with candidates for election [44]. There is also a large potential for sentiment analysis methods for mobile environments and most of the effort has been concentrated on the development of approaches able to measure the well-being of a smartphone user [9, 15, 25]. There are many other applications, for example, to help users organize the information they read and even analyze opinions extracted from the content they receive [38], sentiment analysis of closed caption (in smart TVs), input text from

users like keyboards, etc. Thus, our work provides guidelines for mobile developers and App companies as we identify which method is more suitable for mobile context once they are starting to see sentiment analysis as a valuable tool for incorporate value in the design of novel software.

Overall, all the above techniques are acceptable by the research community and it is common to see in a single computer science conference papers that use completely different sentiment analysis methods. Particularly, a recent effort has shown that there is no single method that always achieves the best prediction performance for different datasets [19]. Thus, Ribeiro *et al.* [40] have compared 22 methods in 18 different labeled datasets showing that there is no single method capable of achieving the best prediction performance for all different datasets. This means the results could be interpreted differently depending on the sentiment method choice. They also show the methods are more accurately for classifying positive than negative text. The choice of a sentiment method is usually related with the kind of data used. For this reason, there are many "state-of-the-practice" sentiment analysis. However, it is still unclear which of these methods are suitable for the mobile environment as existing efforts are concentrated only on measuring the prediction capability (i.e. accuracy) of existing methods [19, 22]. Therefore, the focus of our work is not to show which method is more accurate. Instead, we show which one of them works better on mobile devices. Also, it is important to highlight that our work focuses on "off-the-shelf" methods as they have been extensively and recently used.

This paper aims at filling this research gap. Instead of evaluating the abilities of sentiment analysis methods to detect positive and negative messages, we focus on measuring the performance of existing popular methods to investigate the feasibility of deploying them as part of mobile applications. To do that, we have implemented 16 sentence-level sentiment analysis methods: AFINN [31], Emolex [29], Emoticons [14], NRC Hashtag Sentiment Lexicon [28], OpinionFinder (MPQA) [47], OpinionLexicon [21], Panas-t [20], SASA [45], Sentiment140 Lexicon [30], SenticNet [10], SentiStrength [43], SentiWordNet [16], Stanford Recursive Deep Model [41], Umigon [24], USent [34], and Vader [23]. All these methods are implemented in the iFeel system [5] and are described in [40]. However, Sentiment140 Lexicon and SentiWordNet were not able to run on Android. Thus, we filter them out of our analysis and now we focus on 14 sentiment analysis methods. Then, we evaluate them running each method with an input of 10, 100, $1k$, and $10k$ tweets in English according to the key performance metrics, such as memory usage, battery consumption and run time. To measure these metrics we developed an application that customized the OSMonitor Application, a popular open-source performance monitor available on Google Play.

Our main findings show that it might be hard to use NRCHashtag, OpinionLexicon, SASA, Stanford Recursive Deep Model, and USent on current mobile devices. Another observation is that the lexical methods (e.g. AFINN, Emoticons, Emolex) had a good performance in terms of memory, CPU, and battery usage once they are based on dictionaries to compute the sentiment polarity scores. In general, the machine learning methods obtained the worst results.

We hope our effort can help developers and researchers interested in exploring sentiment analysis as part of a mobile application, instead of creating applications that depend on server-side APIs. Thus, we plan to release the Android API that im-

plements all the 14 sentiment analysis methods as part of a set of resources related to the iFeel system [5]. More specifically, our codes will be shared on a request base at `http://www.ifeel.dcc.ufmg.br/`.

The remaining of this paper is organized as follow. Section 2 briefly presents some related effort. Section 3 introduces the sentiment analysis methods used in our work and it also describes the adaptations required to run these methods on Android OS. Section 4 describes the experimental setup and the results of the experiments. Finally, in Section 5, we present the conclusion and some future directions of our work.

2 Related Work

There has been a considerable amount of efforts towards the development of new sentiment analysis methods, which typically focuses solely on providing evaluations of the predictive power of the new method. Particularly, a novel method called Vader [22] provides an evaluation of 7 methods to show that Vader surpasses all of them in the tested twitter datasets. Another effort compares eight well-known methods [19] using different datasets extracted from online social systems. More recently, Ribeiro *et al.* [40] have extened this effort and compared 22 methods in 18 different labeled datasets showing that there is no single method capable of achieving the best prediction performance for all different datasets. Overall, these efforts highlight a bias towards positive values for most of the methods and also show that there is no "silver bullet" method. In other words, they show that there is not a method that is better in all tested datasets. Our work is complementary to the above efforts as we provide a performance evaluation of them in the mobile environment.

Moving to the sentiment-aware mobile applications, the literature is still scarce on comprehending how sentiment analysis methods behave natively on mobile devices. Thus, the use of client-side applications that send data and process it remotely is usually the strategy adopted. Several efforts use this approach to infer human emotions from mobile data. Particularly, the first demonstration of automatic mood inference using collected smartphone data was presented by LiKamWa *et al.* [25]. They presented initial findings based on the analysis of data collected from a longitudinal field study involving 25 iPhone users. They proposed a model and studied potential mood-enhanced mobile applications. More recently, Chambers *et al.* [13] developed a sentiment analysis Android Application for SMS messages allowing the user to know about their own mood and stress level.

Despite the existing efforts on using mobile data to infer sentiment-related issues, we are not aware of efforts that approached the specific idea of providing a performance evaluation of several sentiment analysis on mobile devices. The most closely related effort [13] contrasts load time, storage space and peak memory usage of a single sentiment analysis method in a desktop and in a mobile implementation. So, their effort basically consists of evaluating the performance of the *Part-Of-Speech* library on different implementations (i.e. they compared the libraries StanfordNLP and OpenNLP).

Our study is the first of a kind effort that compares the performance of a number of sentiment analysis methods in the mobile environment. Thus, our effort is complementary to existing previous efforts.

Unlike the literature, our focus is to evaluate the sentiment analysis methods' execution in mobile devices. Therefore, it was necessary to collect some information such as CPU, memory, and battery usage from the device. Our aim is to show which method runs better in Android devices.

Our approach focuses on sentence-level sentiment analysis since the intention is to explore how faster and dynamic sentiment-aware application can run without depending on Internet connection and server-side processing. Our focus is on "off-the-shelf" methods [37] that can be used in an unsupervised way in mobile devices. Purely supervised methods (e.g. Canuto *et al.* [11]) are out of scope of our effort.

It is important to highlight that this work is an extension of short short paper [26]. Our current manuscript provides much more detailed analysis and many new findings.

3 Sentiment Analysis Methods and Android Adaptation

There are different problems approached within the field of sentiment analysis, including sentence-level, document-level, and aspect-level sentiment analysis. Particularly, given the recent popularity of Online Social Networks and of short texts on the Web, many methods are focused on detecting sentiments at the sentence-level, usually used to measure the sentiment of small sets of sentences in which the topic is known *a priori*. This is the focus of our work.

In Literature, there is a large number of methods for sentiment analysis that rely on different techniques from different fields of Computer Science. For instance, many of methods are based on machine learning techniques. In this category, a sentiment analysis model is built based on a training process given human labeled data, this model will eventually be used to classify a new sentence.

Although huge effort has been done to select which machine learning algorithm is the best, most of the challenge is still choosing an effective set of features that better describes the sentiment analysis problem. As example of current used features in many machine learning based methods are negation (*not*, *but*), opinion words (*good*, *beautiful*, *bad*), part of speech (specially adjectives) and the frequency of terms. In general machine learning has shown good results [40], but it has some counterparts. The requirement of a large human labeled dataset to perform a reasonable training phase is time-consuming. The labeling process has its own challenges since Ogneva *et al.* [32] shows that in general humans agree with themselves in only 79% of the sentence's label. Therefore, it requires many humans to label the same sentence. Despite all the potential for the use of machine learning sentiment analysis as part of mobile applications, little is known about deployability of these methods in the mobile environment.

The lexical-based approaches are another category of sentiment analysis methods, which make use of predefined dictionaries of words, in which each word is associated with a specific sentiment. The lexical methods vary according to the context in which they were created. For instance, LIWC [42] was originally proposed to ana-

lyze sentiment patterns, informally written, in English texts, whereas PANAS-t [20] is a psychometric scale adapted to the Web context. Other techniques for sentiment analysis include deep-learning [41] and other efforts that rely on natural language processing techniques [10].

The exploited methods in our effort are open-source and freely available, except for SentiStrength, which is available only for research purposes. Although some of these methods rely on machine learning techniques, they are *pre-trained* and can be used as an "off-the-shelf" methods as they are used in practice, increasing enormously their applicability for many situations. This section provides a brief description of sentiment analysis methods that are evaluated in this paper. Table 1 shows the 16 sentiment analysis methods used in our performance analysis.

AFINN [31]. It is a public available word list motivated by some classic lexicons such as ANEW [8] and General Inquirer. They use Amazon Mechanical Turk to label several word lists like the Original Balanced Affective Word List and Internet slangs from Urban Dictionary and obscene words. AFINN classifies messages in a range of $[-5, 5]$, with -5 and 5 being the most negative and most positive score, respectively. We implement an approach to read the lexicon dictionary in Java and then adapted the code to run it on Android.

Emolex [29]. Emolex or NRC Emotion Lexicon is a lexical method with up to $10,000$ word-sense pairs. Each entry lists the association of a word-sense pair with 8 basic sentiments: joy, sadness, anger, fear, trust, disgust, anticipation, and surprise, defined by [36]. Besides these sentiments, we consider the polarity positive/negative also given by the lexicon. We use NRC Emotion Lexicon version 0.92, which is available. We have used a Java/Android implementation to read its lexicon dictionary.

Emoticons [14]. It is the simple way to detect the polarity (i.e., positive and negative affect) of a message based on the emoticons it contains. Emoticons are primarily face-based and represent happy or sad feelings, although a wide range of non-facial variations exists: for instance, `<3` represents a heart and expresses love or affection.

NRC Hashtag Sentiment Lexicon [28]. It is a sentiment lexicon of Twitter's hashtags. This is also a dictionary of words with associations to positive and negative sentiments. It consists of $54,129$ unigrams (single words), $316,531$ bigrams (two-word sequence) and $308,808$ of unigram–unigram pair, unigram–bigram pair, bigram–unigram pair, or a bigram–bigram pair. In this paper, we used the NRC Hashtag Sentiment Lexicon version 1.0[1]. We have used a Java/Android implementation to read the lexicon dictionaries of unigrams, bigrams, and pairs.

OpinionFinder (MPQA) [47]. OpinionFinder is a system that performs subjectivity analysis. It has two step process to identify polarity in a sentence. First, it classifies whether the sentence is polar (i.e. positive or negative) or neutral. Second, it takes all phrases and disambiguates their contextual polarity. It uses a prior-polarity lexicon (*) to train a dataset. We used a public version provided by authors. Original code available in Java was customized to run in our project architecture and then adapted to run on Android, specially the intermediate steps generating files. For Part

[1] `http://saifmohammad.com/WebPages/lexicons.html`

of Speech, as it is done with NLTK Library in Python, we have used the Stanford-CoreNLP Library, version 3.4.1.

OpinionLexicon [21]. It is also known as Sentiment Lexicon. OpinionLexicon[2] is a lexical method that measures the polarity of a sentence. It consists of two lists one with $2,006$ positive words and other $4,783$ negative words. The dictionary was built using data mining techniques in consumers reviews about products sold online and then labeling it as positive or negative. We have implemented the lexicon dictionary in Java/Android.

Panas-t [20]. It is a psychometric scale proposed to detect mood fluctuations of users on Twitter. The method consists of an adapted version of the Positive Affect Negative Affect Scale (PANAS) [46], which is a well-known method in psychology. The method is designed to track any increase or decrease in sentiments over time. PANAS-t assumes joviality, assurance, serenity, and surprise to be positive affect, fear, sadness, guilt, hostility, shyness, and fatigue to be negative affect, and attentiveness to be neutral. We have implemented the lexicon dictionary usage in Java and then adapted the code to run it on Android.

SASA [45]. It is also a machine learning-based method called the SailAil Sentiment Analyzer. SASA was originally proposed to be a real-time method that detects public sentiments on Twitter during the 2012 U.S. presidential election. SASA classifies messages in a range of $[-1, 1]$, with -1 and 1 being the most negative and most positive score, respectively. We have translated the original code, available in Python, into Java and then adapted it to run on Android.

SenticNet [10]. Is a method of opinion mining and sentiment analysis that explores artificial intelligence and semantic Web techniques. The goal of SenticNet is to infer the polarity of common sense concepts from natural language text at a semantic level, rather than at the syntactic level. The method uses Natural Language Processing (NLP) techniques to create a polarity for nearly $14,000$ concepts. In this work, we will consider scores less than zero as negative, equals to zero as neutral, and greater than zero as positive. We implement an approach to read the lexicon dictionary in Java and then adapted the code to run it on Android.

Sentiment140 Lexicon [30]. This is also a dictionary of words with associations to positive and negative sentiments. The dictionary of Sentiment140 Lexicon consists of up to $66,000$ unigrams (single words), $677,000$ bigrams (two-word sequence), and $480,000$ of unigram–unigram pair, unigram–bigram pair, bigram–unigram pair, or a bigram–bigram pair. These combinations were extracted from tweets from Stanford Twitter Corpus [18]. We used the Sentiment140 Lexicon version 1.0[3]. We have used a Java/Android implementation to read its lexicon dictionaries of unigrams, bigrams, and pairs.

SentiStrength [43]. It is a well-known method that builds a lexicon dictionary annotated by humans and improved with the use of Machine Learning. The core classification of this work relies on the set of words in the LIWC dictionary [42], and the authors expanded this baseline by adding new features for the social networks context. SentiStrength classifies messages in a range of $[-5, 5]$, with -5 and 5 being

[2] `http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html`

[3] `http://saifmohammad.com/WebPages/lexicons.html`

the most negative and most positive score, respectively. Messages classified between -1 and 1 are considered as neutral. We used SentiStrength version 2.0, which is available at `http://sentistrength.wlv.ac.uk/Download`. We also have used original .jar available and implemented a Java/Android code to access its results.

SentiWordNet [16]. It is a tool that is widely used in opinion mining and is based on an English lexical dictionary called WordNet [27]. This lexical dictionary groups adjectives, nouns, verbs and other grammatical classes into synonym sets called synsets. SentiWordNet associates three scores with synset from the WordNet dictionary to indicate the sentiment of the text: positive, negative, and neutral. We implemented a Java/Android code to read its dictionary.

Stanford Recursive Deep Model [41]. The method uses a deep learning approach. We have used the original .jar available and implemented a Java/Android code to access polarity prediction resource that deals with its output.

Umigon [24]. Belongs to the family of lexicon based sentiment classifiers. It is specifically designed to detect sentiment in tweets. It is composed by 4 parts: Semantic Detection (smiles and onomatopes), evaluation of hashtags, decomposition of tweets in up to 4 n-gram. The execution of this method label the messages as negative, positive, or neutral. Original code available in Java was customized to run in our project architecture and then adapted to run on Android.

USent [34]. It is a method that implements the nearest neighbor technique from Machine Learning field. The execution of this method labels the messages as negative, positive, or neutral. We have translated the original code, available in Python, into Java and then adapted it to run on Android. Thus, the original Part of Speech code is done with NLTK Library in Python and we have used the StanfordCoreNLP Java Library, version 3.4.1.

Vader [22]. The Valence Aware Dictionary for Sentiment Reasoning is a human-validated sentiment analysis method developed for Twitter and social media contexts. We also have translated the original code, available in Python, into Java and then adapted it to run on Android.

4 Results

This section describes the experimental setup we have used to run the experiments. First, we describe our experimental environment and setup and then we present the results of our performance evaluation in 3 scenarios: (i) battery evaluation; (ii) memory evaluation; and (iii) CPU evaluation. We should emphasize that our work focus on off-the-shelf methods as they are used in practice.

4.1 Experimental Setup

In this section, we describe the experimental setup used to run our experiments. Our first step consists of developing an API that offers a call to each sentiment analysis method deployed in the Android environment. Then, we developed two different Android applications, one that calls the API to run the sentiment analysis methods and another that monitors the first application's performance.

Table 1: Overview of the 16 sentiment analysis methods based on Lexicon or Machine Learning step as their core idea. It also shows the output of each method. Lexicon size means the number of terms each Lexicon method contains.

Name	Output	Lexicon size	Core Idea
AFINN [31]	Provides polarity score for lexicons (-5 to 5)	2,477	Lexicon
Emolex [29]	Provides polarities for lexicons	141,820	Lexicon
Emoticons [14]	-1, 1	79	Lexicon
NRC Hashtag [28]	Provides polarities for lexicons	679,468	Lexicon
OpinionFinder (MPQA) [47]	negative, objective, positive	20,611	Lexicon & Machine Learning
OpinionLexicon [21]	Provides polarities for lexicons	6,787	Lexicon
Panas-t [20]	Provides association for each word with eleven moods (joviality, attentiveness, fear, etc.)	50	Lexicon
SASA [45]	negative, neutral, unsure, positive	-	Machine Learning
SenticNet [10]	negative, objective, positive	15,000	Lexicon
Sentiment140 Lexicon [30]	0, 2, 4	-	Lexicon
SentiStrength [43]	-1, 0, 1	2,698	Lexicon & Machine Learning
SentiWordNet [16]	Provides negative, objective, and positive scores for each word (0.0 to 1.0)	117,658	Lexicon & Machine Learning
Stanford Recur. Deep Model [41]	very negative, negative, neutral, positive, and very positive	227,009	Lexicon & Machine Learning
Umigon [24]	negative, neutral, positive	1,053	Lexicon
USent [34]	neg, neu, pos	MPQA (8,226) Their own (9,176)	Lexicon & Machine Learning
Vader [22]	[< -0.05), (-0.05, ..., 0.05), [0.05 >]	7,517	Lexicon

To measure the performance metrics, we adapted the Android resources Monitor, OSMonitor [1] as it is able to collect information about CPU, memory, and battery usage of another application. It is also important to note that we have only collected memory and CPU usage of our API application to measure its performance. Therefore, all the resources used by the OSMonitor during the experiment are not accounted in our results.

The communication between the OSMonitor and our application is done by Android broadcast messages provided by Android SDK. Both applications have a service running in the background that is responsible for sending and receiving messages, running the methods and collecting the hardware information, which was stored in a file.

As the focus of this set of analysis is to evaluate methods' execution performance and their prediction accuracy, we used a single dataset of unlabeled Twitter messages and we test each method for different sizes of input messages. Particularly, Each experiment consists of running each method with an input of 10, 100, $1k$, and $10k$ tweets in English, extracted from a known Twitter dataset [12]. After executing an experiment, we clean the memory and restart the device to make sure that all experiments are running on the same conditions. We also run each method 31 times for each input size in order to present average results with their respective expected errors (i.e. confidence intervals).

Our evaluation consists in two specific moments, during load (initialization of the method) and during execution (moment when the method start to process the input). For the first moment, we measured the time and amount of memory necessary to load the methods on memory. This is useful, for example, to distinguish the impact that loading a large dictionary on memory can have on the load time of the application and also measure the amount of memory it requires. For the second moment, in addition to the necessary time and memory taken to get a response from the sentiment analysis method after its loading process, we also considered the battery usage.

During the evaluation, we had to be aware of how much some resources are scarce such as processing, memory, and Internet bandwidth. Any adapted method for mobile platforms must take into account these features.

Another point of view is the user perspective. Once we have our API running in the background to measure the user's text. In this scenario, the load time would impact during the booting process once the API should be loaded during the boot to be ready for use. Then the user would have to wait a long time if the developers choose a method that has a long load time. Furthermore, when we consider the execution time the user should not wait so long to have an analysis of her text, otherwise it would impact in her application usability.

We have used 5 LG G3 smartphone devices to run our experiments. These devices have a Quad-Core 2.5 GHz Qualcomm Snapdragon 801 processor, $2GB$ of principal memory (RAM), $3,000$ mAh battery, running Android 5.0 (Lollipop).

4.2 Performance Evaluation

This section describes a performance evaluation of popular sentiment analysis methods running on Android devices.

We present the performance evaluation in 3 scenarios: (i) battery evaluation; (ii) memory evaluation; and (iii) CPU evaluation. This is necessary for understanding to what extent is better to run those methods on mobile devices instead of in server side. Our evaluation consists in two specific moments, during load (initialization of the method) and during execution (moment when the method start to process the input).

The expressive use of CPU may decrease the battery level or even increase the mobile temperature that means some user discomfort. On the other hand, we need to measure the memory usage for each method once we have lexicon methods that demand more memory to load their dictionaries. Finally, we also need to measure the execution time because the user usually does not want to wait so long for a response.

The methods OpinionFinder and Stanford Recursive Deep Model, for the instance of $10k$ tweets dataset, were executed until the battery runs out. Therefore, their performance for the dataset of $10k$ tweets was not analyzed.

The battery evaluation is useful once the mobile devices have a limitation on battery life. Then, in this scenario, the best method for mobile devices would be which consumes less battery. The Figure 1 shows the battery consumption for each sentiment method consisting of average values of 31 executions. They are presented with confidence intervals of 95% confidence level. In this analysis, the methods USent and Stanford did not get a good performance. USent spent almost 6% of battery while Stanford spent almost 11.5% for the instances $10k$ and $1k$, respectively.

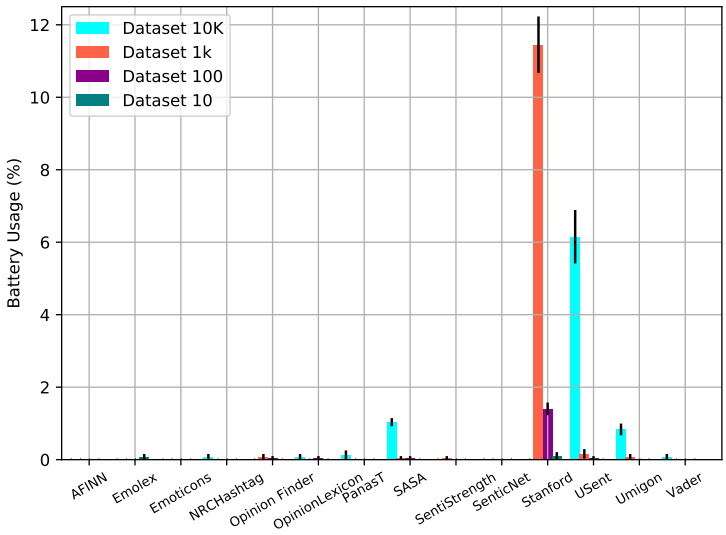


Fig. 1: Performance of the Battery Evaluation of 14 sentiment Analysis Methods. Results are average values of 31 executions with confidence intervals of 95% confidence level.

The next scenario is the memory evaluation. The mobile devices have a limited memory RAM when compared to a desktop. Then, evaluate the memory consumption become an important step for running application on these devices. Some of the sentiment analysis methods requires a huge amount of memory RAM, as we can see in the Figure 2 (a) the methods NRCHashtag, USent, and Stanford Recursive Deep Model use almost $100MB$ of memory RAM while OpinionFinder uses almost

$220MB$ during the load process. Further, for the execution process, see Figure 2 (b), the methods SASA ($384MB$), NRCHashtag ($207MB$), OpinionFinder ($276MB$), USent ($245MB$), and Stanford Recursive Deep Model ($193MB$) also use a lot of memory RAM.

The last scenario, CPU evaluation, is important to evaluate the CPU consumption of each method. This scenario is correlated to the first scenario (Battery consumption) because the battery life decreases due to a long CPU consumption. The Figure 3 (a) shows the CPU usage during the load time. We can note that the method OpinionFinder needs almost 64 seconds to load the method. Also, USent needs almost 10 seconds. The Figure 3 (b) shows the time, also in seconds, necessary to execute the sentiment analysis methods for all the instances (10, 100, $1k$, and $10k$). During the execution, we can note that Stanford Recursive Deep Model consumes almost $1,255$ seconds to run the $1k$ instance. The USent run a $10k$ in around 713 seconds.

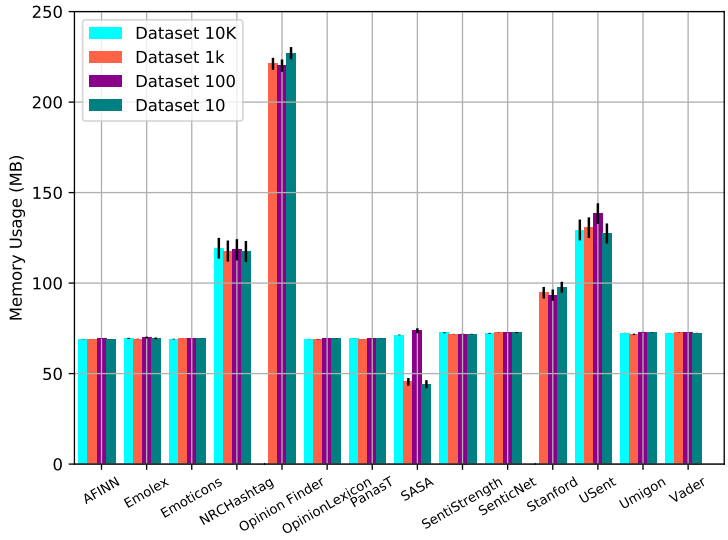
Our results show that some evaluated methods consume many mobile resources as battery, memory, and also CPU. Therefore, methods such as NRCHashtag, OpinionFinder, SASA, Stanford Recursive Deep Model, and USent are not recommended for mobile devices. These methods use machine learning techniques, only exception for NRCHashtag that is a lexical method. In general, the machine learning methods got the worst results. The lexical methods had a good performance once they are based on dictionaries to compute the sentiment polarity scores. The methods Sentiment140 Lexicon and SentiWordNet were not able to run on Android.

Considering all methods in this study that were able to finish the experiment, we showed that it might be hard to use NRCHashtag, OpinionLexicon, SASA, Stanford Recursive Deep Model, and USent on current mobile devices. Another observation is that the lexical methods (e.g. AFINN, Emoticons, Emolex) obtained good results in terms of memory, CPU, and battery usage.

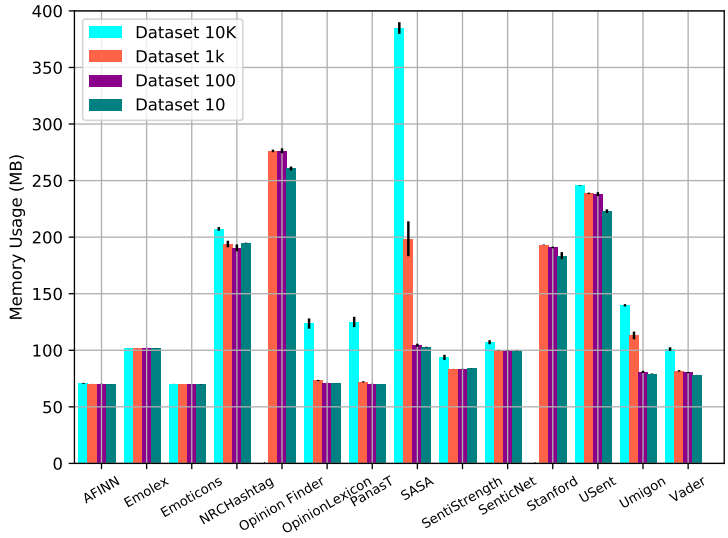
5 Concluding Discussion and Future Work

The real-time analysis of user sentiment inside mobile devices exposes new possibilities for applications, especially those that aim at measuring mood and opinions expressed by users in text data. It also allows them to measure the sentiment of user's instant messages, developed on the top of local mobile data that users may consider sensitive to be shared on social platforms as Facebook or Twitter. This work presents the first of a kind analysis about sentiment analysis in mobile devices, providing a better understanding of the performance of existent and popular sentence-level sentiment analysis methods. We implemented 14 sentence-level sentiment analysis methods that worked on Android and developed two applications, one to run the methods and the other to monitor their performance. With this strategy, we were able to evaluate these methods according to different metrics, such as load time and load memory, execution time, execution memory, and battery usage.

Our main findings show that it might be hard to use NRCHashtag, OpinionLexicon, SASA, Stanford, and USent on current mobile devices. Another observation is that the lexical methods (e.g. AFINN, Emoticons, Emolex) had a good performance in terms of memory, CPU, and battery usage once they are based on dictionaries to

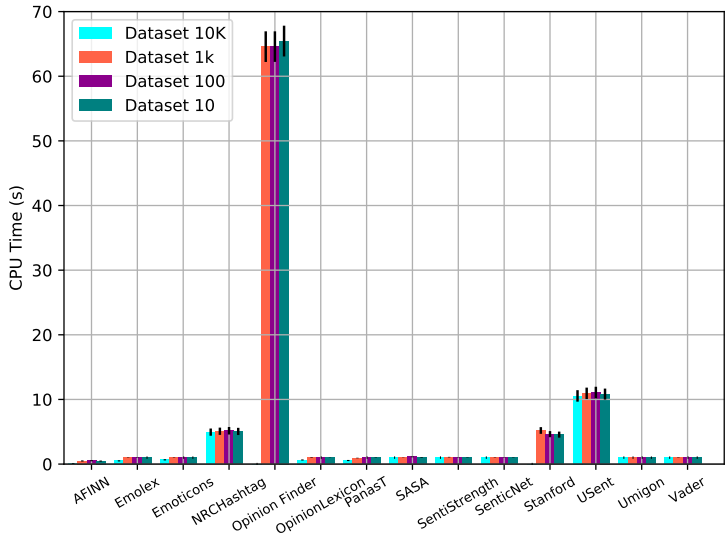


(a) Memory Consumption on Load

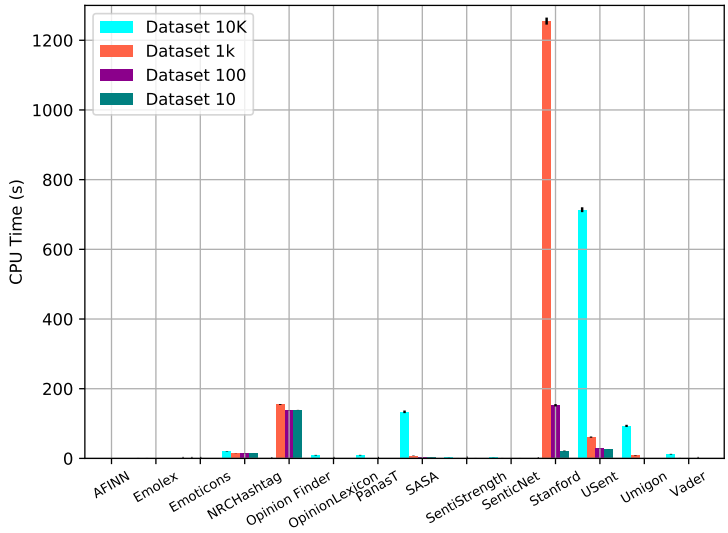


(b) Memory Consumption on Execution

Fig. 2: Consumption of the Memory Evaluation during the load and execution of 14 sentiment Analysis Methods. Results are average values of 31 executions with confidence intervals of 95% confidence level.



(a) CPU Usage on Load



(b) CPU Usage on Execution

Fig. 3: CPU Usage during the load and execution of 14 sentiment Analysis Methods. Results are average values of 31 executions with confidence intervals of 95% confidence level.

compute the sentiment polarity scores. In general, the machine learning methods got the worst results. The methods Sentiment140 Lexicon and SentiWordNet were not able to run on Android.

We hope these observations may guide developers towards the use of sentiment analysis in mobile applications. More important, we also hope our effort can help developers and researchers interested in exploring sentiment analysis as part of a mobile application, instead of creating applications that depend on server-side APIs.

As a final contribution, we note that deploying existing efforts is not an easy task. Thus, we plan to release the Android API that implements all the 14 sentiment analysis methods as part of a set of resources related to the iFeel system [5]. More specifically, our codes will be shared on a request-base at `http://www.ifeel.dcc.ufmg.br/`.

As future work, we expect to run experiments in different devices and also consider different datasets (e.g. a dataset with longer sentences). We also plan to explore multilingual approaches that are feasible to be done in mobile environments, such as machine translation [4]. More important, we envision the deploy a set of applications that use sentiment analysis [35, 39] to run in mobile environments in a native way.

Acknowledgements This project was supported by grants from Humboldt Foundation, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and Fundação de Amparo à Pesquisa do Estado de Minas Gerais (Fapemig).

References

1. Osmonitor for android. `www.osmonitor.mobi`. Accessed September 19, 2015
2. Facebook statistics. `https://zephoria.com/top-15-valuable-facebook-statistics` (2015). URL `https://zephoria.com/top-15-valuable-facebook-statistics/`
3. Youtube statistics. `https://www.youtube.com/yt/press/statistics.html` (2015). URL `https://www.youtube.com/yt/press/statistics.html`
4. Araújo, M., Reis, J., Pereira, A., Benevenuto, F.: An evaluation of machine translation for multilingual sentence-level sentiment analysis. In: Proceedings of the ACM Symposium on Applied Computing (SAC). ACM (2016)
5. Araújo, M., Diniz, J.P., Bastos, L., Soares, E., Júnior, M., Ferreira, M., Ribeiro, F., Benevenuto, F.: ifeel 2.0: A multilingual benchmarking system for sentence-level sentiment analysis. In: Proceedings of the International AAAI Conference on Web-Blogs and Social Media. Cologne, Germany (2016)
6. Bollen, J., Mao, H., Zeng, X.J.: Twitter Mood Predicts the Stock Market. CoRR **abs/1010.3003** (2010)
7. Bonnington, C.: In less than two years, a smartphone could be your only computer. wired (2015). URL `http://www.wired.com/2015/02/smartphone-only-computer/`

8. Bradley, M.M., Lang, P.J.: Affective norms for English words (ANEW): Stimuli, instruction manual, and affective ratings. Tech. rep., Center for Research in Psychophysiology, University of Florida (1999)
9. Burke, M., Marlow, C., Lento, T.: Social network activity and social well-being. In: Proceedings of the SIGCHI conference on human factors in computing systems (2010)
10. Cambria, E., Speer, R., Havasi, C., Hussain, A.: Senticnet: A publicly available semantic resource for opinion mining. In: AAAI Fall Symposium Series (2010)
11. Canuto, S., Gonçalves, M.A., Benevenuto, F.: Exploiting new sentiment-based meta-level features for effective sentiment analysis. In: Proceedings of the Nineth ACM International Conference on Web Search and Data Mining. ACM (2016)
12. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, K.P.: Measuring user influence in twitter: The million follower fallacy. In: International AAAI Conference on Weblogs and Social Media (ICWSM) (2010)
13. Chambers, L., Tromp, E., Pechenizkiy, M., Gaber, M.M.: Mobile sentiment analysis. In Proceedings of KES pp. 470–479 (2012)
14. Cool-Smileys: List of text emoticons: The ultimate resource (2010). `www.cool-smileys.com/text-emoticons`
15. De Choudhury, M., Counts, S., Horvitz, E.J., Hoff, A.: Characterizing and predicting postpartum depression from shared facebook data. In: Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing (2014)
16. Esuli, Sebastiani: Sentiwordnet: A publicly available lexical resource for opinion mining. In: Proc. LREC (2006)
17. Feldman, R.: Techniques and applications for sentiment analysis. Commun. ACM **56**(4), 82–89 (2013)
18. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. Processing **-**, 1–6 (2009)
19. Gonçalves, P., Araújo, M., Benevenuto, F., Cha, M.: Comparing and combining sentiment analysis methods. In: Proceedings of the 1st ACM Conference on Online Social Networks (COSN) (2013)
20. Gonçalves, P., Benevenuto, F., Cha, M.: PANAS-t: A Psychometric Scale for Measuring Sentiments on Twitter **abs/1308.1857v1** (2013)
21. Hu, M., Liu, B.: Mining and summarizing customer reviews. KDD, pp. 168–177 (2004)
22. Hutto, C., Gilbert, E.: Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: Eighth International AAAI Conference on Weblogs and Social Media (2014)
23. Hutto, C.J., Gilbert, E.: Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: ICWSM (2014)
24. Levallois, C.: Umigon: sentiment analysis for tweets based on lexicons and heuristics. In: Proceedings of the International Workshop on Semantic Evaluation, SemEval, vol. 13 (2013)
25. LiKamWa, R., Liu, Y., Lane, N.D., Zhong, L.: Can your smartphone infer your mood? ACM Workshop on Sensing Applications on Mobile Phones (PhoneSense) (2011)

26. Messias, J., Diniz, J.P., Soares, E., Ferreira, M., Araujo, M., Bastos, L., Miranda, M., Benevenuto, F.: Towards sentiment analysis for mobile devices. In: Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM (2016)
27. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
28. Mohammad, S.: #emotional tweets. In: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval). Association for Computational Linguistics (2012)
29. Mohammad, S., Turney, P.D.: Crowdsourcing a word-emotion association lexicon. Computational Intelligence **29**(3), 436–465 (2013)
30. Mohammad, S.M., Kiritchenko, S., Zhu, X.: Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In: Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval) (2013)
31. Nielsen, F.Å.: A new anew: Evaluation of a word list for sentiment analysis in microblogs. arXiv preprint arXiv:1103.2903 (2011)
32. Ogneva, M.: How companies can use sentiment analysis to improve their business. Mashable (2010)
33. Oliveira, N., Cortez, P., Areal, N.: On the predictability of stock market behavior using stocktwits sentiment and posting volume. In: L. Correia, L.P. Reis, J. Cascalho (eds.) EPIA, *Lecture Notes in Computer Science*, vol. 8154, pp. 355–365. Springer (2013)
34. Pappas, N., Popescu-Belis, A.: Sentiment analysis of user comments for one-class collaborative filtering over ted talks. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pp. 773–776. ACM (2013)
35. Pereira, M., Pádua, F., Pereira, A., Benevenuto, F., Dalip, D.: Fusing audio, textual and visual features for sentiment analysis of news videos. In: Proceedings of the International AAAI Conference on Web-Blogs and Social Media. Cologne, Germany (2016)
36. Plutchik, R.: A general psychoevolutionary theory of emotion, pp. 3–33. Academic press, New York (1980)
37. Pollyanna Gonçalves Daniel Hasan Dalip, H.C.M.A.G., Benevenuto, F.: On the combination of "off-the-shelf" sentiment analysis methods. In: Proceedings of the ACM Symposium on Applied Computing (SAC). ACM (2016)
38. Reis, J., Benevenuto, F., Vaz de Melo, P., Prates, R., Kwak, H., An, J.: Breaking the news: First impressions matter on online news. In: Proceedings of the 9th International AAAI Conference on Web-Blogs and Social Media (2015)
39. Reis, J., Goncalves, P., Vaz de Melo, P., Prates, R., Benevenuto, F.: Magnet news: You choose the polarity of what you read. In: International AAAI Conference on Web-Blogs and Social Media (2014)
40. Ribeiro, F.N., Araújo, M., Gonçalves, P., Gonçalves, M.A., Benevenuto, F.: Sentibench - a benchmark comparison of state-of-the-practice sentiment analysis methods. EPJ Data Science (2016)
41. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment tree-

bank. In: 2013 Conference on Empirical Methods in Natural Language Processing (2013)
42. Tausczik, Y.R., Pennebaker, J.W.: The psychological meaning of words: Liwc and computerized text analysis methods. J. of Lang. and Soc. Psych. **29** (2010)
43. Thelwall, M.: Heart and soul: Sentiment strength detection in the social web with sentistrength (2013). http://sentistrength.wlv.ac.uk/documentation/SentiStrengthChapter.pdf
44. Tumasjan, A., Sprenger, T.O., Sandner, P.G., Welpe, I.M.: Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. In: International AAAI Conference on Weblogs and Social Media (ICWSM) (2010)
45. Wang, H., Can, D., Kazemzadeh, A., Bar, F., Narayanan, S.: A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle. In: ACL System Demonstrations (2012)
46. Watson, D., Clark, L.: Development and validation of brief measures of positive and negative affect: the panas scales. Journal of Personality and Social Psychology **54**(1), 1063–1070 (1985)
47. Wilson, T., Hoffmann, P., Somasundaran, S., Kessler, J., Wiebe, J., Choi, Y., Cardie, C., Riloff, E., Patwardhan, S.: Opinionfinder: a system for subjectivity analysis. In: HLT/EMNLP on Interactive Demonstrations, pp. 34–35 (2005)