Scalable and Real-time Sentiment Analysis of Twitter Data

Maria Karanasou, Anneta Ampla, Christos Doulkeridis and Maria Halkidi
Department of Digital Systems, School of Information and Communication Technologies
University of Piraeus, Piraeus, Greece
Email:karanasou@gmail.com, anneta.ampla@hotmail.com, {cdoulk,mhalk}@unipi.gr

***Abstract*—In this paper, we present a system for scalable and real-time sentiment analysis of Twitter data. The proposed system relies on feature extraction from tweets, using both morphological features and semantic information. For the sentiment analysis task, we adopt a supervised learning approach, where we train various classifiers based on the extracted features. Finally, we present the design and implementation of a real-time system architecture in Storm, which contains the feature extraction and classification tasks, and scales well with respect to input data size and data arrival rate. By means of an experimental evaluation, we demonstrate the merits of the proposed system, both in terms of classification accuracy as well as scalability and performance.**

I. Introduction

Online social networking platforms (such as Twitter, Tumblr, Weibo) where users are enabled to send short messages and express opinions on specific topics and their sentiments on them have increased rapidly the last few years. The amount of posted information keeps increasing to unimaginable levels. Thus the requirement for data analysis techniques that process posts online and assist with extracting interesting patterns of knowledge from them is stronger than ever.

Sentiment analysis and opinion mining have attracted the attention of the research community lately, due to numerous applications that are related to automated processing and analysis of text corpora. Traditional sentiment analysis approaches have been designed for static and well-controlled scenarios [11]. In microblogging environment the real-time interaction is a key feature and thus the ability to automatically analyze information and predict user sentiments as discussions develop is a challenging issue. The challenges that data analysis has to tackle in case of microblogging data is the use of informal, abbreviated, evolving language as well as the lack of information due to the short messages that are exchanged.

In this paper, we address the above challenges designing a scalable real-time sentiment analysis system. We proposed a methodology for extracting useful features from posts in order to represent them in sentiment analysis process. Moreover we developed a scalable system that processes tweets in real-time and uses supervised learning techniques to predict their sentiments. Our sentiment analysis models are adapted to the evolution of microblogging data exploiting the feedback that experts provide.

Summarizing, the main contributions of this paper are as follows:

- We develop a framework for sentiment analysis of Twitter data based on supervised learning techniques. The main components of this framework consist of: (i) a preprocessing module that assists with refining the data collection and selecting the features that properly represent the Twitter data (ii) a supervised learning module that aims to identify the sentiment polarity in Twitter data and properly classify them.
- We study the use of ensemble learning methods in the context of sentiment analysis, and we present the use of a feedback mechanism in the sentiment analysis process that is adaptable to dynamic contents.
- We design a real-time system architecture based on Storm to deal with evolution and volume of Twitter data.
- We evaluate our approach using various datasets. The collection of tweets is selected so that it contains a variety of words, expressions, emotional signals as well as indicative examples of sarcastic, ironic, metaphoric language. Also we conducted experiments considering the combination of multiple features (incl. prior polarity, text similarity, pattern detection).

The rest of this paper is organized as follows: Section II provides an overview of related work. Section III describes an overview of our approach, including the feature extraction and classification. In Section IV, we present the system implementation for real-time sentiment analysis using Storm. In Section V, we present the experimental study, and in Section VI we conclude the paper.

II. Related Work

In this section we briefly discuss approaches related to sentiment analysis in microblogging data. For a brief survey we refer to [9], while we point to [5], [6] for a recent overview of the topic of Big Social Data Analysis.

Scalable Sentiment Analysis. Scalable systems for sentiment analysis can be categorized in real-time systems [11], [24] and systems for batch processing [15]. In [24], a system is presented for real-time sentiment analysis on Twitter streaming data towards presidential candidates (US 2012). Results are delivered continuously and instantly, and feedback based on human annotation is proposed, however the online feedback loop and update of the trained model is left as future work. Real-time sentiment analysis is also targeted in [11] by means of transfer learning, where several challenges are identified,

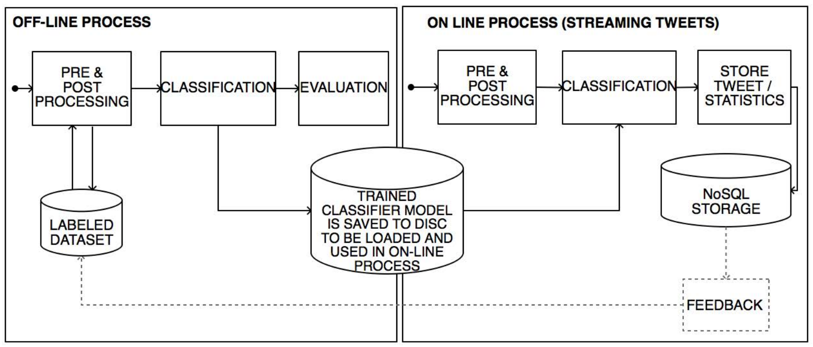


Fig. 1. Overview of our approach.

including highly dynamic textual content, changes in vocabulary, subjective meaning, and the lack of labeled data. In the case of batch processing, a system for sentiment analysis that is built on Hadoop/HBase is presented in [15], and consists of a lexicon builder (a word graph) and sentiment classifier. The Adaptive Logistics Regression is used from Mahout lib, and runs as a MapReduce job where each mapper is a classifier instance.

Sentiment Analysis for Twitter. Several works target sentiment analysis without focusing explicitly on the scalability aspect. Recently, an ensemble classification approach is investigated for airline services based on Twitter [23]. In [3], an approach is presented for sentiment analysis on global entities, such as "Michael Jackson". The aim of this work is to use the Twitter corpus to ascertain the opinion about entities that matter and enable consumption of these opinions in a user-friendly way. In [20], several surveys on consumer confidence and political opinion over the 2008 to 2009 period are analyzed, and they are found to be correlated to sentiment word frequencies in contemporaneous Twitter messages. In [4], tweets are analyzed in an attempt to investigate the correlation of the collective mood state to the value of stocks. In our previous work [14], we proposed a method for sentiment analysis of figurative language on Twitter, where the challenge is to identify sarcasm and irony in tweets. A probabilistic graphical model is proposed in [13] for mining topics and user communities, which associates topic-specific sentiments and behaviors with each user community. In [22], an approach for propagation-based sentiment analysis is proposed, which exploits both labeled and unlabeled data for learning, aiming at providing an adaptive solution for dynamically changing text corpora (tweets) that requires less manual effort. Also, linguistic rules have been used together with concept-level knowledge bases to improve sentiment analysis [7].

Sentiment Analysis in Generic Text Corpora. There exists a bunch of papers that study sentiment analysis in various text corpora, other than tweets (cf. [18]). In [16], the interplay between sentiment and other factors is studied, such as gender, age, education, focusing on data from Yahoo! Answers. A system that assigns scores indicating positive or negative opinion to each distinct entity in the text corpus is presented in [10], focusing on news and blogs. The approach consists of two phases: a sentiment identification phase, which associates expressed opinions with each relevant entity, and a sentiment aggregation and scoring phase, which scores each entity relative to others in the same class. Sentiment analysis on product reviews from Amazon is studied in [12].

III. Our Sentiment Analysis Approach

In this section we present our sentiment analysis approach which is based on supervised learning methods. It is a two-phase approach consisting of an offline and an online process as Fig. 1 depicts.

In the offline process, a labeled dataset of tweets is preprocessed in order to extract useful features, and then it is used to train a classifier. After training, the classification model is stored on secondary storage, in order to be loaded and used in the online phase.

In the online process, tweets from the Twitter stream are received, preprocessed and features are extracted. Then, each tweet (more accurately its representation using features) is given as input to the classifier, which has already loaded the classification model, and is able to predict the sentiment of the tweet. In addition, various statistics are computed, updated and stored to persistent storage.

In the online process, our system aims to deal with evolving data. The evolution of data is related to the rate with which tweets arrive as well as the changes in the textual content (e.g. changes in vocabulary, meaning of words etc). Our classification model adapts to the evolution of tweets exploiting the user (or domain expert) feedback. Thus we considered that the classifier is periodically evaluated based on feedback it receives and if the accuracy of classifier seems to decrease significantly, the classifier is retrained with an updated set of data (i.e., the offline process is re-applied). Due to the separation of these processes, we are able to use more than one model to make predictions on streaming data, thus, evaluate more than one model with the use of feedback.

A. Offline Phase: Feature Extraction

The goal of this phase is to build the model that is used to predict the sentiment of tweets. At the heart of the

Feature	Description
$HT(t)$	Hashtag categorization
HASHTAG-LEXICON-SUM	Same preprocessing for hashtags average of hashtags scores (from NRCHashtag lexicon [19])
POS-SMILEY	Presence of common positive emoticons
NEG-SMILEY	Presence of common negative emoticons
OH-SO DONT-YOU AS-*-AS-*	Presence of patterns “Oh so*”, “Don’t you*”,and “As * as *” that may indicate ironic or sarcastic text
CAPITAL	Presence of capitalized words
MULTIPLE-CHARS-IN-ROW	Presence of multiple characters
LINK	Presence of urls
NEGATION	Presence of negating words
REFERENCE	Presence of user mentions, e.g. @user
QUESTIONMARK	Presence of ”?”
EXCLAMATION	Presence of ”!”
FULLSTOP	Presence of more than 2 consecutive dots
LAUGH	Presence of common laughter indications, such as haha, lol, etc
PUNCT	The percentage of punctuation
RT	Presence of retweet
$sim(t)$	Text semantic similarity of tweet
POS-tags	Words to part of speech (POS) correspondence
POS-POSITION-i	Match between word position and part of speech
POLARITY	Polarity of a tweet t based on $swnScore(t)$
POLARITY-Words	Polarity of words w_i in a tweet as defined by $swnScore(w_i)$
IS-METAPHOR	True/False as described in the preprocessing section
SYN-SET-LENGTH	For each word, True if current word’s length is greater than the length of any of the word’s synonyms. False otherwise

TABLE I
OVERVIEW OF FEATURE EXTRACTION

proposed system, two main modules are employed: (a) the preprocessing, and (b) the classification module.

1) Preprocessing - Feature Extraction.: Data preprocessing is an important step in our sentiment analysis approach. It includes cleaning, transformation, feature extraction and selection. The feature extraction task is challenging due to various reasons, including the brevity of tweet messages, the noisy contents, and the fact that the language is informal, abbreviated and evolving. The result of data preprocessing is the final training set.

Given a set of posts (tweets) $\mathcal{T}$, the pre-processing module aims to extract from $\mathcal{T}$ a set of features that properly represent posts and provide useful information in the sentiment analysis process. Considering that $\mathcal{F} = \{f_1, \dots, f_n\}$ is the set of extracted features, each post t is represented as a vector (feature dictionary): $fd(t) : [f_1(t), \dots, f_n(t)]$, where $f_i(t)$ denotes the value of the feature f_i for post t.

Feature extraction regarding morphological features takes place before the cleaning of the text in order to avoid loss of information related to punctuation, urls and emoticons. This process checks if a tweet contains question marks or exclamation marks, capitalized words, urls, negations, laughter indications, retweet, positive/negative emoticons and hashtags. Table I summarizes the features extracted from a post to be used in our sentiment analysis approach.

Emoticons and hashtags classification. The emoticons and hashtags are classified based on the sentiment that they may convey. Specifically we manually classify the top-20 emoticons and some variations of them[1] as positive or negative. The hashtags identified in a post are classified as positive, negative or neutral based on SentiWordNet score ($swnScore$). SentiWordNet [2] is a lexical resource for opinion mining that assigns to each synset of WordNet three sentiment scores: positivity, negativity, objectivity. Based on this, for each hashtag h_t that is extracted from a post t we split it and spellcheck it (if necessary) and retrieve its $swnScore(h_t)$. Then, the feature, $HT(t)$ that indicates the hashtag classification of a post t is defined based on the number of positive, negative, and neutral hashtags present in the post t. That is:

$$HT(t) = \begin{cases} HT_{pos}(t) & c(htPos) > c(htNeg) > 0 \\ HT_{neu}(t) & c(htPos) = c(htNeg) = 0 \\ HT_{neg}(t) & c(htNeg) >= c(htPos) > 0 \end{cases}$$

where $c(htPos)$, $c(htNeg)$ denote the count of positive and negative hashtags in a tweet t respectively. The last hashtag is weighted more when the total hashtag polarity is calculated.

Pattern detection. Moreover, in the feature selection process we identify the presence of various patterns, such as: “Oh so*”, “Don’t you*”, and “As * as *”. Such patterns have been identified in previous work to indicate sarcastic or ironic

[1]http://datagenetics.com/blog/october52012

language, metaphors, and similes. All the above patterns, when present in a tweet, may affect the sentiment, and as such they are used as features.

Cleaning. The cleaning proceeds with punctuation, stop-words, urls, common emoticons and hashtags, references removal. Additionally, multiple consecutive letters in a word are reduced to two. Finally, spell-checking is performed to words that have been identified as misspelled in order to deduce the correct word. Then, we extract various features such as the aforementioned ones, which are summarized in Table I.

Semantic text similarity. After cleaning, the process continues with part-of-speech (POS) tagging that is performed with the use of a custom model [8]. Given a specific POS (e.g., VB), we denote as a_i a word in this category. We compute the similarity $sim(a_i, a_j)$ for each pair of words (a_i, a_j) of a tweet in a specific category. However a word a_i may have multiple synonyms. Let $SYN(a_i)$ denote the set $\{a_i^k\}_{k=1}^N$ of N synonyms of a_i including a_i itself. Then we take into account the synonyms of words in a tweet and we define the similarity of each pair of words in this tweet as the maximum similarity among all possible pairs of their synonyms. That is, $sim(a_i, a_j) = max\{sim(a_i^k, a_j^m)\}$, where $a_i^k \in SYN(a_i), a_j^m \in SYN(a_j)$. Hence, for a given category C that has n words $\{a_1, \dots, a_n\}$ in a tweet t, we compute the following vector[2]:

$sim_C(t) = [sim(a_1, a_2), \dots, sim(a_{n-1}, a_n)]$

Capitalizing on this, we compute the text semantic similarity $sim(t)$ of a tweet t as:

$$sim(t) = \frac{sim_{POS}(t)}{count(POS)}$$

where $sim_{POS}(t) = \sum sim_{VB}(t) + \sum sim_{NN}(t) + \sum sim_{ADJ}(t) + \sum sim_{RB}(t)$ and $count(POS) = count(VB) + count(NN) + count(ADJ) + count(RB)$.

Words that belong to the same part of speech are used to define the feature regarding semantic text similarity. Different similarity measures (Resnik's, Lin's, and Wu and Palmer's) are used [21], provided by nltk [3]. We employ Resnik's measure in our experiments.

Prior polarity. Moreover, the SentiWordNet score for each word in a tweet is calculated, ignoring words that have fewer than two letters. If the score of a word cannot be determined, then we calculate the SentiWordNet score of the stemmed word. If no score can be retrieved, then neutral is assumed. Given that the word w_i occurs n times in the SentiWordNet corpus, the total score of w_i is given by:

$$swnScore(w_i) = \frac{\sum_{i=1}^{n}(1/rank_i) * score_i}{\sum_{i=1}^{n} 1/rank_i}$$

where $score_i = 1 + PosScore_i - NegScore_i$, and $PosScore_i$ and $NegScore_i$ is the positive and negative score respectively of w_i in SentiWordNet. Also, $rank_i$ is given by SentiWordNet and corresponds to the weight of the score of a word, when this word belongs to a set of synsets. The index i of each word was used in an attempt to correlate each word's position with the calculated sentiment. Moreover, the total score of a tweet t is calculated as the average of SentiWordNet scores of the words in t, i.e., $swnScore(t) = \frac{\sum_{\forall w_i \in t} swnScore(w_i)}{\sum_{\forall w_i \in t} 1}$.

Based on the values of $swnScore$ the tweets are classified as *positive*, *somewhat positive (sPositive)*, *negative*, *somewhat negative (sNegative)*, or *neutral* as follows:

$$polarity(t) = \begin{cases} positive & swnScore(t) \geq 1.2 \\ negative & , swnScore(t) \leq 0.2 \\ neutral & , 0.95 \leq swnScore(t) \leq 1.05 \\ sNegative & , 0.2 < swnScore(t) < 0.95 \\ sPositive & , 1.05 < swnScore(t) < 1.2 \end{cases} \tag{1}$$

Similarly, the words of each tweet are categorized based on the values of their $swnScore(w_i)$ and eq. 1.

Detecting metaphors. Another feature that we extract from tweets is if a tweet can be considered as metaphor. The classification of tweet as metaphoric or not is based on the use of a classifier that we have properly trained. Specifically we consider a Linear SVM classifier that is trained with 12,000 tweets (collected from MetaphorMagnet, MetaphorMinute, and Twitter Sentiment Classification using Distant Supervision [4]). The trained classifier achieved 90% accuracy in the aforementioned dataset.

B. Offline Phase: Building the Classifier

The preprocessing step is applied to a set of tweets $\mathcal{T}$ and the result is the vector representation of tweets as described in section III-A1 . Let $fd(\mathcal{T})$ be the set of vectors representing the tweets in $\mathcal{T}$. The feature vectors of tweets $fd(t)$ are processed by a vectorizer [5] to produce a vector array which is passed to a Tfidf transformer [6].

Then we proceed with building the classification model. Our system has been designed to be independent of the classifier. In our study we have experimented with different learning algorithms (Linear SVM, Naive Bayes, Decision Tree, Stochastic gradient descent) while we also studied the use of ensemble learning approaches. Our experiments (see Section V) show that the SVM classifier seems to outperform the other considered classifiers. In ensemble learning, we generate the model for predicting tweet sentiments by combining the classification models built by different algorithms (Linear SVM, Decision Trees, and Stochastic gradient descent (SGD)). The *MaxVote* method (also known as Majority vote [17]) has been used for combining the predictions of classifiers.

[2]We consider the following categories: verbs (VB), nouns (NN), adjectives (ADJ), and adverbs (RB).

[3]The Natural Language Toolkit http://www.nltk.org/.

[4]http://help.sentiment140.com/for-students

[5]http://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html

[6]http://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html

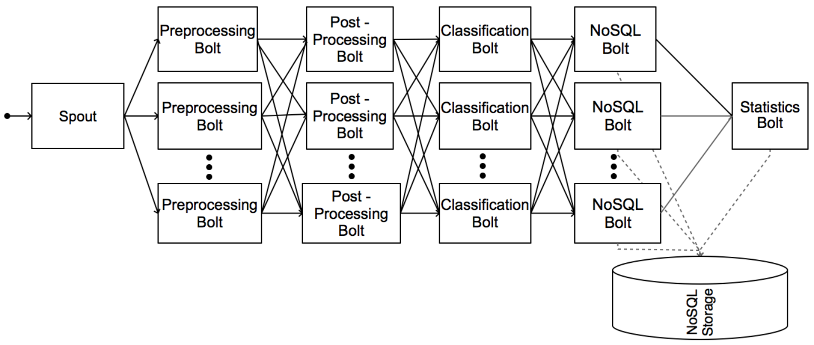


Fig. 2. Topology in Storm.

IV. Real-time Sentiment Analysis

In order to deliver a scalable and real-time solution for sentiment analysis, we implement our techniques in Storm [1]. Storm is a framework that targets real-time processing and analysis of data streams, with salient features such as scalability, parallelism and fault-tolerance.

Figure 2 shows the Storm-based topology of our system. There exists a node (called *Spout*) that acts as the source of Twitter data stream. Each process of our system is modeled as a node (called *Bolt*) of a *Topology* graph. Specifically in our system we have the following types of bolts: the preprocessing bolt, the post-processing bolt, the classification bolt, the NoSQL bolt and the statistics bolt. We have also to note that in our topology we have adopted the shuffle grouping partitioning approach, that is we consider random distribution of tuples across the tasks of bolts.

Each tweet from the input stream is preprocessed (includes cleaning and feature extraction), classified into one of the available classes, and stored into a NoSQL database for persistence. Since each tweet is independent, this process is highly scalable, meaning that for example, the number of preprocessing bolts can increase or decrease to accommodate the volume and speed of the stream as it is needed, without any issue other than hardware support.

In more detail, we collect from the Spout only English tweets and emit for each tweet a JSON string. In the preprocessing bolt, we extract the morphological features, we clean the Tweet's text, and extract other features. In the post-processing bolt, we select the features that will be used by the classifier, while we also perform grouping of SentiWordNet values. In the classification bolt, the classifier loads the pre-trained model, and for each incoming tweet its feature representation is passed from the model, and a prediction given the current model is made. The NoSQL bolt receives the classification results and stores them on disk for evaluation purposes. Finally, the statistics bolt accesses the classification results and computes various statistics that can be used to evaluate the quality of the predicted result. In our implementation, we use Shuffle grouping to distribute tuples to tasks for all bolts (random partitioning), except from the statistics bolt where Global grouping is used.

Furthermore, we employ a feedback mechanism. A small percentage of the tweets that go through the Storm topology are hand-annotated in order to evaluate the classification process. These tweets are then used to re-train the classifier and perform tests in order to conclude whether the use of feedback brings improvement. Thus, the re-trained model is loaded in the classification bolt and used in the topology.

V. Experimental Evaluation

In our implementation, we used Python to develop the sentiment analysis part (preprocessing, classification). The part of topology and components (spout/bolts) has been implemented in Java. Python scripts are called and run from Java bolts by implementing the Storm's multilang protocol.

Platform. We performed experiments on a cloud platform, provided by *Okeanos*[7], an IAAS service for the Greek Research and Academic Community. Due to limited availability of resources, we configured six virtual machines (VMs) with Ubuntu 14.04.2 LTS and installed Apache Storm v0.9.4: one VM was used to set up a single-node zookeeper, one is used as the master node and the remaining four VMs are used to host the topology spout and bolts with the following configuration: (a) spout and preprocessing, (b) post-processing, (c) classification, (d) NoSQL and statistics, respectively. Each VM has 2-4 CPUs, is equipped with 4-8GB RAM, and has disk size of 10-20GB. For instance, the configuration of the master node, which hosts Storms Nimbus deamon and Storm UI, is 4 CPUs, 6GB RAM and 10GB disk size. With respect to the NoSQL store, we used MongoDB v3.0.3 in our implementation.

Datasets. For our experimental study we have used both labeled datasets that are widely used in sentiment analysis literature and manually annotated ones. Specifically we experimented with SemEval 2015 Task 11, SemEval 2013 Task 2, emoticons-harvested (1.6M tweets[8]) and two manually annotated datasets of tweets: ManualTest that contains 480 tweets from 1.6M and 1,000 from the dataset used in [15] and ManualNeutral provided by [15]. Table II shows the details of the datasets. We have considered three-class labels (positive (1), negative (-1), neutral(0)) for tweets in all datasets. The

[7]https://okeanos.grnet.gr/home/

[8]http://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf

Dataset	**Total**	**Positive**	**Negative**	**Neutral**	**No emoticons**	**Positive & negative emoticons**
Task11	12,529	1,340	10,326	863	11,109	21
Task2b	9,059	3,349	1,351	4,359	6,359	170
"Emoticon-harvested"	60,000	30,000	30,000	0	0	0
ManualTest	1,480	758	441	281	1,202	13
ManualNeutral	26,336	0	0	26,336	23,810	16
Total	109,404	35,447	42,118	31,839	42,480	220

TABLE II
OVERVIEW OF DATASETS.

Dataset	**25K**	**50K**	**70K**	**90K**	**110K**
Task11	12,529	12,529	12,529	12,529	12,529
Task2b	9,059	9,059	9,059	9,059	9,059
"Emoticon-harvested"	0	20,000	40,000	60,000	60,000
ManualTest	1,480	1,480	1,480	1,480	1,480
ManualNeutral	1,000	10,000	10,000	10,000	26,336
Total	24,068	53,068	73,068	93,068	109,404

TABLE III
OVERVIEW OF DATASETS OF VARYING SIZE.

datasets were split to training and testing sets (80-20 or 60-40). We note that the percentage of positive, negative and neutral is selected to be the same in both train and test.

Also, we constructed datasets of varying size that have the settings presented in Table III to conduct experiments with different volumes of data. In case of the 50K datasets we use the whole Task11, Task2b, ManualTest, a subset of 10,000 tweets from ManualNeutral and 20,000 tweets from "Emoticon-harvested" tweets (10,000 positive and 10,000 negative). For the 70K and 90K datasets, we increase the number of considered tweets by 20,000 (half positive and half negative) only in the "Emoticon-harvested" dataset. The 110K datasets contain all the tweets of the Task11, Task2b, Manualtest and ManualNeutral datasets. In case of "Emoticon-harvested", we keep 60,000 tweets in order to have a relatively balanced dataset in terms of polarity.

Parameters. We study the effect of (a) different dataset synthesis, (b) varying the dataset size, (c) varying the different features used, and (d) testing different classifiers.

Metrics. We evaluate the classification process using:

- *Accuracy* that is defined as the proportion of correctly classified tweets among the total number of tweets processed (N), i.e. $accuracy = \frac{\text{num of correctly classified tweets}}{\text{number of tweets}}$
- The *mean square error (MSE)* of the classifier which is defined as $MSE = \frac{\sum_{i=1}^{N}(X_i - \hat{X}_i)^2}{N}$, where N is the number of tweets, $\hat{X}_i$ is the predicted label of the i-th tweet and X_i is the actual classification label of $i-th$ tweet. We note that the MSE of a classifier is calculated considering the value of tweet polarity instead of its label (e.g. if a tweet is classified as positive we consider the value of its polarity that is 1).
- The *cosine similarity* between predicted labels of tweets $\hat{X}$ and actual classification labels X, that is given by $cos(X, \hat{X}) = \frac{\sum_{i=1}^{N} X_i \cdot \hat{X}_i}{\sqrt{\sum_{i=1}^{N} X_i^2} \cdot \sqrt{\sum_{i=1}^{N} \hat{X}_i^2}}$

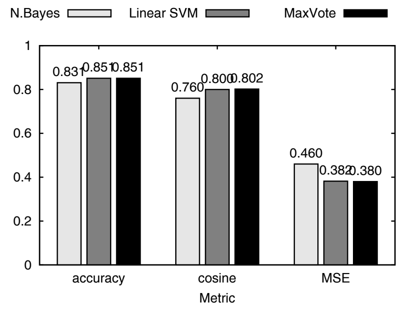
Algorithms. We also used Linear SVM, Naive Bayes trained with all features, and the MaxVote technique (also known as Majority vote [17]) for the ensemble classifier that combines the following three classifiers: Linear SVM, Decision Trees, and SGD. In all cases, we performed 10-fold cross validation to reduce the effect of variance.

A. *Qualitative Results*

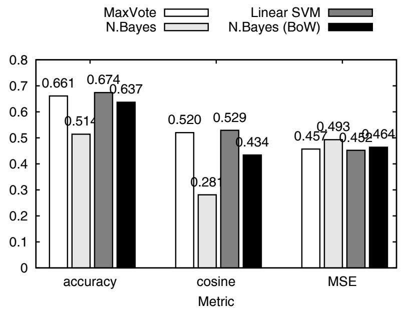
Results with different classifiers. Figs. 3(a) and 3(b) present the accuracy of different classifiers applied to Task11 and Task2b datasets, respectively. The results show that LinearSVM and MaxVote exhibit the best performance with respect to all metrics (accuracy, cosine, and MSE) consistently. Similar trends we have observed when we experimented with all the considered datasets.

Effect of varying dataset size. The main purpose of this series of experiments was to see how the dataset volume affects the classification results. Fig. 4(a) shows the results obtained when we considered the LinearSVM classifier, all features, and the size of datasets varies from 25K to 110K tweets. As a general trend, all metrics are improved as the size of datasets increases. Table IV depicts how the classification accuracy of SVM classifier changes with respect to the dataset size. In our study we increase the volume of our data sets by adding more data samples from the "Emoticon-harvested" dataset. We can observer that the size of the training data set slightly affects the classification results of all the considered datasets except for Emoticons harvested. This happens because the additional samples provide new knowledge only for the "Emoticon-harvested" data. It is obvious then that the quality depends not only from the number of training samples but also from the quality of them.

Effect of different features. In addition to data volume, we conducted experiments with different combinations of features. Specifically we experimented with and without the use of emoticons as features. Other studies[8] on the "emoticons-harvested" dataset have shown that the use of emoticons decreases the classification performance. In our case, as shown



(a) All features for Task11.

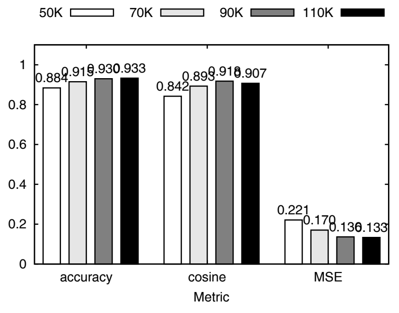


(b) All features for Task2b.

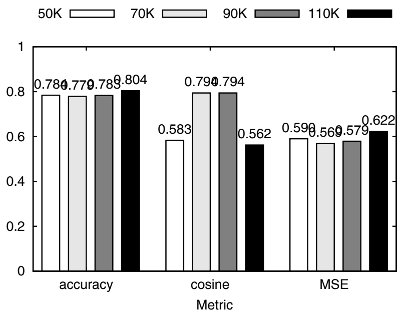
Fig. 3. All features for Task11 and Task2b.

Dataset	50K	70K	90K	110K
Task11	0.844	0.832	0.831	0.834
Task2b	0.641	0.625	0.617	0.621
"Emoticon-harvested"	0.988	0.993	0.995	0.993
ManualTest	0.588	0.569	0.560	0.552
ManualNeutral	0.987	0.986	0.986	0.989

TABLE IV
ACCURACY OF INDIVIDUAL DATASETS AS SIZE INCREASES.



(a) All features.



(b) No emoticons.

Fig. 4. Linear SVM with varying dataset size.

in Fig. 4(b), the accuracy decreases when the emoticons are left out, even though there is a significant amount of tweets (e.g 40% of 110K tweets) that contain neither positive nor negative emoticons (from the ones we can identify).

Effect of feedback. We observed that in most cases accuracy increases with the volume of the dataset. However, this does not guarantee that the classification model will be accurate when used to make predictions on streaming data. Therefore,

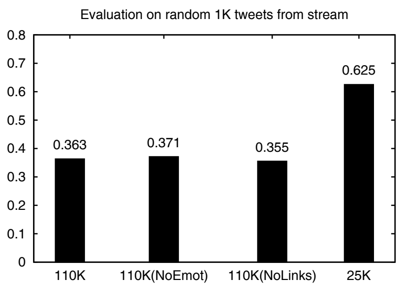


Fig. 5. Evaluation of feedback on the Twitter stream.

we conducted a feedback experiment. A thousand random tweets gathered from the online process were manually corrected and embodied in the dataset that was used to train and test the classifier. Basic tests are re-run with the new dataset.

Based on feedback results (Table V), accuracy increases with feedback when "emoticon-harvested" tweets are not a major part in the dataset. This happens because the process of categorizing positive/ negative tweets based on emoticons only is more coarse grained than using manual evaluation.

In our tests we concurrently used four classifier models that would be interesting to evaluate with the use of feedback on streaming data. After observing an amount of the predictions, it seems that 110K model does not perform well, because it classifies the majority of tweets as negative. However, the majority of the tweets classified as neutral and positive by this model is correct. The manual annotation results of a random sample of 1,000 tweets is presented in Fig. 5. We observe that the 25K model has the best performance in this data sample.

Discussion. Our experimental study shows that SVM gave the most accurate classification results and thus it is also used in the on-line process experiments.

Parallelism hint	Rate of tweets generated by Twitter	Rate of tweets processed by our system
1	1,008 tweets/min	673 tweets/min
2	1,535 tweets/min	1,535 tweets/min
3	1,475 tweets/min	1,475 tweets/min
4	1,573 tweets/min	1,573 tweets/min

TABLE VI
PERCENTAGE OF PROCESSED TWEETS DEPENDING ON PARALLELISM.

Dataset Settings	**25K + No Emoticon harvested**	**25K + 2K Emoticon harvested**	**25K + 10K Emoticon harvested**	**50K**	**70K**	**90K**	**110K**
No feedback	0.762	0.776	0.839	0.883	0.913	0.929	0.936
5K feedback	0.769	0.778	0.833	0.872	0.904	0.922	0.928

TABLE V
ACCURACY WITH RESPECT TO FEEDBACK

B. Performance Results

We conducted an experiment that aims to assess the effect of parallelism to the scalability of our system. It should be noted that the rate of tweet generation varies based on time, since more tweets per second are created at different times of the day. As we restrict the tweets that are processed by our system to only those using English language, we observed tweet generation rates in the range of 1,000-2,000 tweets/minute[9], during the last weeks of October 2015. This is the throughput requirement for our system, when all processing tasks (feature extraction, classification, etc) are taken into account.

Initially, we run our system with no parallelism at all. It turned out that the rate of tweets generated by the stream, was higher than the achieved throughput of our system. This motivates the need for designing a scalable solution for real-time sentiment analysis in Twitter, as the processing overhead imposed by the various modules (e.g., feature extraction) can be significant. We increased the parallelism by providing different parallelism hints to our topology. Our experiments showed that even when using moderate parallelism (two instances of each task), we achieved to process all incoming tweets and match the rate of tweet generation. Table VI summarizes the obtained results.

VI. Conclusions

In this paper, we presented a real-time system to identify the sentiment polarity in microblogging. Innovative features of our approach include: (a) the use of train datasets with contents of high variety, (b) the provision of a feedback mechanism that is adaptable to dynamic contents, (c) the combination of multiple features (incl. prior polarity, text similarity, pattern detection), (d) the use of ensemble learning methods, and (e) a scalable system implementation for real-time sentiment analysis in Storm. Our experimental study shows that Part-Of-Speech tags, Emoticons and prior polarity are the most significant features in the sentiment analysis of Twitter data. In the online process, our study shows that the feedback may contribute to the classification accuracy, especially when "emoticon-harvested" tweets are not used.

Acknowledgment

This work has been co-financed by ESF and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Aristeia II, Project: ROADRUNNER. This work has been partly supported by the University of Piraeus Research Center.

[9]The public Streaming API sample endpoints are reported to provide approximately 1% of the public Tweet volumes at any time.

References

[1] Storm: Distributed and fault-tolerant real-time computation. http://storm.apache.org/.

[2] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proc. of LREC*, 2010.

[3] S. Batra and D. Rao. Entity based sentiment analysis on twitter. *Science*, 9(4):1–12, 2010.

[4] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *J. Comput. Science*, 2(1):1–8, 2011.

[5] E. Cambria, N. Howard, Y. Xia, and T. Chua. Computational intelligence for big social data analysis [guest editorial]. *IEEE Comp. Int. Mag.*, 11(3):8–9, 2016.

[6] E. Cambria, H. Wang, and B. White. Guest editorial: Big social data analysis. *Knowl.-Based Syst.*, 69:1–2, 2014.

[7] P. Chikersal, S. Poria, E. Cambria, A. F. Gelbukh, and C. E. Siong. Modelling public sentiment in twitter: Using linguistic patterns to enhance supervised learning. In *Proc. of CICLING*, pages 49–65, 2015.

[8] L. Derczynski, A. Ritter, S. Clark, and K. Bontcheva. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proc. of RANLP*, pages 198–206, 2013.

[9] R. Feldman. Techniques and applications for sentiment analysis. *Commun. ACM*, 56(4):82–89, 2013.

[10] N. Godbole, M. Srinivasaiah, and S. Skiena. Large-scale sentiment analysis for news and blogs. In *Proc. of ICWSM*, 2007.

[11] P. H. C. Guerra, A. Veloso, W. M. Jr., and V. Almeida. From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In *Proc. of SIGKDD*, pages 150–158, 2011.

[12] Y. Haimovitch, K. Crammer, and S. Mannor. More is better: Large scale partially-supervised sentiment classication. In *Proc. of ACML*, pages 175–190, 2012.

[13] T. Hoang, W. W. Cohen, and E. Lim. On modeling community behaviors and sentiments in microblogging. In *Proc. of ICDM*, pages 479–487, 2014.

[14] M. Karanasou, C. Doulkeridis, and M. Halkidi. DsUniPi: An SVM-based approach for sentiment analysis of figurative language on Twitter. In *Proc. of SemEval*, 2015.

[15] V. N. Khuc, C. Shivade, R. Ramnath, and J. Ramanathan. Towards building large-scale distributed systems for twitter sentiment analysis. In *Proc. of SAC*, pages 459–464, 2012.

[16] O. Kucuktunc, B. B. Cambazoglu, I. Weber, and H. Ferhatosmanoglu. A large-scale sentiment analysis for yahoo! answers. In *Proc. of WSDM*, pages 633–642, 2012.

[17] L. I. Kuncheva. *Combining Pattern Classifiers, Methods and Algorithms (2nd Edition).* Wiley, 2014.

[18] B. Liu. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing, Second Edition.*, pages 627–666. 2010.

[19] S. Mohammad. #emotional tweets. In *Proc. of *SEM*, pages 246–255, 2012.

[20] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *Proc. of ICWSM*, 2010.

[21] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet: : Similarity - measuring the relatedness of concepts. In *Proc. of HLT-NAACL (Demos)*, pages 1024–1025, 2004.

[22] J. Tang, C. Nobata, A. Dong, Y. Chang, and H. Liu. Propagation-based sentiment analysis for microblogging data. In *Proc. of ICDM*, pages 577–585, 2015.

[23] Y. Wan and Q. Gao. An ensemble sentiment classification system of Twitter data for airline services analysis. In *Proc. of ICDMW*, pages 1318–1325, 2015.

[24] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan. A system for real-time twitter sentiment analysis of 2012 U.S. presidential election cycle. In *Proc. of ACL (Demos)*, pages 115–120, 2012.