If You are Happy and Know It ... Tweet

Amir Asiaee Taheri, Mariano Tepper, Arindam Banerjee, and
Guillermo Sapiro

June 18, 2012

If You are Happy and Know It ... Tweet

Amir Asiaee Taheri
Department of Computer Science and Engineering
University of Minnesota
MN 55455 USA
ataheri@cs.umn.edu

Mariano Tepper
Department of Electrical and Computer Engineering
University of Minnesota
MN 55455 USA
mhtepper@umn.edu

Arindam Banerjee
Department of Computer Science and Engineering
University of Minnesota
MN 55455 USA
banerjee@cs.umn.edu

Guillermo Sapiro
Department of Electrical and Computer Engineering
University of Minnesota
MN 55455 USA
guille@umn.edu

ABSTRACT

Extracting sentiment from Twitter data is one of the fundamental problems in Social Media Analytics. The length constraint of Twitter, an average of about six words per message, renders determining the positive or negative sense of a tweet difficult even for a human judge. In this work we present a general framework for single tweet (in contrast with batches of tweets) sentiment analysis which consists of three steps: extracting tweets about a desired target, separating tweets with sentiment, and discriminating positive and negative tweets. We study the performance of a number of classical and new machine learning algorithms for classification in each step. We also show that the intrinsic sparsity of tweets enables us to perform classification with their low dimensional random projections without losing accuracy. In addition, we present weighted variants of all employed algorithms which exploit available labeling uncertainty in order to further improve classification accuracy. Finally, we show that aggregating per-tweet sentiment analysis results, improve the accuracies and make 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

Social media has become an important part in the everyday life of millions of people around the world. Data is being produced at tremendous rates providing many scientific disciplines with a wealth of data to analyze, with a wide variety of applications. Recent studies, e.g., [7, 8, 24, 29], have shown the predictive value of social media content in domains such as marketing, business, and politics.

Twitter, a micro-blogging website, is among the most pervasive of these social media. 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.

Twitter data presents two fundamental and almost contradictory properties: it is at the same time overabundant at the global scale but scarce at the individual level. On the one hand, because of its widespread use and streaming nature, it can be considered as big data with all the computational constraints that this implies. On the other hand, the 140 characters limit (a maximum of about fifty words but only six on average, according to our experiments), severely restricts the amount of information per tweet, rendering per-tweet analysis, the main goal of this paper, very challenging. The information is so scarce that sometimes even a human cannot analyze tweets accurately without prior information such as knowledge about their author. Including context knowledge, like geographic location and age may lead to different interpretations of a tweet.

Several data mining tasks can be defined for Twitter data with diverse applications. Among them, sentiment analysis [22, 26] has increasingly gained attention from both academia and industry. Sentiment analysis tries to identify the positive or negative sentiment of a text. Using sentiment analysis, opinion polls can be done in a natural and *non-intrusive* way by monitoring tweets about a given topic and analyzing the sentiment content [7, 29]. Detecting major events based on tweets' sentiments [1, 7, 8], and finding the pattern of temporal happiness and mood in human behavior [12, 16] are examples of other explored applications of sentiment analysis for Twitter data.

Although Twitter sentiment analysis increasingly gained attention, there are several issues that limit its usage in practical applications. In general tweets do not always contain 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. Knowing this fact, tweet anal-

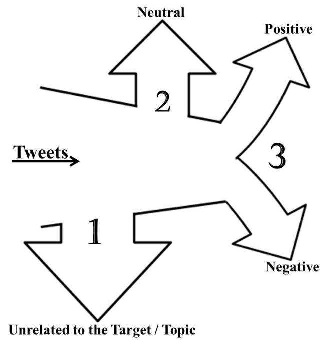


Figure 1: Three steps for Twitter sentiment analysis.

ysis literature have moved from just sentiment analysis to also considering neutral tweets in classification [1, 21].

Ignoring the objects, individuals, or products that the tweet is expressing emotion about them is another 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 [19]. There has been initial work on target-dependent sentiment analysis, [19], which exploits history of users' tweets to do the sentiment analysis and does not work with a single tweet.

We delineate three fundamental steps required for sentiment analysis, Figures 1:

1. **Target extraction:** detecting tweets related to the target of interest.
2. **Sentiment detection:** separating tweets that have feelings.
3. **Sentiment classification:** distinguishing sentiment types (i.e., positive or negative).

The first step toward realistic sentiment analysis is target extraction, by which we mean distinguishing between the tweets that are related to our topic of interest from the unrelated ones [19]. This 2-class classification problem has strong relationship with topic modeling (unsupervised extraction of meaningful topic) and classification (supervised labeling based on topic information). While topic modeling/classification are mature techniques when applied to usual texts [6, 28], and recent works have addressed topic modeling for Twitter data [17, 20, 34], classification of topics for micro-blogs has not been fully explored yet.

The second step, sentiment detection, is carried out with the goal of determining which tweets have emotional content. This is sometimes referred to as the polar-neutral classification problem [1, 2, 21, 25].

Finally, sentiment classification is performed only on tweets with emotional content. Many emotional dimensions can be extracted from rich text data such as weblogs, but because of the meager information contained in single tweets, they are usually classified into two main groups, according to their emotional energy: negative emotions (e.g., fear, hatred, resentment, anger, hostility), and positive emotions (e.g., enthusiasm, laughter, empathy, happiness).

The other popular method, which attempts to circumvent the scarcity of per-tweet information, analyze batches of tweets. These batches can be built using different criteria such as spatial (geographical location of senders), temporal (time of postings), 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 [7, 8, 12, 16]. As expected, these approaches perform poorly on a per-tweet basis as later shown in our experiments. Overall, standard text analysis techniques are not suited to work with the limited data available in single tweets. Moreover, batch analysis methods are not suited for cases in which the grouping criteria are not trivial to obtain (e.g., when grouping by age, or gender), or are unknown (then the goal might actually be to find those groups).

To the best of our knowledge, this work is the first attempt that addresses all of the aforementioned classification tasks together for sentiment analysis of *single* tweets. Specially sentiment analysis for specific target, is a major step that is missing in related works [19]. All steps are formulated as 2-class classification problems, and several supervised learning methods are tested and developed. Unlike many previous works that use sophisticated language features [1, 2, 21], with heavy pre-processing, we show that high accuracy in each step is achievable by just using bag-of-words as the input for classification. In addition, we show that classification can be performed on random reconstructible projections of high dimensional sparse input data without losing performance accuracy. Finally, since our data has crowdsourced soft labels (i.e., multiple labels for each tweet provided by a group of evaluators), we present a weighted variant of all presented classification methods.

In addition, we supplement our work with the results of sentiment classification for spatially aggregated tweets. We show that performing single tweet sentiment analysis followed by aggregation results in high accuracy. We also provide a new method for polar-neutral identification exploiting results from experimental psychology [16].

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 some concluding remarks.

2. THE CLASSIFICATION PIPELINE

In this section we discuss the whole classification process. We begin by explaining how the data is labeled for training. Then we comment on the parsing and preprocessing procedures which output tweets represented by high-dimensional feature vectors using a bag-of-words approach. Following the literature [3, 33], the support of the bag-of-words vector is used as input feature vector. The high dimension and sparsity of such input vectors enable us to use random reconstructible projection in order to reduce its dimensionality [10, 18].

2.1 Labeling the Data

Supervised learning algorithms obviously rely on the availability of labeled data. The use of specific words to label tweets is common. For example, Pak and Paroubek [25] use an emoticon list as indicator for positive or negative content, and Bollen et al. [7] use the phrase "I feel" to label tweets as polar (i.e., expressing emotions). Other works, e.g., [2, 15], gather noisy labels from multiple sources, like unevaluated sentiment analysis tools, and then incorporate this uncertainty into the classification algorithm.

In recent years, crowdsourcing has emerged as a cost-effective way to carry out labor-intensive tasks, thus becoming popular in the machine learning community [30]. In this work, the process of data labeling is crowdsourced as a part of the Dialogue Earth Project,[1] which kindly provided the data for the experiments of this report.

Since we are potentially facing a dataset which is growing with the rate of 200 millions data points a day, it would be wise to first prune it using simple techniques. The first task in the hierarchy of the aimed tasks is target extraction. Thus, we first perform gross filtering on a collection of tweets based on an extensive list of words associated to the target/topic of interest and filter out tweets that do not contain any of the indicator words. For example, if the target topic is weather, by using a list of words that relates to weather (like snow, cold, hot, etc.), we can separate relevant tweets. Although crude filtering eliminates many unrelated tweets, there are still many irrelevant tweets in our dataset which make the topic classification task a necessity. We may also miss some topic-related tweets that do not contain any of our compiled words, which is inevitable because of the dataset size. In this article, we use databases collected for weather and gas price topics.

The data is then hand labeled by several trustworthy evaluators (i.e., people that consistently showed good accuracy during quality control tests), with 4 labels: positive, negative, neutral, and not related to the target topic. An additional label is reserved for cases in which the evaluator cannot assign a tweet to any of the aforementioned classes. It must be noted that the quality control tests ensure that the labels are not too noisy, and when an evaluator cannot label a tweet, it can be interpreted as if there is no context-independent information in it. It should be mentioned that having trustworthy evaluators makes our data source different from other crowdsourced data whose annotation quality should be evaluated itself [30]. Thus, disagreement between evaluators shows the inherent difficulty of the task at hand.

Let C be the set of all classes. For each tweet i, evaluator j chooses a label $\mathbf{l}_{ij}$, 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 $\boldsymbol{\omega}_i = (\sum_{j=1}^{m} \mathbf{l}_{ij})/m$ where m is the number of evaluators.

Having the confidence vector $\boldsymbol{\omega}_i$ in hand, 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. On the other hand 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 configuration in which each data point has a single label, here denoted as *dominant* labels. Since we have worked with both soft label Y and hard label $\tilde{Y}$ when we discuss weighted algorithms, we refer to those that use the soft label set Y.

Finally, since in each of the three steps of sentiment analysis introduced in Section 1 we perform 2-class classification on two subsets of C, (e.g., related vs. not related, which contains neutral, positive and negative), we should compute the weights of these subsets. Assume that we want to classify $C_1, C_2 \subset C$ where $C_1 \cap C_2 = \emptyset$ and they do not necessarily partition C. Then for each tweet i we should compute two weights ω_{iC_1} and ω_{iC_2}, which are simply the sum of the labels' weights that are present in C_1 and C_2: $\omega_{iC_j} = \sum_{c_k\in C_j} \omega_{ik}, j = 1, 2$.

2.2 Parsing and Preprocessing

In this work, we use the bag-of-words model for representing the tweets. One of the main advantages of this approach is that the parsing procedure is very simple. Many previous works have included language level features, like part of speech tags as input [25, 2, 1, 21], making the pre-processing steps more complicated.

We begin by extracting the words in a tweet. Here we use the term word in a broad sense, since it encompasses actual words and also numbers, usernames, emoticons, URLs, etc. Some of these however receive special treatment. We do not care about the actual value/content of numbers, usernames, and URLs, and thus replace them by special generic identifiers. Notice that simply removing these words would be harmful, since these type of words are common in neutral tweets which just share information. We do remove re-tweet signs (RT), special characters (not contained in emoticons), and stop words (e.g., 'the,' 'of,' 'about'). Note that we have removed all polar words (e.g., 'great,' 'bad,' 'better') from the stop word list to prevent loss of emotional signals. Also, all negation words (e.g., 'not,' 'never') have been removed from the stop word list, since they can change the sentiment completely [25]. Hashtags are a special kind of keywords used in Twitter that are often a concatenation of words (e.g., '#rainymorning,' '#loveThisWeather'); when the words in a hashtag begin with an uppercase character, we break it into separate words.

We automatically spell check the words using three dictionaries: an English dictionary, a Twitter dictionary which contains specific lingo, and an emoticon dictionary. The Twitter dictionary has been gathered from several online Twitter dictionaries that list popular words coined by Twitter users. After spell checking we remove words that are not in any of the mentioned dictionaries. We found that stemming did not improve the classification results and thus we omit it from the preprocessing procedure.

After cleaning raw tweets using all of above steps, 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 based on the desired task, highly frequent non-distinctive words are being removed. A word is considered highly frequent in a class if it appears on average more than a "high frequency threshold" in each tweet of that class. Among high frequent words of the two classes, those with average frequency closer than a "similarity threshold" are called non-distinctive.

2.3 Representing and Compressing Tweets

The bag-of-words model is one of the most commonly employed feature extraction approaches in text (and image) classification. A text (document) is simply represented as an unordered collection (i.e., a set that may contain repeated elements) of words W. A predefined set of words $L = \{l_i \mid i = 1,\ldots,d\}$ is then used to build a d-dimensional feature vector $\mathbf{v}$ for each document W such that $(\forall i = 1,\ldots,d)$ $\mathbf{v}(i) = \#(W, l_i)$, where $\#(W, l_i)$ is the number of times word l_i appears in W.

Usually d is counted in the tens of thousands (e.g., in our

[1]www.dialogueearth.org

experiments $d \approx 10^4$), but since the length of a tweet is limited to 140 characters, the bag-of-words approach produces extremely sparse feature vectors.

Although we have done experiments using the original high-dimensional bag-of-words data, we extend our experiments and also use low dimensional projection of it as input. Working in the original high-dimensional domain imposes computational difficulties which naturally lead to dimensionality reduction techniques. Here we show that working with (extremely) sparse d-dimensional feature vectors directly is not necessary, one can instead reduce their dimensionality using random reconstructible projections and perform classification in the resulted domain without considerable loss of accuracy. Since random reconstructible projection is a well-known technique in the compressed sensing literature [9], we use compression and projection terms interchangeably. Recent results [10], show theoretically that learning can be done in the compressed domain without significant loss in classification accuracy for support vector machine. In this paper we show empirically that compressed learning (i.e., learning in compressed domain) is also possible for other well-known classification algorithms.

In this framework an $m \times d$ matrix $\mathbf{P}$ ($m \ll d$) is used 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 $\mathbf{x}$. The best reconstruction performance is obtained when $\mathbf{P}$ is a random matrix, i.e., when its entries p_{ij} are sampled from i.i.d. random variables [18]. In this paper 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 $d \approx 10^4$, $h \approx 20$). We have tried other types of methods for generating $\mathbf{P}$ [18] in preliminary experiments and based on performance chose to work with a Gaussian random projection.

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 $\mathbf{x}_i = \mathbf{P}_{m\times d}\, \mathbf{v}_i$. The goal is now to learn to classify the tweets based on this compressed representation.

It is worth mentioning that random projections (RP) have been used as a dimensionality reduction technique previously in the literature [4], and its classification performance has been compared with other dimensionality reduction methods like PCA [14]. The result of this comparison is that PCA outperforms RP but RP is computationally more efficient. In this work, we also used PCA and surprisingly it underperforms random reconstructible projections (RRP) in almost all classification methods. This is in accordance with the theoretical result of [10], that shows that when the original input vector is sparse, classification with high accuracy in the compressed domain constructed by RRP is possible.

2.4 Classification Methods

Several well-known supervised learning algorithms and their weighted variant have been selected and tested. Support Vector Machine(SVM) [11], K Nearest Neighbor(KNN) [5], and Naïve Bayes(NB) [5] are well known. Hence, we only explain the classification algorithm which is based on dictionary learning, [27], in detail, along with brief descriptions of weighted variants of the other methods.

2.4.1 *Sparse Modeling Approach to Classification*

In recent years, sparse modeling techniques gained popularity among the signal processing and machine learning communities for their ability to provide efficient representations for a great variety of signals such as audio and natural images. This efficiency is achieved by approximating a signal 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 [9].

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 optimization

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

which is convex with respect to the variables $\boldsymbol{\alpha}_i$ when $\mathbf{D}$ is fixed and viceversa (here λ a positive constant). The optimization is then commonly solved by alternatively fixing one and minimizing over the other.

Sparse modeling has been previously employed for supervised classification tasks, exhibiting state-of-the-art performance in visual and audio applications such as face recognition and the PASCAL challenge [27]. 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

$$f(\mathbf{x}) = \operatorname*{argmin}_{c\in C}\ \ell(\mathbf{x}, \mathbf{D}_c),$$

and $\ell(\mathbf{x}, \mathbf{D}_c) = \min_{\alpha} \frac{1}{2}\|\mathbf{x} - \mathbf{D}_c\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1$.

For each step of tweet classification $|C| = 2$ and Y is the set of binary values that represent the dominant label of each data point. For exploiting the possible available information in the non-binary confidence ω_{ic} introduced in 2.1, we propose to redefine the cost function for each datum $\mathbf{x}_i$ and each class dictionary $\mathbf{D}_c$ as

$$\ell_\omega(\mathbf{x_i}, \mathbf{D}_c) = \min_{\boldsymbol{\alpha}}\ \omega_{ic}\left[\frac{1}{2}\|\mathbf{x} - \mathbf{D}_c\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1\right],$$

thereby using this cost instead of ℓ when learning each class dictionary. The contribution of the sample $\mathbf{x}_i$ to the class c is then weighted by its non-binary label $\omega_c(\mathbf{x}) = \omega_{ic}$. The closer ω_{ic} is to one, the more $\mathbf{x}_i$ contributes to class c. In the extreme case that $\omega_{ic} = 0$, $\mathbf{x}_i$ does not contribute at all to the learning of the dictionary for class c.

We then solve the optimization problem

$$\min_{\mathbf{D}} \frac{1}{n}\sum_{i=1}^{n} \ell_\omega(\mathbf{x}_i, \mathbf{D}) \tag{2}$$

for each class c, by alternating the minimization over $\mathbf{D}$ and the sparse codes $\boldsymbol{\alpha}_i$.

Sparse coding: Minimizing Equation (2) over the sparse codes $\boldsymbol{\alpha}_i$ with $\mathbf{D}$ fixed involves solving for each $i = 1, \ldots, n$,

$$\min_{\boldsymbol{\alpha}_i}\ \omega_{ic}\left[\frac{1}{2}\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1\right].$$

Since for each subproblem ω_{ic} is constant, this is a classical LASSO problem and there is no need for designing particular minimization techniques and methods like LARS [13] can be used directly.

Dictionary learning: Because of its streaming nature and widespread use, Twitter data can be considered as a massive data source. Therefore, online learning algorithms arise as an obvious choice for analyzing them. Following the online dictionary learning approach of [23], for minimizing Equation (2) over $\mathbf{D}$ (one such $\mathbf{D}_c$ per class c), with the sparse codes $\boldsymbol{\alpha}_i$ fixed for all i, we rewrite it as

$$\min_{\mathbf{D}} \frac{1}{n}\left(\frac{1}{2}\mathrm{Tr}(\mathbf{D}^T\mathbf{D}\mathbf{A}) - \mathrm{Tr}(\mathbf{D}^T\mathbf{B})\right) \tag{3}$$

where $\mathbf{A} = \sum_{i=1}^n \omega_i\ \boldsymbol{\alpha}_i\boldsymbol{\alpha}_i^T$ and $\mathbf{B} = \sum_{i=1}^n \omega_i\ \mathbf{x}_i\boldsymbol{\alpha}_i^T$ (since in this case for simplicity we dropped the subindex c from $\mathbf{D}$, for consistency we write ω_i when meaning ω_{ic}). [23] have shown that there is a closed form for updating each column of $\mathbf{D}$, and this also follows when we add the weights as in Equation (3).

The complete optimization scheme for this online weighted dictionary learning algorithm is depicted in Algorithm 1 (recall that we learn one dictionary per class). The implementation is obtained by adding the weights to the publicly available SPAMS library.[2]

We wrap up the dictionary learning based classification with a discussion about dictionary learning in the original (uncompressed) domain. Dictionary learning in the original domain would try to minimize $\ell(\mathbf{v}_i, \mathbf{D}) = \frac{1}{2}\|\mathbf{v}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1$ for each feature vector $\mathbf{v}_i$. But it is well known that for the l_2 penalty we intrinsically assume that the error $\mathbf{v}_i - \mathbf{D}\boldsymbol{\alpha}_i$ is Gaussian noise [31], and since text documents do not usually satisfy this assumption, in this article we only perform dictionary learning in compressed domain. Recently, Kasiviswanathan et al. [20] have presented an alternative formulation using an l_1 reconstruction error for novel topic detection in Twitter.

2.4.2 Weighted Variant of Classification Algorithms

In this section we introduce weighted variants of well-known classification algorithms which exploit the available soft labels to improve classification. We cover Naïve Bayes(NB) [5], K Nearest Neighbor(KNN) [5], and Support Vector Machine(SVM) [11].

Naïve Bayes: A naïve Bayes classifier, in its general form, assigns to each test data point the maximum a posteriori class

$$y_i = \operatorname*{argmax}_{c\in C} P(c|\mathbf{v}_i).$$

Using Bayes' rule $P(c|\mathbf{v}_i) = P(c)P(\mathbf{v}_i|c)/P(\mathbf{v}_i)$, and following the fact that $P(\mathbf{v}_i)$ is constant for all classes, we have

$$y_i = \operatorname*{argmax}_{c\in C} P(c)P(\mathbf{v}_i|c),$$

which is simplified assuming the conditional independence of the input vector's features,

$$y_i = \operatorname*{argmax}_{c\in C} P(c)\prod_{j=1}^{d} P(v_{ij}|c), \tag{5}$$

where $P(c)$ and $P(v_{ij}|c)$s are computed from their corresponding frequencies in training data.

[2]http://www.di.ens.fr/willow/SPAMS/

Algorithm 1 Weighted Online Dictionary Learning [23]

```
input a random variable $\mathbf{x} \in \mathbb{R}^m$ with p.d.f. $p(\mathbf{x})$ (the training data), a weighting function $\omega : \mathbb{R}^m \to [0,1]$, a regularization parameter $\lambda \in \mathbb{R}$, an initial dictionary $\mathbf{D}_0 \in \mathbb{R}^{m\times k}$, the number of iterations $T$.
output the dictionary $D_T$.
1: $\mathbf{A}_0 \in \mathbb{R}^{k\times k} \leftarrow 0$ , $\mathbf{B}_0 \in \mathbb{R}^{m\times k} \leftarrow 0$
2: for $t = 1$ to $T$ do
3:     Draw $\mathbf{x}_t$ from $p(\mathbf{x})$
4:     Sparse coding: compute (e.g., using LARS)
           $\alpha_t = \operatorname*{argmin}_{\alpha\in\mathbb{R}^k} \frac{1}{2}\|\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1.$
5:     $\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \omega(\mathbf{x}_t)\ \boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^T$
6:     $\mathbf{B}_t \leftarrow \mathbf{B}_{t-1} + \omega(\mathbf{x}_t)\ \mathbf{x}_t\boldsymbol{\alpha}_t^T$
7:     Compute $\mathbf{D}_t$ with $\mathbf{D}_{t-1}$ as warm restart, solving
           $\mathbf{D}_t = \operatorname*{argmin}_{\mathbf{D}} \frac{1}{t}\sum_{i=1}^{t}\omega(\mathbf{x}_t)\left[\frac{1}{2}\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1\right]$
               $= \operatorname*{argmin}_{\mathbf{D}} \frac{1}{t}\left(\frac{1}{2}\mathrm{Tr}(\mathbf{D}^T\mathbf{D}\mathbf{A}_t) - \mathrm{Tr}(\mathbf{D}^T\mathbf{B}_t)\right).$   (4)
8: end for
```

Now we should incorporate the confidence weights in the Naïve Bayes formulation. So instead of computing $P(c)$ using class frequency, we use weighted class frequency

$$P(c) = \frac{\sum_i \omega_{ic}}{\sum_c\sum_i \omega_{ic}},$$

in which each data point $\mathbf{v}_i$ contributes to the class c's probability the amount that is proportional to its label confidence ω_{ic}. Also for each feature $P(v_{ij}|c)$, the probability should be computed based on weights of data:

$$P(v_{ij}|c) = \frac{\sum_i \omega_{ic}\times v_{ij}}{\sum_i \omega_{ic}}.$$

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:

$$y_i = \operatorname*{argmax}_{c\in C} \sum_{j\in \mathrm{KNN}(i)} \omega_{jc}$$

Support Vector Machine: Support vector machine (SVM) tries to find a separating hyperplane which maximizes the margin between two classes. In its original formulation, the following optimization should be solved

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

where $\mathbf{W}$ is the normal vector to the hyperplane and ϕ maps $\mathbf{v}_i$ to higher dimensional space. The constant B determines the trade-off between margin maximization and classification violation. For finding $\mathbf{W}^*$ one can maximize its dual,

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{n} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq B, \quad i = 1, ...n,$$

where $K(x_i, x_j) = (\langle \phi(x_i)\phi(x_j) \rangle)$ is the kernel.

For including weights in SVM we follow [32], which introduced weighted SVM (WSVM) to decrease the effect of outliers in SVM.

The key idea is that for a point $\mathbf{v}_i$ that we are sure about its label (i.e., $|w_{iC_1} - w_{iC_2}|$ is near 1), we should have larger penalty, which reinforces correct classification more that margin maximization. So the primal will change to

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{2} ||\mathbf{W}||^2 + B \sum_{i=1}^{n} |\omega_{iC_1} - \omega_{iC_2}| \xi_i$$

$$\text{s.t. } y_i(\langle \mathbf{W}, \phi(\mathbf{v}_i) \rangle) + b) \geq 1 - \xi_i, \ \xi_i \geq 0, \quad i = 1, ..., n.$$

And relatively the dual will change to:

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{s.t.} \sum_{i=1}^{n} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq |\omega_{iC_1} - \omega_{iC_2}| B, \quad i = 1, ...n.$$

As it is clear the only difference of WSVM with SVM is in the box constraint's upper bound for each lagrange multiplier.

2.4.3 Testing Procedure

Once labels have been assigned to the testing data following the classification procedure, a loss function is usually used to determine the accuracy of the assignment. When the label set is binary (i.e., each datum belongs to only one class), testing data $X_{\text{test}} = \{\mathbf{x}_i \mid i = 1, \ldots, n_{\text{test}}\}$ is accompanied by a label set $\tilde{Y}_{\text{test}} = \{y_i = \operatorname{argmax}_{c \in C} \omega_{ic} \mid i = 1, \ldots, n_{\text{test}}\}$, and the per-sample loss function is usually

$$\mathbb{1}_{[f(\mathbf{x}_i) \neq y_i]},$$

where f is the mapping of input to output produced by classification algorithm and $\mathbb{1}_{[\bullet]}$ is the indicator function. Therefore, the classification error is defined as

$$\frac{\sum_{i=1}^{n} \mathbb{1}_{[f(\mathbf{x}_i) \neq y_i]}}{n}.$$

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 needed for sentiment analysis we perform 2-class classification to classify two disjoint subsets of C namely C_1 and C_2. So the weighted per-tweet loss would be

$$\omega_{iC_l} \cdot \mathbb{1}_{[f(\mathbf{x}_i) \neq C_l]}, \tag{6}$$

instead of regular loss where l is computed as

$$\omega_{iC_j} = \sum_{c \in C_j} \omega_{ic}, \ l = \underset{j}{\operatorname{argmax}} \, \omega_{iC_j}, \ j = 1, 2.$$

Here we reiterated the definition of ω_{iC_j} from Section 2.1. In words, based on the task, we aggregate the weights to only two weights ω_{iC_1} and ω_{iC_2}. Then we consider the larger one as the label of the data point. The higher the weight of a datum is, the more it costs to classify it mistakenly. Accordingly, we should redefine the error as the total loss over all data, normalized by total possible loss:

$$\frac{\sum_{i=1}^{n} \omega_{iC_l} \cdot \mathbb{1}_{[f(\mathbf{x}_i) \neq c_l]}}{\sum_{i=1}^{n} \omega_{iC_l}}.$$

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 the last one is about gas price (GP), and they encompass 4490, 8850, and 12770 tweets respectively. We have used DB2 to adjust and validate our parameters. Then based on the best setting of parameters we perform experiments for all databases.

The databases are built by first collecting tweets with the Twitter API.[3] After crude filtering based on topic related word list, a few human evaluators, as explained in Section 2.1, were asked to assign to each tweet one of the following $C = 5$ classes:

- "Not related:" the tweet is not about the target topic;
- "Neutral:" the tweet contains no emotion;
- "Positive:" the tweet reflects positive feelings;
- "Negative:" the tweet reflects negative feelings;
- "I can't tell:" none of the above can be assessed.

Finally each tweet i receives a soft label, a weight ω_{ic} for each class $c = 1 \ldots 5$, equal to the proportion of evaluators that have chosen that class for the tweet. As commented in Section 2.1, notice that the "I can't tell" class simply means that the evaluator could not assign the tweet to any of the other classes, thus indicating the tweet actually has no label. We therefore discard from our analysis those tweets for which the "I can't tell" class gets the maximum weight, and we have $C = 4$ classes.

Next, preprocessing steps are performed as explained in Section 2.2. We picked 0.05 as high frequency threshold and select words that appeared on average more that 0.05 in each tweet as the candidates for removal. On average less than 30 words satisfy this condition in all databases. Then we remove those candidates that are close to each other. Here we consider words close if their frequency difference (i.e., similarity threshold defined in Section 2.2) is less than 0.2. We found that although numbers and links are highly frequent in many tasks, they are usually distinctive (i.e., not close) for tasks involving neutral tweets. One possible explanation is that the neutral tweets about weather and gas price usually share information that contains numbers and a link to the source of information.

We have a 4-class classification problem. One approach is to perform multi-class classification. But since these classes have a natural hierarchical structure, we can solve the classification problem with a cascade type of approach. The steps in the cascade are the following: (1) topic classification, filtering out "not related" tweets; (2) polar-neutral classification; and (3) sentiment classification. We will present results for all these steps.

[3]https://dev.twitter.com/

We present the results obtained with Dictionary Learning (DL), Support Vector Machines (SVM), K Nearest Neighbors (KNN), Naïve Bayes (NB), and their weighted variants. For DL and WDL, the number of atoms and λ were fixed for all experiments.

Although in a natural scenario, the first task is topic classification, because sentiment analysis is at the center of attention in the literature, we start from it and subsequently discuss the other tasks. We also use sentiment analysis as a platform to explain our parameters and their assigned values. All reported results are obtained using 10-fold cross validation.

3.1 Sentiment Classification

In this section we focus on sentiment analysis. In order to test the performance of the algorithms for this single task, 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}, all defined in Section 2.1, 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, while in the compressed domain using kernel density estimation was most helpful. K for KNN is set to 10, and since l_1 and l_2 distance metrics' performances were similar, we report only results of l_2 distance. Linear kernel SVM performed better than other kernels like RBF, quadratic, and polynomial. Also results of SVM in original and compressed domains were very close to each other, which is what we expected from the theoretical guarantee in [10]. 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 for dictionary to control the ratio of number of atoms to length of atoms (i.e, number of columns to the number of rows of a dictionary). For all experiments we set aspect ratio to 0.5. All weighted experiments are done with the same parameter setting as their unweighted variants.

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. Secondly, each word-count or binary vectors can be projected to a compressed domain using random projection. So we end-up with four different configurations for input vectors. The result of each setting is presented for the sentiment analysis task in Table 1 just for DB2. Based on the theoretical reasons explained in 2.4.1, we do not perform DL in the original domain. In addition, we compared RRP with PCA in our preliminary experiments with similar destination dimension. In all classification methods except SVM, RRP outperforms PCA. Since the computational cost of PCA for large feature vectors is high and increase in SVM's performance using PCA in comparison with RRP is negligible (less than 2%), we report only results of RRP dimensionality reduction.

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. Both SVM's performance and speed increased using compressed binary vectors. Also DL performs better when fed with binary vectors. Table 2 shows these results for all three databases. Note that the prior of GP is much higher than for the other databases, this is the reason why all methods perform better for the GP database. It is interesting to note that most tweets pertaining to gas price are negative.

Table 1: Classification accuracies for the sentiment classification for DB2 with different versions of the input vector. Prior of the experiment 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: Classification accuracies of unweighted algorithms for the sentiment classification with 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%

We also compare our results with a lexicon-based methods specifically designed for Twitter sentiment analysis. Google has recently provided an API[4] that classifies a tweet, represented as a list of words, as either neutral, positive, or negative [15]. Linguistic Inquiry and Word Count (LIWC)[5] is a text analysis software that was recently used by [16] for sentiment analysis of collections of Twitter data. Another lexicon based method has been recently proposed by [12]. Following an extensive (and crowdsourced) study of words' sentiment in [12], 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 by also taking into account word frequencies. All aforementioned batch methods have been tested for our databases. Google 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 with sentimental 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. In the case of the Google API, the inability to retrain the classifier may be the cause of its poor performance. As shown in Table 2, NB almost always outperforms other methods while KNN always has the worst performance.

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

[4]http://twittersentiment.appspot.com/
[5]http://www.liwc.net/

Table 3: Classification accuracies of weighted algorithms for the sentiment classification with 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: Classification accuracies of unweighted algorithms for the sentiment detection with 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%

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.

We also investigated the effects of projections. In order to mitigate possible "randomness"-related effects, we used ten different random projection matrices $\mathbf{P}$ and then merge the results (by using majority voting). Since the accuracy of each method appeared robust to the number of projection, i.e., less than 1% variation was observed, we use only one projection from here on.

3.2 Sentiment Detection and Target Extraction

Different algorithms have been recently applied to sentiment analysis and polar-neutral classification, such as NB [25], SVM [2] and AdaBoost.MH [21]. All these approaches use rich feature vectors, that incorporate higher-level grammatical or semantical knowledge of some form. However we show that high accuracy can be achieved even with a simple bag-of-words approach.

As in the previous section, we assume to have an oracle that discards tweets not related to the topic of interest. We therefore use only tweets for which the dominant label is positive, negative, or neutral. Results for unweighted algorithms are shown in Table 4, and results for weighted algorithms with none-binary loss function are presented in Table 5. In both cases NB (WNB) performance is the best and is followed closely by DL (WDL).

Experimental psychology studies show that positive and negative sentiments are not opposite extremes of the same dimension but are, on the contrary, independent [16]. Based on this point, we introduce a new method for polar-neutral classification. We train two independent classifiers for separating positive (negative) tweets from all the rest. Then we use a simple aggregation scheme for classifying neutral and polar tweets. If a tweet is classified as positive or negative by one of the mentioned classifiers, it will be polar, otherwise neutral. Results are shown in Table 6. Although the accuracies are not as high as direct polar-neutral classification, the results show a promising direction.

Table 5: Classification accuracies of weighted algorithms for the sentiment detection with 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: Classification accuracies of the sentiment classification using positive vs. background and negative vs. background classifiers.

	DB1	DB2	GP
DL	**79.95**	79.51	72.00
SVM	71.41	74.34	**73.67**
KNN	68.59	71.97	68.42
NB	78.56	**81.56**	72.78
Prior	59.95%	58.22%	50.06%

We now turn our attention to the detection of tweets belonging to a given topic of interest. Since the required label for the GP database is not available, we only report results for DB1 and DB2. Tables 7 and 8 show the results of topic classification for unweighted and weighted algorithms respectively. Again NB (WNB) and DL (WDL) are competing for the best performance.

3.3 Spatially Aggregated Results

We close the experimental section by showing spatially agglomerated results of the positive/negative classification. As mentioned in Section 1 sentiment analysis sometimes is being done on batches of tweets instead of single tweet. One common way of making batches is aggregating tweets based on the geographic location of the authors (e.g., state or county).

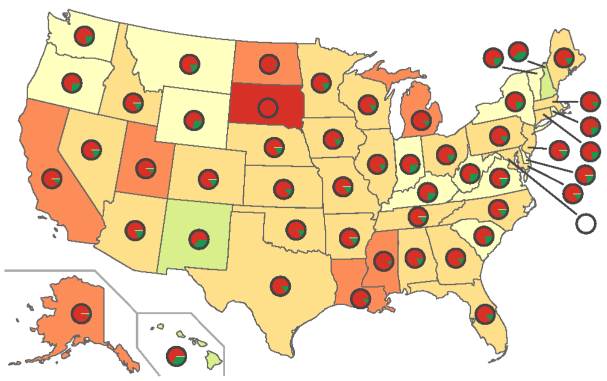
Despite the fact that our methods are not specifically designed for processing batches of tweets, batch results are easily obtained once the individual classification is done. For this experiment, we only use the database for which the topic is gas prices. Figure 2 shows the results aggregated per state for the GP database using WDL and the ground truth map. Both maps and the additional statistics provided in Table 9 show that the state mood is correctly recovered by the proposed classification procedure. 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 complete framework for tweet

Table 7: Classification accuracies of unweighted algorithms for target extraction with 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%


(a) Ground truth map.

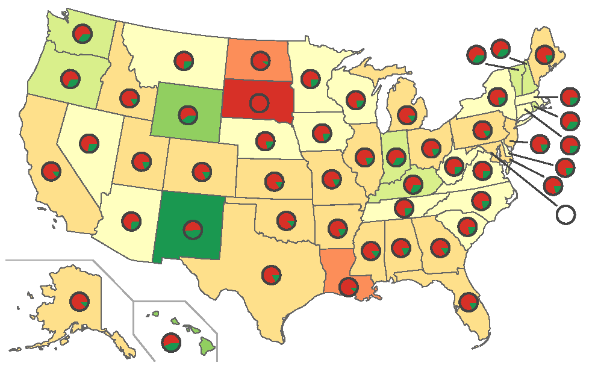

(b) Aggregated WDL result.

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

Table 8: Classification accuracies of unweighted algorithms for the target extraction with 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 9: Statistics of the per state error for the agglomerated WDL result in comparison with batch method LIWC for GP database. SD stands for standard deviation.

	Error (in %)			
	mean	SD	min	max
WDL	5.21	3.71	0.00	13.76
LIWC	38.00	10.55	3.48	67.24

sentiment analysis in which we covered tasks that should be performed before any sentiment extraction. We formulated the sentiment analysis as three sequential 2-class classification schemes. 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 perform sentiment classification on the resulted collection of tweets.

Results of several classification algorithms were presented both the original bag-of-words feature space and in the compressed space. Compression is performed using random reconstructible projections borrowed from the compress sensing community. Empirical results show that learning in the compressed domain (compressed learning) is possible.

In addition we presented a modification of all tested classifiers that can deal with weighted data labels obtained from crowdsourcing. Finally we supplemented our per-tweet analysis with spatially aggregated results and show that our approach also works well for batch-tweet analysis.

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

[3] A. Bermingham and A. F. Smeaton. Classifying sentiment in microblogs: is brevity an advantage? In *CIKM*, pages 1833–1836, 2010.

[4] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD*, pages 245–250, 2001.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. 2007.

[6] D. M. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

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

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

[9] 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.

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

[11] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[12] 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.

[13] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–451, 2004.

[14] D. Fradkin and D. Madigan. Experiments with random projections for machine learning. In *KDD*, pages 517–522, 2003.

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

[16] S. Golder and M. Macy. Diurnal and seasonal mood

vary with work, sleep, and daylength across diverse cultures. *Science*, 333(6051):1878–1881, 2011.

[17] L. Hong and B. Davison. Empirical study of topic modeling in twitter. In *SOMA*, pages 80–88, 2010.

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

[19] 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.

[20] S. P. Kasiviswanathan, P. Melville, A. Banerjee, and V. Sindhwani. Emerging topic detection using dictionary learning. In *CIKM*, pages 745–754, 2011.

[21] E. Kouloumpis, T. Wilson, and J. Moore. Twitter sentiment analysis: The good the bad and the OMG! In *ICWSM*, 2011.

[22] B. Liu. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing*. CRC Press, Taylor and Francis Group, 2nd edition, 2010.

[23] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *J. of Mach. Learn. Res.*, 11:19–60, 2010.

[24] B. OConnor, R. Balasubramanyan, B. Routledge, and N. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *ICWSM*, 2010.

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

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

[27] 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.

[28] H. Shan, A. Banerjee, and N. Oza. Discriminative mixed-membership models. In *ICDM*, pages 466–475, 2009.

[29] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting elections with Twitter: what 140 characters reveal about political sentiment. In *ICWSM*, 2010.

[30] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *NIPS*, pages 2424–2432. 2010.

[31] A. Y. Yang, S. S. Sastry, A. Ganesh, and Y. Ma. Fast $l1$-minimization algorithms and an application in robust face recognition: A review. In *ICIP*, pages 1849–1852, 2010.

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

[33] L. Zhang, R. Ghosh, M. Dekhil, M. Hsu, and B. Liu. Combining lexicon-based and learning-based methods for twitter sentiment analysis. Technical report, HP Laboratories, 2011.

[34] W. X. Zhao, J. Jiang, J. Weng, J. He, and E. Lim. Comparing twitter and traditional media using topic models. In *ECIR*, pages 338–349, 2011.