If You are Happy and You Know It... Tweet

Amir Asiaee T.
CSE, University of Minnesota
MN 55455 USA
ataheri@cs.umn.edu

Mariano Tepper
ECE, Duke University
NC 27707 USA
mariano.tepper@duke.edu

Arindam Banerjee
CSE, University of Minnesota
MN 55455 USA
banerjee@cs.umn.edu

Guillermo Sapiro
ECE, Duke University
NC 27707 USA
guillermo.sapiro@duke.edu

ABSTRACT

Extracting sentiment from Twitter data is one of the fundamental problems in social media analytics. Twitter's length constraint renders determining the positive/negative sentiment of a tweet difficult, even for a human judge. In this work we present a general framework for per-tweet (in contrast with batches of tweets) sentiment analysis which consists of: (1) extracting tweets about a desired target subject, (2) separating tweets with sentiment, and (3) setting apart positive from negative tweets. For each step, we study the performance of a number of classical and new machine learning algorithms. We also show that the intrinsic sparsity of tweets allows performing classification in a low dimensional space, via random projections, without losing accuracy. In addition, we present weighted variants of all employed algorithms, exploiting the available labeling uncertainty, which further improve classification accuracy. Finally, we show that spatially aggregating our per-tweet classification results produces a very satisfactory outcome, making our approach a good candidate for batch tweet sentiment analysis.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms

Algorithms, Experimentation

Keywords

Twitter sentiment analysis, compressed learning, supervised learning, sparse modeling, Bayes classification, SVM.

1. INTRODUCTION

Twitter, a micro-blogging service, is among the most pervasive social media services. On a regular basis, its users *willingly* share their thoughts, preferences, and emotions, in the form of (up to) 140-characters length messages (a.k.a. tweets). Although the field of social media analytics for rich sources of information, like weblogs, is becoming mature, micro-blog analysis is in its initial stages. Recent studies [5, 6] have shown the predictive value of Twitter content in domains such as marketing, business, and politics.

Several data mining tasks can be defined for Twitter data, considering very diverse applications. Among them, sentiment analysis [16] has increasingly gained attention. Sentiment analysis tries to identify the positive or negative sentiment of a corpus. Detecting major events based on tweets' sentiments [1, 5, 6], and finding the pattern of temporal happiness and mood in human behavior [9, 11] are examples of applications of Twitter sentiment analysis.

In spite of the growing literature of Twitter sentiment analysis, there are several issues that limit its usage in practice. Ignoring the objects, individuals, or products that a tweet is expressing emotion about, is a major gap between current state of the art approaches and practical applications of Twitter sentiment analysis; in practice we are interested in discovering people's feelings about a certain product, topic, or in general a target [13]. There has been initial work [13] on target-dependent sentiment analysis. We name the process of separating tweets that are related to a target of interest *target extraction*.

In general, tweets do not always express sentiments. They may contain information, facts, or any kind of objective expressions. Thus, before actual sentiment analysis, polar tweets (i.e., those with sentiment) should be separated from neutral ones. Recently tweet analysis literature has also considered neutral tweets in classification [1]. We refer to this step as *sentiment extraction*.

The three steps required for sentiment analysis are delineated in Figure 1. Target extraction distinguishes between tweets that are related to our topic of interest and unrelated ones [13]. Sentiment extraction separates tweets with emotional content. This is sometimes referred to as the polar-neutral classification problem [1, 3, 15]. Finally, *sentiment classification* sets apart positive from negative tweets.

To the best of our knowledge, the present work is the first attempt that addresses all the aforementioned tasks (Figure 1) together for sentiment analysis of *single* tweets. Specially target extraction is a major step that is missing in related works [13]. Every step is formulated as 2-class classification problem, and several supervised learning methods were developed and evaluated. Unlike previous works

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

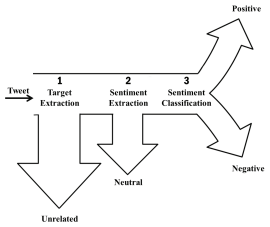


Figure 1: Cascade classifier for Twitter sentiment analysis.

that use sophisticated language features [1, 3] with heavy pre-processing, we show that high accuracy in each step is achievable by just using bag-of-words as the classification input. In addition, we show that this *sparse* high-dimensional data can be projected in a low-dimensional space, using random reconstructible projections, without losing performance accuracy. Besides single label per tweet, our database has multiple labels (soft label) for each tweet which enables us to perform confidence weighted classification as well.

Multiple studies try to circumvent the scarcity of per-tweet information by analyzing batches of tweets. These batches can be built using different criteria, such as spatial (location of senders), temporal (time of post), or by author. The most common methods for batch analysis are lexicon-based, which use pre-compiled lists of polar words as indicators of the sentiment type [5, 6, 9, 11]. We supplement our work with sentiment classification for spatially aggregated tweets and show that performing per-tweet sentiment analysis followed by aggregation, results in high accuracy.

The rest of this paper is organized as follows. Labeling, pre-processing, and classification algorithms are discussed in Section 2. In Section 3 we present experimental results and detailed comparisons of several methods. Finally, in Section 4 we provide concluding remarks.

2. THE CASCADE CLASSIFIER

In this section we explain how the data is labeled. Then we comment on the preprocessing procedure and discuss the method for compression of the sparse input vector. Finally we explain classification methods and their weighted variants that are being used in the cascade classifier.

2.1 Labeling the Data

Supervised learning algorithms rely on the availability of labeled data. The use of specific words to label tweets is common. For example, Pak and Paroubek [15] use an emoticon list as indicator for positive or negative content, and Bollen et al. [5] use the phrase "I feel" to label tweets as polar (i.e., expressing emotions). Other work, e.g., [3, 10], gather noisy labels from multiple sources, like unevaluated sentiment analysis tools, and then incorporate uncertainty into the classification algorithm.

In recent years, crowdsourcing has emerged as a cost-effective way of carrying out labor-intensive tasks. In this work, the process of data labeling is crowdsourced as a part of the Dialogue Earth Project (`www.dialogueearth.org`).

First gross filtering on a collection of tweets is performed and tweets that do not contain any of the indicator words (i.e., words that are associated to the target) are filtered out. The data is then hand labeled by several evaluators with 4 labels: positive, negative, neutral, and not related to the target topic. The disagreement between evaluators shows the inherent difficulty of the task at hand. An additional label is reserved for cases in which the evaluator cannot assign a tweet to any of the aforementioned classes. Tweets that majority of evaluators can not assign any label to are discarded from further consideration.

Let C be the set of all classes. For each tweet i, evaluator j chooses a class label $\mathbf{e}_{ij}$, that is a $|C|$ dimensional vector in which one element equals 1 and all remaining elements are zero. By normalizing the sum of these vectorial labels for each tweet i we get our soft vector label as $\boldsymbol{\omega}_i = (\sum_{j=1}^{r} \mathbf{e}_{ij})/r$, where r is the number of evaluators.

Now we can work with two variants of the label set. The first one is soft labels contained in each $\boldsymbol{\omega}_i$, $Y = \{\omega_{ic} \in [0,1] \mid i = 1,\ldots,n; c = 1,\ldots,|C|\}$, where ω_{ic} represents the confidence of label c for data point i and n is number of tweets. Alternatively we can consider hard labels derived from $\boldsymbol{\omega}_i$ that is $\tilde{Y} = \{y_i = \text{argmax}_{c\in C}\omega_{ic} \mid i = 1,\ldots,n\}$, and thus falling back into usual classification in which each data point has a single label, referred to here as *dominant* labels. Finally, we name two classes that we classify them in each step of cascade classifier C_1 and C_2.

2.2 Preprocessing

Since we use the bag-of-words model for representing the tweets, our preprocessing is very simple. We begin by extracting the words, i.e., actual words and also numbers, usernames, emoticons, URLs, etc. Actual value/content of numbers, usernames, and URLs is not important for us and thus we replace them by special generic identifiers. We do remove re-tweet signs (RT), special characters (not contained in emoticons), and stop words. Note that we keep emotional stop words and negations [15] since they are crucial for sentiment analysis. Hashtags are Twitter tags which are often a concatenation of words (e.g., '#loveThisWeather'); when the words in a hashtag begin with an uppercase letters, we break it into separate words.

We spell check the words using three dictionaries: an English dictionary, a Twitter dictionary which contains specific lingo, and an emoticon dictionary.

Finally, we perform another step which empirically proved to be effective in both speed and accuracy of classifications. Words that appeared in the cleaned database less than thrice are pruned. Also we remove words that are highly frequent and their frequencies in C_1 and C_2 are close. A word is highly frequent in a class if its frequency is more than 0.05 in the tweets of that class. Frequencies of a word in two classes are close if their difference is less that 0.2.

2.3 Representing and Compressing Tweets

In the bag-of-words model a document is simply represented as an unordered collection of words T. A predefined set of words $\mathcal{W} = \{w_i \mid i = 1,\ldots,d\}$ is then used to build a d-dimensional feature vector $\mathbf{v}$ for each document T such that $(\forall i = 1,\ldots,d)\ \mathbf{v}(i) = \#(T, w_i)$, where $\#(T, w_i)$ is the number of times word w_i appears in T. Here $d \approx 10^4$ but since the tweet's length is limited, $\mathbf{v}$ is extremely sparse.

Although we have done experiments using the original high-dimensional bag-of-words vector $\mathbf{v}$, we extend our experiments and also use low dimensional projection of it as input. Using random reconstructible projection, we project (compress) the sparse vector $\mathbf{v}$ to lower dimension and perform classification in that domain. Projection to lower dimensional space is also known as the hashing trick [18].

We use an $m \times d$ matrix $\mathbf{P}$ ($m \ll d$) to create a compressed representation $\mathbf{x} = \mathbf{P}\mathbf{v}$ of a feature vector $\mathbf{v}$ in such a way that m is as small as possible and $\mathbf{v}$ can be reconstructed from $\mathbf{x}$. $\mathbf{P}$ is a random matrix, i.e., its entries p_{ij} are sampled from i.i.d. random variables [12]. We build $\mathbf{P}$ by sampling its entries p_{ij} from a Gaussian distribution $\mathcal{N}(0, 1/m)$. The value of m is chosen such that $m = O(h \log(d/h))$ where $h = \max_{\mathbf{v} \in V} \|\mathbf{v}\|_0$ (while $d \approx 10^4$, $h \approx 20$).

To conclude, the set of (one per-tweet) feature vectors $V = \{\mathbf{v}_i \mid i = 1, \ldots, n\}$ is represented in the compressed domain by a set of vectors $X = \{\mathbf{x}_i \mid i = 1, \ldots, n\}$, where $\mathbf{x}_i = \mathbf{P}_{m \times d}\, \mathbf{v}_i$.

2.4 Classification Methods

Several well-known supervised learning algorithms and their weighted variant have been used for classification of each step. We start by explaining the newest method which is based on dictionary learning [17] and subsequently discuss weighted variant of Support Vector Machine(SVM) [4], K Nearest Neighbor(KNN) [4], and Naïve Bayes(NB) [4] briefly.

2.4.1 Sparse Modeling Approach to Classification

In sparse coding a signal is approximated with a linear combination of a few elements (atoms) of some (often) redundant basis. When these bases are learned from the data itself, they are usually called dictionaries [7].

Formally, we aim at learning a dictionary $\mathbf{D} \in \mathbb{R}^{m \times k}$ such that a training set of signals $X = \{\mathbf{x}_i \in \mathbb{R}^m \mid i = 1, \ldots, n\}$ (and later testing data from the same class) can be well represented by linearly combining a few of the basis vectors formed by the columns of $\mathbf{D}$. This problem can be casted as the following optimization:

$$\min_{\substack{\mathbf{D}, \boldsymbol{\alpha}_i \\ i=1,\ldots,n}} \frac{1}{n} \sum_{i=1}^{n} \tfrac{1}{2} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1, \lambda > 0, \qquad (1)$$

which is convex with respect to $\boldsymbol{\alpha}_i$s when $\mathbf{D}$ is fixed and viceversa. The optimization is then commonly solved by alternatively fixing one and minimizing over the other. Fixing $\mathbf{D}$ and solving for $\boldsymbol{\alpha}_i$s is known as *sparse coding* and finding $\mathbf{D}$ for fixed $\boldsymbol{\alpha}_i$s is referred to as *dictionary learning*. We use publicly available SPAMS library (`www.di.ens.fr/willow/SPAMS`) for solving (1)

Sparse modeling has been previously employed for supervised classification tasks [17]. Classification is often done by first learning, following the above optimization, a dictionary $\mathbf{D}_c$ for each class $c \in C$ using only training data from the set $\{\mathbf{x}_i \in X \mid y_i = c\}$. Classification is then performed with testing data X_{test}, assigning a label $c^* = f(\mathbf{x})$ to each $\mathbf{x} \in X_{\text{test}}$ where $f(\mathbf{x}) = \text{argmin}_{c \in C}\ \ell(\mathbf{x}, \mathbf{D}_c)$, and $\ell(\mathbf{x}, \mathbf{D}_c) = \min_{\alpha} \frac{1}{2} \|\mathbf{x} - \mathbf{D}_c\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1$.

2.4.2 Confidence Weighted Classification

Dictionary Learning: For exploiting the possible available information in the non-binary confidence ω_{ic} introduced in section 2.1, we redefine the optimization (1) as:

$$\min_{\substack{\mathbf{D_c}, \boldsymbol{\alpha}_i \\ i=1,\ldots,n}} \frac{1}{n} \sum_{i=1}^{n} \omega_{ic} (\tfrac{1}{2} \|\mathbf{x}_i - \mathbf{D_c}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1), \lambda > 0. \qquad (2)$$

The closer ω_{ic} is to one, the more $\mathbf{x}_i$ contributes to class c dictionary. We then solve (2) by alternating minimization on $\boldsymbol{\alpha}_i$s and $\mathbf{D}_c$. The implementation is obtained by adding the weights to the SPAMS library.

Naïve Bayes: A naïve Bayes classifier, assigns the maximum a posteriori class to each test data point. Assuming conditional independence of the input vector's features, classification simplifies to $y_i = \text{argmax}_{c \in C} P(c) \prod_{j=1}^{d} P(v_{ij}|c)$, where $P(c)$ and $P(v_{ij}|c)$s are computed from their corresponding frequencies in the training data.

We incorporate the confidence weights in the naïve Bayes formulation. Methods for using soft labels in naïve Bayes have been proposed previously [14], here we present a simple approach in which weights are used to compute $P(c) = \frac{\sum_i \omega_{ic}}{\sum_c \sum_i \omega_{ic}}$ and $P(v_{ij}|c) = \frac{\sum_i \omega_{ic} \times v_{ij}}{\sum_i \omega_{ic}}$.

K Nearest Neighbor: In K Nearest Neighbor, class label is assigned to each test data point based on the labels of K closest training examples in the feature space. In order to use the weight information in KNN, instead of majority voting between the K nearest neighbor of $\mathbf{v_i}$, we add their K confidence vectors and pick the label with highest confidence: $y_i = \text{argmax}_{c \in C} \sum_{j \in \text{KNN}(i)} \omega_{jc}$.

Support Vector Machine: For including weights in SVM we follow [19], which introduced weighted SVM. The key idea is that we want SVM to classify point $\mathbf{v}_i$ that has high confidence label (i.e., $|w_{iC_1} - w_{iC_2}|$ is near 1) correctly, but for points with low confidence, SVM can prefer margin maximization to correct classification. Thus, for each data point we have different coefficient for slack variable that is proportional to the confidence of the data label. So the primal will change to

$$\mathbf{W}^* = \underset{\mathbf{W}}{\text{argmin}}\ \tfrac{1}{2} ||\mathbf{W}||^2 + B \sum_{i=1}^{n} |\omega_{iC_1} - \omega_{iC_2}| \xi_i,$$
$$\text{s.t. } y_i(\langle \mathbf{W}, \phi(\mathbf{v}_i) \rangle) + b) \geq 1 - \xi_i,\ \xi_i \geq 0, \quad i = 1, ..., n.$$

2.4.3 Testing Procedure

When the label set is binary, testing data $X_{\text{test}} = \{\mathbf{x}_i \mid i = 1, \ldots, n_{\text{test}}\}$ is accompanied by a label set $\tilde{Y}_{\text{test}} = \{y_i = \text{argmax}_{c \in C} \omega_{ic} \mid i = 1, \ldots, n_{\text{test}}\}$, and the per-sample loss function is$1_{[f(\mathbf{x}_i) \neq y_i]}$, where f is the mapping of input to output produced by classification algorithm and $1_{[\bullet]}$ is the indicator function. Therefore, the classification error is defined as $\frac{\sum_{i=1}^{n} 1_{[f(\mathbf{x}_i) \neq y_i]}}{n}$.

In our weighted framework, the testing label set takes the form $Y_{\text{test}} = \{\omega_{ic} \mid i = 1, \ldots, n_{\text{test}}; c = 1, \ldots, |C|\}$. As mentioned in Section 2.1, at each step of cascade classifier we perform 2-class classification to separate two disjoint subsets of C namely C_1 and C_2. So the weighted per-tweet loss would be $\omega_{iC_l} \cdot 1_{[f(\mathbf{x}_i) \neq C_l]}$, where l is computed as $l = \text{argmax}_j\, \omega_{iC_j}$, $j = 1, 2$. The higher the weight of a datum is, the more it costs to miss-classify it. Accordingly, we should redefine the error as the total loss over all data, normalized by total possible loss: $\frac{\sum_{i=1}^{n} \omega_{ic_l} \cdot 1_{[f(\mathbf{x}_i) \neq c_l]}}{\sum_{i=1}^{n} \omega_{ic_l}}$. Note that the prior for weighted methods should also be computed accordingly.

3. EXPERIMENTAL VALIDATION

For the main part of the experiments, we used three collections of tweets. Two of them (DB1, DB2) are about weather and one is about gas price (GP), and they contains 4490, 8850, and 12770 tweets respectively.

Although in a natural scenario the first task is target extraction, because sentiment classification is at the center of attention in the literature, we start from it and subsequently

Table 1: Accuracies for the sentiment classification for DB2 with different versions of the input vector. Prior is 64.44%.

Binary?	Compressed?	SVM	NB	KNN	DL
True	True	79.19	75.04	74.50	78.94
True	False	80.16	82.95	75.01	-
False	True	77.67	71.49	73.60	77.02
False	False	76.86	81.33	74.17	-

Table 2: Accuracies of sentiment classification (binary loss).

	DB1	DB2	GP
DL	78.72 ± 2.52	78.94 ± 0.96	86.46 ± 1.15
SVM	78.99 ± 2.65	79.19 ± 1.79	**87.34 ± 1.46**
KNN	75.20 ± 2.97	75.01 ± 2.30	86.88 ± 1.28
NB	**82.23 ± 3.24**	**82.95 ± 2.10**	87.29 ± 1.25
WAH	59.55	75.01	19.43
LIWC	59.23	62.40	30.40
G-API	39.97	45.13	12.38
Prior	51.72%	64.44%	83.29%

discuss the other tasks in reverse order of Figure 1. We also use sentiment classification to explain our parameters and their assigned values. All reported results are obtained using 10-fold cross validation.

3.1 Sentiment Classification

In order to test the performance of the algorithms for sentiment classification (third classification task of Figure 1), we only consider tweets which have an associated positive or negative sentiment. For unweighted experiments we consider only dominant labels and for weighted experiments we use the aggregated weights ω_{iC_1} and ω_{iC_2}, where here C_1 and C_2 are representing negative and positive classes respectively.

For each algorithm, different parameter settings have been verified and results for best configurations are reported. Multinomial Naïve Bayes (MNB) outperformed other variants of NB in the original feature domain. K for KNN is set to 10. Linear kernel SVM performed better than other kernels. The main parameter in DL and WDL is the number of atoms in the dictionary. Our experiments showed that under-complete dictionaries (i.e., tall matrices) yield higher accuracy. We introduced ratio parameter to control the proportion of number of atoms to length of atoms. For all experiments we set ratio to 0.5. For detailed explanation of parameters setting refer to our technical report [2].

We consider two ways for modifying the input vectors. First, we can use their support (i.e., binary version) of the original word-count vector. Notice that tweets are themselves near binary. Second, each word-count or binary vectors can be projected to a compressed domain using random projection. The result of each setting is presented for the sentiment classification in Table 1 just for DB2.

As it is clear from Table 1 classification in low dimensional space without major loss of classification accuracy (in comparison with classification in $\mathbf{v}$ space) is possible. Recent results, [8], show theoretically that learning can be done in the compressed domain without significant loss in classification accuracy for support vector machine.

Based on the results of this step, we picked for each method the setting for input vectors which yields the best accuracy. NB and KNN best results are achieved with uncompressed binary vectors. Performance of both SVM and DL were increased using compressed binary vectors. Table 2 shows these results for all three databases.

We also compare our results with lexicon-based methods. Google has recently provided an API (G-API) that classifies a tweet, as either neutral, positive, or negative (`twittersentiment.appspot.com`) [10]. Linguistic Inquiry and Word Count (LIWC) is a text analysis software that was recently used by [11] for sentiment analysis of collections of Twitter data (`www.liwc.net`).

Another lexicon based method (WAH) has been recently proposed by [9]. Following an extensive study of words' sentiment in [9], Dodds et al. generated a list of words with happiness score from 1 to 10. After eliminating neutral words (i.e., with score around 5), they compute the weighted average happiness (WAH) of a batch of tweets.

G-API and LIWC perform 3-class classification (i.e., positive vs. negative vs. neutral) and since their source code is not available we could not modify it for 2-class classification. Therefore we feed them only with positive and negative tweets and report the results. On the other hand WAH is working only for sentiment classification step. As expected, lexicon based algorithms are not suited for per-tweet tasks (Table 2).

Table 3 presents the results of the weighted variants for the weighted loss functions introduced in Section 2.4.3. Note that weighted priors of Table 3 is different from unweighted prior of Table 2. Here again WNB has the highest accuracy in almost all databases, but its margin with WSVM and WDL is reduced in comparison with the unweighted case.

Table 3: Accuracies of sentiment classification (weighted loss).

	DB1	DB2	GP
WDL	**81.12 ± 2.97**	81.43 ± 1.82	86.50 ± 1.02
WSVM	78.84 ± 3.77	82.13 ± 1.58	87.53 ± 1.18
WKNN	76.34 ± 3.80	78.92 ± 1.76	86.32 ± 1.62
WNB	80.35 ± 2.93	**83.28 ± 2.34**	**88.01 ± 1.28**
Prior	73.40%	56.99%	83.28%

Table 4: Accuracies of sentiment detection (binary loss).

	DB1	DB2	GP
DL	80.29 ± 2.56	82.19 ± 1.65	74.00 ± 1.25
SVM	77.53 ± 2.35	79.80 ± 1.39	73.94 ± 1.50
KNN	74.26 ± 1.94	78.49 ± 1.88	70.47 ± 1.38
NB	**80.77 ± 2.00**	**82.53 ± 1.49**	**74.77 ± 1.15**
LIWC	54.83	57.84	39.50
G-API	57.33	57.511	50.16
Prior	59.95%	58.22%	50.06%

3.2 Sentiment and Target Extraction

Different algorithms have been recently applied to sentiment extraction (second classification task of Figure 1), such as NB [15] and SVM [3]. All these approaches use rich feature vectors, that incorporate higher-level grammatical or semantical knowledge of some form. However we show that we can separate polar and neutral tweets with high accuracy by using simple bag-of-words input vector.

As in the previous section, we assume an oracle that discards tweets that are not related to the topic of interest before sentiment detection step. We therefore use only tweets for which the dominant label is positive, negative, or neutral. Results are summarized in Table 4 and Table 5. In both cases NB (WNB) performance is the best.

We now turn to the target extraction in which we detect tweets belonging to the given topic of interest (first classification task of Figure 1). Since the required label for the GP database is not available, we only report results for DB1 and DB2. Tables 6 and 7 show the results of target extraction for unweighted and weighted algorithms respectively. Again NB (WNB) and DL (WDL) are the best performing methods.

Table 5: Accuracies of sentiment detection (weighted loss).

	DB1	DB2	GP
WDL	84.29 ± 2.66	85.50 ± 1.31	74.37 ± 1.38
WSVM	81.92 ± 2.86	84.45 ± 1.20	73.43 ± 0.95
WKNN	80.49 ± 2.78	82.89 ± 1.14	**70.44 ± 1.46**
WNB	**84.58 ± 2.04**	**86.04 ± 1.46**	74.14 ± 1.59
Prior	59.10%	61.96%	50.06%

Table 6: Accuracies of target extraction (binary loss).

	DB1	DB2
DL	80.85 ± 2.12	81.15 ± 1.15
SVM	80.00 ± 1.04	78.73 ± 1.58
KNN	77.04 ± 2.03	75.51 ± 2.51
NB	**82.64 ± 1.93**	**81.93 ± 1.43**
Prior	72.24%	72.06%

3.3 Spatially Aggregated Results

As mentioned in Section 1 sentiment analysis sometimes is being done on batches of tweets instead of single tweet [5, 6, 9, 11]. One common way of making batches is aggregating tweets based on the geographic location of the authors (e.g., state or county). We supplement our experiments with the spatially agglomerated results of the sentiment classification.

Figure 2 shows the results aggregated per state for the GP database using WDL and the ground truth map. Maps and the additional statistics provided in Table 8 show that the state mood is correctly recovered by WDL. Note that WDL outperforms LIWC (an off-the-shelf software for batch tweet processing) by substantial margin.

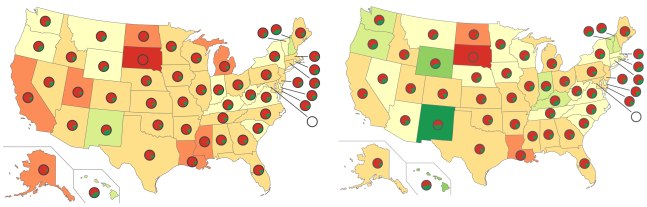
4. CONCLUSION

In this paper we presented a cascaded classifier framework for tweet sentiment analysis. We formulated the sentiment analysis as a cascade classifier with three sequential 2-class classification steps. In the first step, we separate tweets that are about the topic of interest, and then filter out tweets that do not contain any emotion. Finally we set apart positive from negative tweets.

Many previous work have tried to enrich the per-tweet information in various ways. Some added language level features [1, 3, 15] to improve input signal. The other used batches of tweets [5, 6, 9, 11] to compensate scarcity of per-tweet information. We showed that even with simple bag-of-words feature vector high accuracy is achievable for per-tweet classification tasks. We also showed that projecting the sparse feature vector into a lower-dimensional space is computationally beneficial and does not significantly affect the classification accuracy. Considering performance of classification in compressed domain, tailoring methods to work in that domain is a possible direction of future works.

5. ACKNOWLEDGEMENTS

The research was supported by NSF grants IIS-0812183, IIS-0916750, IIS-1029711, and NSF CAREER award IIS-0953274. GS and MT acknowledge partial support from DARPA, ONR, NSSEFF, ARO, NGA, and NSF. We thank Kent Cavender-Bares and dialogueearth team for providing the databases and Karl Otness for the preprocessing code.



(a) Ground truth map. (b) Aggregated WDL result.

Figure 2: Comparison of WDL with ground truth map. Red and green are negative and positive sentiments respectively.

Table 7: Accuracies of target extraction (binary loss).

	DB1	DB2
WDL	83.07 ± 2.29	**82.85 ± 1.14**
WSVM	82.55 ± 2.04	81.18 ± 1.69
WKNN	79.28 ± 2.39	73.32 ± 1.50
WNB	**83.93 ± 2.08**	82.59 ± 1.03
Prior	74.75%	74.85%

Table 8: Statistics of the per state error in %.

	mean	std	min	max
WDL	**5.21**	**3.71**	**0.00**	**13.76**
LIWC	38.00	10.55	3.48	67.24

6. REFERENCES

[1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. Sentiment analysis of twitter data. In *LSM*, pages 30–38, 2011.

[2] A. Asiaee Taheri, M. Tepper, A. Banerjee, and G. Sapiro. If you are happy and know it ... tweet. Technical Report 12-017, University of Minnesota, 2012.

[3] L. Barbosa and J. Feng. Robust sentiment detection on twitter from biased and noisy data. In *COLING*, pages 36–44, 2010.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2nd edition, 2007.

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

[6] J. Bollen, A. Pepe, and H. Mao. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. In *ICWSM*, 2011.

[7] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[8] R. Calderbank, S. Jafarpour, and R. Schapire. Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain. Technical report, Rice University, 2009.

[9] P. Dodds, K. Harris, I. Kloumann, C. Bliss, and C. Danforth. Temporal patterns of happiness and information in a global social network: hedonometrics and Twitter. *PLoS ONE*, 6(12):e26752, 2011.

[10] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. Technical report, Stanford University, 2009.

[11] S. Golder and M. Macy. Diurnal and seasonal mood vary with work, sleep, and daylength across diverse cultures. *Science*, 333(6051):1878–1881, 2011.

[12] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*. 2009.

[13] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao. Target-dependent twitter sentiment classification. In *ACL HLT*, volume 1, pages 151–160, 2011.

[14] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2):103–134, 2000.

[15] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, 2010.

[16] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, 2008.

[17] I. Ramirez, P. Sprechmann, and G. Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *CVPR*, pages 3501–3508, 2010.

[18] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120, 2009.

[19] X. Yang, Q. Song, and A. Cao. Weighted support vector machine for data classification. In *IJCNN*, volume 2, pages 859– 864, 2005.