Mobile Sentiment Analysis

Lorraine Chambers[1], Erik Tromp[2], Mykola Pechenizkiy[2], Mohamed Medhat Gaber[1]
[1]School of Computing
University of Portsmouth
Portsmouth, Hampshire, PO1 3HE, England, UK
{lorraine.chambers@myport.ac.uk, mohamed.gaber@port.ac.uk}
[2]Department of Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
{e.t.tromp@gmail.com,m.pechenizkiy@tue.nl}

Abstract. Mobile devices play a significant part in a user's communication methods and much data that they read and write is received and sent via mobile phones, for instance SMS messages, e-mails, Twitter tweets and social media networking feeds. One of the main goals is to make people aware of how much negative and positive content they read and write via their mobile phones. Existing sentiment analysis applications perform sentiment analysis on downloaded data from mobile phones or use an application installed on another computer to perform the analysis. The sentiment analysis described in this paper is to be performed locally on the mobile phone enabling immediate and private analysis of personal messages and social media contents, allowing the users to be able to reason about their mood and stress level that may be affected by what they had been receiving. Experimental results showed the effectiveness of the proposed system on Android smartphones with varying computational capabilities.

Keywords. Sentiment analysis, data mining, and mobile computing

1 Introduction

Sentiment analysis is a branch of natural language processing and one stage in the process of opinion mining. To achieve opinion mining there are five tasks that need to be carried out, these are: Entity extraction and grouping, aspect orientation and grouping, opinion holder and time extraction, aspect sentiment classification and opinion quintuple generation for summarization of the opinions. The stage of aspect sentiment classification attempts to classify the sentiment of a particular aspect of a sentence; in the context of this paper, the general sentiment of the text will be considered and not the sentiment of a particular aspect of the sentence, as the aim is to provide the user with a general view of the sentiment of the texts within the mobile device and not the sentiments of particular products or services.

SMS messages are a means of sending short text messages not longer than 160 characters for the Latin alphabet. Although nowadays most mobile phones have the capability to split longer texts into multiple messages and the recipient's phone to receive this as one message, an SMS message is typically one or two sentences long.

This is similar to tweets on Twitter which allows text-based posts of up to 140 characters and has been described as "the SMS of the internet" [14]; because of these similarities, SMS messages will be considered to be the same as tweets.

Nowadays most personal communication is recorded digitally in mediums such as SMS text messages, e-mails, tweets, Facebook updates and other social media networking sites. All of these are accessible from mobile phones and as these become faster and more powerful, the possibility of performing sentiment analysis on mobile devices has become more attractive, enabling the user to have sentiment analysis performed locally, rather than having personal information sent to a server or downloaded to another application on a computer to be analyzed. The overall goal is to perform sentiment analysis locally on the user's mobile phones to enable the user to reflect upon how much general positive or negative content they are reading or writing.

We have developed a mobile application for the Android operating system that performs sentiment analysis locally on the mobile phone for SMS messages. Due to the similarity of Twitter messages to SMS messages, the work reported in this paper generally target both Twitter tweets and SMS messages. It is based on the SentiCorr [2] system which performs multi-lingual sentiment analysis of personal correspondence on e-mails, Twitter tweets, Facebook and other social networking media. The multilingual aspect is beyond the scope of this paper, and the only language that shall be considered for SMS messages in this paper is English. It is worth noting that the system can be easily extended to handle all text received on the mobile phone. Our system reported in this paper has extended the SentiCorr system in the following aspects: (1) applying a number of POS taggers and experimentally assessing their computational performance and accuracy; (2) experimental study of the performance with respect to the system configuration of the mobile device; and (3) a temporal aspect of the sentiment has also been developed allowing the users to reason about the effect on the sentiment on their level of stress.

In the following section work related to the different methods of sentiment analysis and the applications of sentiment analysis to mobile applications are considered. Section 3 describes our approach to mobile sentiment analysis, while section 4 summarizes our experimental study. Finally, section 5 concludes the findings and suggests further work.

2 Related Work

There has been much work done on sentiment analysis, especially in the last five years due to the abundance of data being available due to the explosion of social media sites and blogging sites like Twitter. This is reflected in the differences in Liu's opinion mining and sentiment analysis chapters in subsequent versions [6] published in 2006 and [7] published in 2011. There has also been a comprehensive survey of sentiment analysis techniques as summarized by Pang [1]. In this paper sentiment analysis shall be considered at the sentence level where there are two tasks to be performed; subjectivity classification to determine whether it is a subjective sentence or

an objective sentence and sentiment classification – if the sentence is subjective, determine if it contains a positive or negative opinion. The assumption at this level is that the sentence expresses a single opinion from a single opinion holder. Both of these tasks are classification problems, so supervised learning methods are applicable. Early methods used the Naïve Bayes classifier [9], subsequent methods have applied regression models [11] and support vector regression and boosting [12]. Semi-supervised [10] and unsupervised techniques have also been applied [13].

For mobile devices, sentiment analysis has been utilized for reflection of personal informatics; analyzing SMS messages and emails to provide a general overview of a person's life at a particular point, but this is not performed on the mobile device but by another application to which the data is downloaded [3], and sentiment analysis has been employed with SMS messages to gather feedback about teaching, where SMS messages are sent by students at the end of a lecture where they are automatically analyzed at the server-end to provide information about the teaching [8]. There are mobile phone applications available that involve sentiment analysis; for instance, for the iPhone there is an application available that tells you what people think about your area by analyzing the sentiment of tweets around your location by sending them to an analytics engine for sentiment analysis [4]. There is also an Android mobile application that analyses tweets and tries to determine if the attitude of the writer is positive or negative; it uses the internet and returns the results that the twitter sentiment analysis engine has generated [5]. To our knowledge there are no mobile applications that locally execute sentiment analysis on the mobile phone.

3 Mobile Sentiment Analysis System Description

The target platform selected for the mobile sentiment analysis application is Android due to its "open" nature, availability of development resources and its wide usage. The sentiment analysis is to be performed using the principles of the SentiCorr sentiment analysis engine; firstly it will be described how SentiCorr achieves sentiment analysis and then the different aspects of enabling it to work on the Android platform will be further discussed.

SentiCorr achieves sentiment classification at the sentence level by using POS (Position Of Speech) tagging to identify the types of the words in the sentence; the subjectivity detection stage then uses the POS tags to identify opinion lexicon and hence if the sentence is subjective or objective; the polarity detection stage also utilizes the POS tags to search for patterns in the sentence that indicate positive or negative expressions.

For the POS tagging stage, Stuttgart University's Tree Tagger was employed; AdaBoost for subjectivity detection and an in-house method called Rule-Based Emission Model (RBEM) was developed for the polarity detection stage. It is not the aim of this paper to propose new solutions for the sentiment analysis as extensive experiments have been conducted in comparing these classification techniques with others such as Majority class, Prior Polarity, Naïve Bayes and Support Vector Machines, in which AdaBoost and RBEM outperform these other methods [15]. The following

paragraphs summarize the algorithms used for subjectivity detection and polarity detection, a comprehensive description can be found in [15].

The principle employed for subjectivity detection is boosting, by use of the AdaBoost (adaptive boosting algorithm) [19] and is a general method for generating a strong classifier from a set of weak classifiers. Each time an instance is incorrectly classified, it is given a greater weight for use in the next round of classification. This process continues until the maximum number of rounds is reached or the weighted error is more than 50%. The weak learners used are decision stumps of the form *if f present then label = a else label = b* where *f* is a feature and *a* and *b* are labels. Features utilized are POS tags, pre-defined lexicons that contain positive, negative and negation words, the presence of exactly one positive word, the presence of multiple positive words, the presence of exactly one negative word and the presence of multiple negative words, and whenever a positive or negative word is directly preceded by a word from the negation list, its polarity is flipped.

The principle employed for polarity detection is RBEM which uses rules to define an emissive model. The rules emerge from eight different pattern groups which are positive patterns, negative patterns, amplifier patterns, attenuator patterns, right flip patterns, left flip patterns, continuator patterns and stop patterns. These patterns are combined with rules to define an emissive model. A model is constructed by representing patterns as lists of words and corresponding POS tags. In the patterns, word wildcards are allowed which means that wildcards for a word can appear at any position of a pattern. Single-position wildcards are allowed so that a single entity in the pattern can be any word and any POS tag. Multi-position wildcards are also allowed so that any number of word tag pairs can occur in-between two elements that are not multi-position or single position wildcards. The model consists of a set of patterns per pattern group, each pattern except for the positive and negative patterns adhere to an action radius, which is set to 4 in this case.

When classifying previously unseen data, all of the patterns that match the sentence are collected from the model, and a rule associated with each pattern group is applied to each pattern in the message. All patterns of all groups are evaluated for a match within the sentence; if there is a match, the start position and the end position of the pattern in the sentence is recorded. Some patterns may occur within other patterns in the sentence, if so these subsumed patterns are removed from the final pattern collection. Once the patterns that occur in the sentence have been collected, the rules for each pattern group are applied. The rules must be applied in the correct order as outlined in the following paragraph.

The first rule to be applied is Setting Stops; this sets a stop at the starting position of all the left flip and stop patterns. The second rule to be applied is Removing Stops; if there is a stop to the left of a continuator pattern within the pre-set action radius it is removed. The third rule to be applied is Positive Sentiment Emission; for each positive pattern an emission value is calculated based on the distance of the elements in the sentence from the centre of the positive pattern, which decays the further the element is from the centre of the pattern, e^{-x} is used as the decaying function, this is calculated for each element until stops are reached.

The fourth rule to be applied is Negative Sentiment Emission; this is handled the same way as Positive Sentiment Emission except that the decay function is $-e^{-x}$. The fifth rule to be applied is Amplifying Sentiment; amplifier patterns amplify sentiment emitted by positive or negative patterns and similarly to the positive and negative patterns, amplification reduces over distance. The function used is $1+ e^{-x}$ where x is the distance within the action radius. The sixth rule to be applied is Attenuating Sentiment; this performs the reverse of Amplifying Sentiment and the decay function applied is $1 - e^{-x}$.

The seventh rule to be applied is Right Flipping Sentiment; if there is a right flip pattern the emission of sentiment is flipped to the right and if there is a stop at the exact centre of the right flip pattern, it is ignored. The eighth rule to be applied is Left Flipping Sentiment; this mirrors the effect of the right flip pattern.

Once the rules have been applied, every element of the sentence has an emission value and the final polarity of the message is calculated by summing the emission values for each element. If the final polarity of the sentence is greater than zero, the sentence is positive; if it is less than zero the sentence is negative, if it is zero the polarity of the sentence is unknown due to insufficient patterns in the sentence model.

The main aim of mobile sentiment analysis is to perform sentiment analysis on SMS messages which were earlier likened to tweets and that language identification was beyond the scope of this paper, this reduces the original SentiCorr framework to that of the POS tagger, Subjectivity Detection and Polarity Detection. Although we have focused in this paper on short text sentiment analysis as applied to SMS messages and tweets received onboard the mobile phone, the work could be easily generalized to other social media items like Facebook and LinkedIn.

Android mobile phones use ARM processors (Advanced RISC Machine) and the original POS tagger (TreeTagger) could not be used on the Android operating system as the source code was not available for re-compilation suitable for an ARM processor. This reduced the POS taggers available as they are constrained to POS taggers written in Java for which either a library where all the components and dependencies are capable of executing on an ARM compiler or the source code is available so it can be compiled to execute on an ARM processor. The shortlist of POS taggers tested were Stanford POS tagger and OpenNLP POS tagger.

The subjectivity detection stage and the polarity detection stages required that the POS tags were in the format of the Penn Tree Bank set 1 for the software to operate correctly with the pre-trained models used within the subjectivity and polarity detection stages.

POS taggers are usually supervised and as such require a model. Loading these models contributed to how the software was architected. On a PC, the time to load these models is small compared to the time to load them on a mobile device. In SentiCorr, the language of the text is assessed at the sentence level and if the sentence is in a different language to the previous one, a different model needs to be loaded for the POS tagging, subjectivity and polarity classification stages. As we are not yet considering the multi-lingual aspect, the model is loaded once at startup of the application and the same language is assumed throughout each usage of the application and

the model is loaded only once per application usage. These constraints shaped the workflow of the mobile sentiment analysis system as shown in Figure 1.

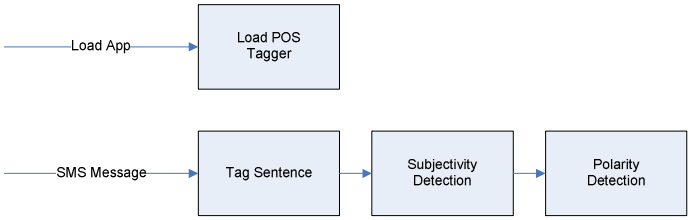

Fig. 1. Mobile Sentiment Analysis Workflow

In the mobile sentiment analysis solution the POS taggers are dynamically interchangeable for analysis purposes and can be loaded via settings menus and the output form each POS tagger are transformed into a standard output so that the interface to the rest of the application remains constant regardless of the POS tagger. Tagging of the sentence splits the text into sentences and tags them using the selected POS tagger. The tagged sentences are then passed to the subjectivity detection stage where the algorithm operates in the same way as within SentiCorr as described in the previous paragraphs.

Once the subjectivity has been determined, if it is subjective, the sentence is then passed to the polarity detection stage where the algorithm operates in the same way as the SentiCorr algorithm also as summarized in the previous paragraphs; if the sentence is objective, no further processing is applied to it.

The sentiment analysis code is implemented as a standard Java library that the Android application uses, ensuring its use is not limited to an Android mobile application. The models for the subjectivity and polarity detection stages are stored in XML format and serializable directly into Java objects using Simple XML [20] so that it has the possibility to be extended to allow creation of new models that can be stored in the correct format for later use.

Having discussed the technical and implementation details of our mobile sentiment analysis system, the following section provides an experimental study of the system proving empirically its feasibility and efficiency.

4 Experimental Results

The mobile sentiment analysis application was installed on three mobile phones, the specifications of which are shown in Table 1. The aim of varying the mobile phones is to conduct stress testing, so that to reveal the minimum configuration of computational power that is able to run our system.

The load time of the POS tagger models was the major factor in the duration of the execution time of the application. The POS taggers evaluated were OpenNLP and Stanford POS Tagger. Each of these taggers has a number of models which are summarized in the following paragraphs; these models are also listed in Table 2 along

with the time taken to load these models for each phone. Each POS tagger model was loaded ten times and an average taken to give the model load times in Table 2.

The Open NLP POS tagger comes with two different types of models: maximum entropy model and a perceptron model, each with the option of using a dictionary. A token can have many tag possibilities depending on the token and the context. Open NLP uses a probability model to guess the correct POS tag out of the tag set, to limit the possible tags, a dictionary can be used. The maximum entropy model with a dictionary produced the best results at 87%. The perceptron model is a linear classifier and relies on Viterbi decoding of training examples [18].

The Stanford POS tagger comes with trained English tagger models and taggers trained on the Wall Street Journal corpus from the Penn Treebank Project., as depicted by 'wsj' in the model name; the 'left3' in the model name means that the model uses the left3 words architecture and includes word shape features; the 'distsim' part of the tagger name means that the model includes distributional similarity features. All of the Stanford POS tagger models utilize a maximum entropy method of tagging where a probability is assigned for every tag in a set of possible tags for a word where the possible tags are determined from the sequence of words preceding the word that is to be tagged [16]. The Stanford POS tagger model that performed the best on the tested data was the english-left3words-distsim model and the wsj-0-18-left3words model, showing that in general, the left3 words architecture was a successful model for the data. The bidirectional models use a cyclic dependency networks to achieve bi-directional traversing of the words in a given sentence [17].

As the sentiment analysis code is implemented as a standard java library, we were able to calculate the accuracy of the sentiment analysis for each model on a PC where the accuracy is taken as the number of correctly tagged tokens divided by the total number of tokens and represented as a percentage. The sentiment was analyzed on part of the original data used in the evaluation of SentiCorr and is based on 60 texts and utilizes the POS tags from the original data and uses this as the gauge for accuracy. These 60 texts were obtained from Twitter by scraping all public data and manually labeling 20 negative, 20 objective and 20 positive tweets, according to the tweet's text. The results are shown in table 2. The peak memory usage of the application during the loading of the POS tagger models was also recorded and is shown in Table 2.

Table 1. POS tagger model load times on mobile phone

Mobile Phone Model	**Android Version**	**SD Card Available Space (GB)**	**Internal Phone Storage (GB)**	**Total Available Memory (GB)**
GT540	2.1	1.77	0.099	1.869
HTC Desire HD	2.3.5	3.0	0.815	3.815
Galaxy Nexus	4.0.2	N/A	N/A	13.33

Table 2. POS tagger model size and accuracy

POS Tagger Model		Model Size (MB)	Accuracy (%)	Peak Memory Usage (MB)	Model Load Times (s)			MinSpec		
					GT540	HTC Desire HD	Galaxy Nexus	Min Android Ver	Storage Space (MB)	Heap Space (MB)
Open NLP	Maxent no dictionary	5.46	85	105	OSM	17.78	23.03	2.1	180	61
	Maxent with dictionary	5.56	87	-	Invalid format for dictionary file			-	-	-
	Perceptron no dictionary	3.78	83	105	OSM	17.78	22.21	2.1	178	62
	Perceptron with dictionary	3.88	85	-	Invalid format for dictionary file			-	-	-
Stanford	english-left3words-distsim	20.15	90	230	OSM	OHM	250.79	3.0	182	233
	english-caseless-left3words-distsim	19.77	88	225	OSM	OHM	180.26	3.0	182	229
	wsj-0-18-left3words-distsim	17.38	87	207	OSM	OHM	205.44	3.0	181	207
	wsj-0-18-left3words	7.98	90	110	OSM	OHM	50.71	3.0	177	110
	wsj-0-18-caseless-left3words-distsim	17.12	88	201	OSM	OHM	179.71	3.0	180	201
	wsj-0-18-bidirectional-distsim	31.65	87	285	OSM	OHM	OHM	3.0	-	-

OSM = Out of Storage Memory, **OHM** = Out of Heap Memory

The accuracy of the models ranges from 83% to 90% across the Stanford and the Open NLP POS tagger models, with the most accurate being the Stanford English left 3 words and the Stanford wsj-0-18left3words which also had a low load time but only ran on the Galaxy Nexus phone. In fact, no Stanford model could be loaded on a phone that had an Android version of lower than 3.0 because the Android application setting `largeHeap` was required to be set to true in the Android manifest file to allow more memory to be dynamically allocated to the application, hence no results for the Stanford POS tagger for the LG GT540 or the HTC Desire HD mobile phones could be recorded for the Stanford POS taggers. No results could be recorded for the GT540 phone as there was not enough storage memory. From this information minimum phone specifications for each tagger have been calculated and included in Table 2. Overall, the Stanford POS tagger model was more accurate but requires more mo-

bile phone resources such as memory and load time, which limits the targets it can be installed on. In most cases, it took nearly ten times as long to load the Stanford models as it did the Open NLP model, for a 2% increase in accuracy.

The duration of the loading of the POS models has been focused on but the AdaBoost and Emission Miner models also have an overhead when they are loaded. The AdaBoost and Emission Miner models use the POS tags that are produced by the POS taggers. The original POS tagger – TreeTagger output Penn tree bank I tag set whereas the Stanford POS tagger and Open NLP POS tagger both output Penn tree bank II tag set. The difference in these tag sets are that in tree bank II the individual tags for "VH" tags are now included in "VB" tags as re "VV" tags, meaning that the verbs "to have" and "to take" are now included under the general verb tags. This meant that the existing AdaBoost and Emission Miner models had to be adjusted to take this into account. The result is that if the POS tagger model is changed to one which uses a different tag set, then different AdaBoost and Emission Miner models are also required to be loaded. The time it takes to load these models will be critical if a dynamic multi-language approach is required as in the original SentiCorr system.

5 Conclusion and Future Work

Sentiment analysis can be performed locally on a mobile phone, as we have proved empirically in this paper. To the best of our knowledge, the proposed and implemented system reported in this paper is the first mobile sentiment analyzer. The amount of time and resources this takes largely depends upon the POS tagger model that is utilized. The Stanford POS tagger is more accurate, but takes longer to load and requires more memory. In a non-language dynamic environment, the model is only loaded once per start-up of the application but limits the user to the same language unless this is specifically changed by the user. The change of a POS tagger could affect the subsequent subjectivity and polarity detection stages, meaning that new models for these stages may also need to be loaded. If the application was to incorporate dynamic language selection then the load time of the models would become critical.

Future work is to include experimentation on a wider set of Android mobile phones with differing memory and Android operating systems, and inclusion of analysis on other POS taggers implemented in Java and those specifically aimed at Twitter data such as Tweet NLP. Experimentation could also be extended to the training of smaller models to widen the target mobile phones of the application such that the application could be extended to dynamically interrogate and change the language during sentiment classification.

References

1. Pang, B., Lillian, L.: Opinion Mining and Sentiment Analysis. In: Foundations and Trends in Information Retrieval 2, pp. 1-135 (2008)

2. Tromp, E., Pechenizkiy, M.: SentiCorr: Multilingual Sentiment Analysis of Personal Correspondence. In: IEEE 11th International Conference on Data Mining Workshops, pp. 1247-1250 (2011)
3. Hangal, S., Lam, M.: Sentiment Analysis on Personal Email Archives. Proceedings of the 24th annual ACM symposium on User interface software and technology (2011)
4. Chatterbox analytics, http://cbanalytics.co.uk/apps/sentimental
5. SmmMobile – Sentiment Analyser https://market.android.com/details?id=com.cyhex.smmMobile
6. Liu, B.: Chapter 11 Opinion Mining. In: Web Data Mining. pp. 411-448. Springer , Heidelberg (2006)
7. Liu, B.: Chapter 11 Opinion Mining. In: Web Data Mining. pp. 459-526. Springer , Heidelberg (2011)
8. Leong, C.K., Lee, Y.H.: Mining sentiments in SMS texts for teaching evaluation. In Expert Systems with Applications 39, pp. 2584-2589 (2012)
9. Yu, H.: Towards Answering Opinion Questions: Separating facts from Opinions and identifying the Polarity of Opinion Sentences. In: EMNLP '03 Proceedings of the 2003 conference on Empirical methods in natural language processing pp. 129-136 (2003)
10. Dasgupta, S.: Mine the easy, classify the hard: A semi-supervised approach to Automatic Sentiment Classification. In: ACL '09 Proceedings of the joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP 2 pp. 701-709 (2009)
11. Hatzivassiloglou, V.: Predicting the semantic orientation of Adjectives. In: ACL '98 Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics. Pp. 174-181 (1997)
12. Wilson, T., Wiebe, J., Hwa, R.: Just How Mad Are you? Finding Strong and Weak Opinion Clauses. In: Proceedings of the National Conference on Artificial Intelligence. 10 pp.761-769 (2004)
13. Turner, D.: Thunbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL). pp. 417-424 (2002)
14. Wikipedia – Twitter. http://en.wikipedia.org/wiki/Twitter
15. Tromp, E.: Multilingual Sentiment Analysis on Social Media. http://alexandria.tue.nl/extra1/afstversl/wsk-i/tromp2011.pdf
16. Toutanova, K., Manning, C.: Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70 (2000)
17. Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. In: Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of HLT-NAACL 2003, pp. 252-259 (2003)
18. Collins, M.: Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1-8 (2002)
19. Freund, Y., Shapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Proceedings of the Second European Conference on Computational Learning Theory, pp. 23-37 (1995)
20. Gallagher, N.: Simple Framework for XML. http://simple.sourceforge.net/home.php