Economically-Efficient Sentiment Stream Analysis

Roberto Lourenço Jr.
Computer Science Dept.
Universidade Federal de Minas Gerais
robertolojr@dcc.ufmg.br

Adriano Veloso
Computer Science Dept.
Universidade Federal de Minas Gerais
adrianov@dcc.ufmg.br

Adriano Pereira
Computer Science Dept.
Universidade Federal de Minas Gerais
adrianoc@dcc.ufmg.br

Wagner Meira Jr.
Computer Science Dept.
Universidade Federal de Minas Gerais
meira@dcc.ufmg.br

Renato Ferreira
Computer Science Dept.
Universidade Federal de Minas Gerais
renato@dcc.ufmg.br

Srinivasan Parthasarathy
Dept. of Computer Science and Engineering
The Ohio-State University
srini@cse.ohio-state.edu

ABSTRACT

Text-based social media channels, such as Twitter, produce torrents of opinionated data about the most diverse topics and entities. The analysis of such data (aka. sentiment analysis) is quickly becoming a key feature in recommender systems and search engines. A prominent approach to sentiment analysis is based on the application of classification techniques, that is, content is classified according to the attitude of the writer. A major challenge, however, is that Twitter follows the data stream model, and thus classifiers must operate with limited resources, including labeled data and time for building classification models. Also challenging is the fact that sentiment distribution may change as the stream evolves. In this paper we address these challenges by proposing algorithms that select relevant training instances at each time step, so that training sets are kept small while providing to the classifier the capabilities to suit itself to, and to recover itself from, different types of sentiment drifts. Simultaneously providing capabilities to the classifier, however, is a conflicting-objective problem, and our proposed algorithms employ basic notions of Economics in order to balance both capabilities. We performed the analysis of events that reverberated on Twitter, and the comparison against the state-of-the-art reveals improvements both in terms of error reduction (up to 14%) and reduction of training resources (by orders of magnitude).

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis; I.5.2 [**Pattern Recognition**]: Classifier Design and Evaluation

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Sentiment Analysis; Economic Efficiency; Streams and Drifts

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.

1. INTRODUCTION

The need for real-time text analytics is clear and present given the ubiquitous reach of social media sites like Facebook and Twitter. Specifically, recognizing customer sentiment in real-time and enabling advertising on-the-fly have the potential to be a breakthrough technology [**?**]. Early examples of such technology in use were demonstrated in this year's National Football League's Superbowl (a premier sporting event in the USA) where a well known manufacturer of *Oreo* cookies took advantage of a third quarter blackout (and associated Twitter sentiment) to embed a contextual advertisement. Another example at the same event was the advertisement for a Hollywood movie, where, based on the initial advertisement which happened before the start of the first quarter (and associated Twitter sentiment), the decision on which of several possible advertisements to run later on in the program was apparently taken as a runtime decision. Examples like these are likely to occur more frequently due to lightweight and easy communication mechanisms, such as Twitter microblogging, which makes people eager not only to exchange information, but also to convey their opinions and emotions. People watch events together on television, while tweeting out about things happening around them. As a result, opinionated content is created almost at the same time the event is happening in the real world, and becomes available shortly after. The analysis of such content (aka. sentiment analysis) in order to exploit the aggregate sentiment of the online crowd goes beyond advertising, and is becoming crucial to recommender systems and search engines.

There is a growing trend in performing sentiment analysis using classification-related techniques: a process that automatically builds a classification model by learning, from a set of previously labeled data (i.e., the training-set), the underlying characteristics that distinguish one sentiment from another (i.e., happiness, madness, surprise, suspicion). The success of these classifiers rests on their ability to judge attitude by means of textual-patterns present in the data, which usually appear in the form of (idiomatic) expressions and combinations of words. Sentiment analysis over Twitter real-time messages, however, is particularly challenging, because: (i) Twitter follows the data stream model[1], requiring classifiers to operate with limited computing and training resources, and (ii) either sentiment distribution or the characteristics related to certain

[1]There are three main source streams in Twitter. The Firehose provides all status updates from everyone in real-time. Spritzer and Gardenhose are two sub-samples of the Firehose. The current sampling rates are 5% and 15%, respectively.

sentiments may change over time in almost unforeseen ways (i.e., sentiment drift).

Our Approach to Sentiment Stream Analysis

A possible strategy to cope with the aforementioned challenges is to employ selective sampling algorithms in order to focus only on the most relevant training examples/messages at each time step and to creating training sets from which classifiers are built. Such training sets are kept as small as possible to ensure fast learning times, since a new classifier must be built at each time step, after a new target message arrives. Also, messages should be selected so that the resulting training set provides sufficient resources to enable the resulting classifier to be effective under the occurrence of drifts. In order to provide sufficient training resources while keeping sets small, our algorithms select training messages by taking into account two important properties, that we define as adaptiveness and memorability. Informally, adaptiveness enables the classifier to adapt itself to drifts, and thus, improving adaptiveness involves incorporating fresh messages into the current training set, while discarding obsolete ones. Memorability, on the other hand, involves retaining messages belonging to pre-drift distributions, therefore enabling the classifier to recover itself from drifts.

We hypothesize that adaptiveness and memorability are both necessary to make classifiers robust to drifts. However, given their antagonistic natures, improving both properties may lead to a conflicting-objective problem, in which the attempt to improve memorability further may result in worsening adaptiveness. Thus, we tackle the problem by proposing selective sampling algorithms based on multi-objective optimization, that is, we propose to select training messages so that the resulting classifier achieves a proper balance between memorability and adaptiveness. Our algorithms are based on central concepts in Economics, namely *Pareto* and *Kaldor-Hicks* efficiency criteria [**?**, **?**, **?**]. The Pareto Efficiency criterion informally states that "when some action could be done to make someone better off without hurting anyone else, then it should be done." This action is called Pareto improvement, and a system is said to be Pareto-Efficient if no such improvement is possible. The Kaldor-Hicks criterion is less stringent and states that "when some action could be done to make someone better off, and this could compensate those that are made worse off, then it should be done."

Contributions and Findings

The main contribution of this paper is to exploit the intuition behind the aforementioned concepts for devising new algorithms for sentiment stream analysis. In practice, we claim the following benefits and contributions:

- We formulate simple-to-compute yet effective utility measures that capture the notions of adaptiveness and memorability. For instance, the similarity between messages that are candidate to compose the current training set and the target message, as well as the freshness of the candidate messages, are measures that tend to privilege adaptiveness. In contrast, candidate messages are also randomly shuffled, thus privileging memorability. These utility measures result in a utility space, and the extent to which each candidate message contributes to adaptiveness and memorability depends on where it is placed in this space.

- We exploit the concept of Pareto Efficiency by separating messages (viewed as points in the utility space) that are not dominated by any other message. These messages compose the Pareto frontier [**?**], and messages lying in this frontier correspond to cases for which no Pareto improvement is possible. These messages privilege either adaptiveness or memorability, and thus they are selected to compose the current training set from which the classifier is built.

- We exploit the concept of Kaldor-Hicks Efficiency by selecting an additional set of messages that, although not lying in the Pareto frontier, correspond to a positive trade-off between adaptiveness and memorability. These messages are selected to compose the current training set from which the classifier is built.

- Our algorithms may operate either on an instance-basis or in batch-mode, by employing classification models based on sentiment rules that are kept incrementally as the stream evolves and training sets are modified.

To evaluate the effectiveness of our algorithms, we performed experiments using Twitter data collected from three important events in 2010, spanning different sentiments expressed in different languages. Results show that our algorithms make classifiers extremely effective, with gains in prediction performance that are up to 14% when compared against the state-of-the-art. Further, the amount of training resources needed is decreased by two orders of magnitude.

2. RELATED WORK

In the data stream model, data arrives at high speed and algorithms must work in real time and with limited resources. Further, in some domains, algorithms must deal either with burst detection [**?**] and concept drift (i.e., data which nature or distribution change over time). Žliobaitė [**?**] categorizes such drifts as sudden, gradual, incremental and recurring. When data distribution or nature change over time, its relevance must be recalculated to avoid harming the model. This kind of data stream is known as evolving data streams.

Many techniques have been proposed to allow accurate classification in evolving data streams. Núñez et al. [**?**] proposed a method for keeping a variable training window by adjusting internal structures of decision trees. An ensemble of Hoeffding trees have been proposed in [**?**], each tree is limited to a small subset of attributes. Gama et al. [**?**] proposed a mechanism to discard old information based on sliding windows. Bifet et al. [**?**, **?**] proposed an adaptive sliding window algorithm, called ADWIN, suitable for data streams with sudden drifts. The approach presented in [**?**] suggests that a time-based forgetting function, which makes more recent observations more significant, provides adaptiveness to the classifier. Klinkenberg [**?**] compares example selection, often used in windowing approaches with example weights. Experiments show that both approaches are effective. In [**?**] the authors proposed an approach based on a training augmentation procedure, which automatically incorporates relevant training messages into the training-set.

Some works have focused on feature similarity, such as Torres et al. [**?**] that studied different methods for data stream classification and proposed a new way of keeping the representative data models based on similarity measures. Feng et al. [**?**] extracted the concept from each data block using feature similarity probabilities. Masud et al. [**?**] proposed a novel technique to overcome the lack of labeled examples by building models from unlabeled instances and a small amount of labeled ones. Zhu et al. [**?**] employed active learning to produce a classifier ensemble that selects labeled instances from data streams to build classifiers. Also, in [**?**, **?**] active learning approaches are presented for data streams that explicitly handle

concept drifts. They are based on uncertainty [**?**], dynamic allocation of labeling efforts over time, and randomization of the search space. Žliobaitė et al. [**?**] proposed a system that implements active learning strategies, extending the Massive Online Analysis (MOA) framework [**?**].

Works above cited attempt to face concept drift in data stream through manipulation of classifiers, with mechanisms such as training windows and decay functions, active learning and sampling. In this paper we present new algorithms that select high-utility examples in order to provide adaptiveness and memorability to the classifier. In order to balance adaptiveness and memorability, we formalized this issue as a multi-objective problem. The sample selection is performed using economic efficiency criteria: Pareto and Kaldor-Hicks. We did not find in the recent literature approaches that employ multi-objective models based on economic efficiency criteria to deal with issues in the data stream environment.

3. ALGORITHMS

In this section we present novel selective sampling approaches for learning classifiers to distinguish between different sentiments expressed in Twitter messages. We start by discussing models based on specialized association rules. Then we present measures for adaptiveness and memorability, and describe the message utility space. Finally, we discuss Pareto and Kaldor-Hicks criteria, and algorithms that select training messages using these criteria.

3.1 Sentiment Stream Analysis

In our context, the task of learning sentiment streams is precisely defined as follows. At time step n, we have as input a training set referred to as $\mathcal{D}_n$, which consists of a set of records of the form $< d, s_i >$, where d is a message (represented as a list of terms), and s_i is the sentiment implicit in d. The sentiment variable s draws its values from a pre-defined, fixed and discrete set of possibilities (e.g., s_1, s_2, ..., s_k). The training set is used to build a classifier relating textual patterns in the messages to their corresponding sentiments. A sequence of future messages referred to as $\mathcal{T} = \{t_n, t_{n+1}, \ldots\}$, consists of messages for which only their terms are known, while the corresponding sentiments are unknown. The classifier obtained from $\mathcal{D}_n$ is used to score the sentiments for message t_n in $\mathcal{T}$. Messages in $\mathcal{T}$ are eventually incorporated into the next training set.

There are countless strategies for devising a classifier for sentiment analysis. Many of these strategies, however, are not well-suited to deal with data streams. Some are specifically devised for offline classification [**?**, **?**], and this is problematic because producing classifiers on-the-fly would be unacceptably costly. In such circumstances, alternate classification strategies may become more convenient [**?**].

3.2 Sentiment Rules and Classifiers

Next we describe classifiers composed of association rules, and how these rules are used for sentiment-scoring. Such classifiers are built on-the-fly [**?**, **?**], being thus well-suited for sentiment stream analysis, as shown in [**?**].

Definition 1. A sentiment rule is a specialized association rule $\mathcal{X} \rightarrow s_i$, where the antecedent $\mathcal{X}$ is a set of terms (i.e., a termset), and the consequent s_i is the predicted sentiment. The domain for $\mathcal{X}$ is the vocabulary of the training set $\mathcal{D}_n$. The support of $\mathcal{X}$ is denoted as $\sigma(\mathcal{X})$, and is the number of messages in $\mathcal{D}_n$ having $\mathcal{X}$ as a subset. The confidence of rule $\mathcal{X} \rightarrow s_i$ is denoted as $\theta(\mathcal{X} \rightarrow s_i)$ and is given as $\dfrac{\sigma(\mathcal{X} \cup s_i)}{\sigma(\mathcal{X})}$.

Sentiment Scoring

We denote as $\mathcal{R}(t_n)$ the classifier obtained at time step n, by extracting rules from $\mathcal{D}_n$. Basically, the classifier is a poll of rules, and each rule $\{\mathcal{X} \rightarrow s_i\} \in \mathcal{R}(t_n)$ is a vote given for sentiment s_i. Given message t_n, a rule is a valid vote if it is applicable to t_n.

Definition 2. A rule $\{\mathcal{X} \rightarrow s_i\} \in \mathcal{R}(t_n)$ is said to be applicable to message $t_n \in \mathcal{T}$ if all terms in $\mathcal{X}$ are in t_n.

We denote as $\mathcal{R}_a(t_n)$ the set of rules in $\mathcal{R}(t_n)$ that are applicable to message t_n. Thus, only rules in $\mathcal{R}_a(t_n)$ are considered as valid votes when scoring sentiments in t_n. Further, we denote as $\mathcal{R}_a^{s_i}(t_n)$ the subset of $\mathcal{R}(t_n)$ containing only rules predicting sentiment s_i. Votes in $\mathcal{R}_a^{s_i}(t_n)$ have different weights, depending on the confidence of the corresponding rules. The weighted votes for sentiment s_i are averaged, giving the score for s_i with regard to t_n:

$$s(t_n, s_i) = \sum \frac{\theta(\mathcal{X} \rightarrow s_i)}{|\mathcal{R}_a^{s_i}(t_n)|} \tag{1}$$

Finally, the scores are normalized, thus giving the likelihood of sentiment s_i being the attitude in message t_n:

$$\hat{p}(s_i|t_n) = \frac{s(t_n, s_i)}{\sum_{j=1}^{k} s(t_n, s_j)} \tag{2}$$

Rule Extraction

The simplest approach to rule extraction is the offline one. In this case, rule extraction is divided into two steps: support counting and confidence computation. Once the support $\sigma(\mathcal{X})$ is known, it is straightforward to compute the confidence $\theta(\mathcal{X} \rightarrow s_i)$ for the corresponding rules [**?**]. There are several smart support-counting strategies [**?**, **?**, **?**], and many fast implementations [**?**] that can be used. We employ the vertical counting strategy, which is based on the use of inverted lists [**?**]. Specifically, an inverted list associated with termset $\mathcal{X}$, is denoted as $\mathcal{L}(\mathcal{X})$, and contains the identifiers of the messages in $\mathcal{D}_n$ having termset $\mathcal{X}$ as a subset. An inverted list $\mathcal{L}(\mathcal{X})$ is obtained by performing the intersection of two proper subsets of termset $\mathcal{X}$. The support of termset $\mathcal{X}$ is given by the cardinality of $\mathcal{L}(\mathcal{X})$, that is, $\sigma(\mathcal{X}) = |\mathcal{L}(\mathcal{X})|$.

Usually, the support for different sets of terms in $\mathcal{D}_n$ are computed in a bottom-up way, which starts by scanning all messages in $\mathcal{D}_n$ and computing the support of each term in isolation. In the next iteration, pairs of terms are enumerated, and their support values are calculated by performing the intersection of the corresponding proper subsets. The search for sets of terms proceeds, and the enumeration process is repeated until the support values for all sets of terms in $\mathcal{D}_n$ are finally computed. Obviously, the number of rules increases exponentially with the size of the vocabulary (i.e., the number of distinct terms in $\mathcal{D}_n$), and computational cost restrictions have to be imposed during rule extraction. Typically, the search space for rules is restricted by pruning rules that do not appear frequently in $\mathcal{D}_n$ (i.e., the minimum support approach). While such restrictions make rule extraction feasible, they also lead to lossy classifiers, since some rules are pruned and therefore are not included into $\mathcal{R}(t_n)$.

Online Rule Extraction. An alternative to offline rule extraction is to extract rules on-the-fly. Such alternative, which we call online rule extraction, has several advantages [**?**]. For instance, it becomes possible to efficiently extract rules from $\mathcal{D}_n$ without performing support-based pruning. The idea behind online rule extraction is to ensure that only applicable rules are extracted by projecting $\mathcal{D}_n$ on a demand-driven basis. More specifically, rule extraction is delayed until a message $t_n \in \mathcal{T}$ is given. Then, terms in t_n are used as a filter which configures $\mathcal{D}_n$ in a way that only rules that are applicable to t_n can be extracted. This filtering process produces a projected training-set, denoted as $\mathcal{D}_n^*$, which contains only terms that are present in message t_n.

Lemma 1. All rules extracted from $\mathcal{D}_n^*$ are applicable to t_n.

Proof. Since all training messages in $\mathcal{D}_n^*$ contain only terms that are present in message t_n, the existence of a rule $\mathcal{X} \rightarrow s_i$ extracted from $\mathcal{D}_n^*$, such that $\mathcal{X} \nsubseteq t_n$, is impossible. ■

Lemma 1 implies that online rule extraction assures that $\mathcal{R}(t_n) = \mathcal{R}_a(t_n)$. The next theorem states that search space for rules induced by $\mathcal{D}_n^*$ is much narrower than the search space for rules induced by $\mathcal{D}_n$. Thus, rules can be efficiently extracted from $\mathcal{D}_n^*$, no matter the minimum-support value (which can be arbitrary low).

Theorem 1. The number of rules extracted from $\mathcal{D}_n^*$ increases polynomially with the number of distinct terms in $\mathcal{D}_n$.

Proof. Let k be the number of distinct terms in $\mathcal{D}_n$. Since an arbitrary message $t_n \in \mathcal{T}$ contains at most l terms (with $l \ll k$), then any rule applicable to t_n can have at most l terms in its antecedent. That is, for any rule $\{\mathcal{X} \rightarrow s_i\}$, such that $\mathcal{X} \subseteq t_n$, $|\mathcal{X}| \leq l$. Consequently, the number of possible rules that are applicable to t_n is $l + \binom{l}{2} + \ldots + \binom{l}{l} = O(2^l) \ll O(k^l)$. Thus, the number of applicable rules increases polynomially in k. ■

Extending Classifiers Dynamically. Let $\mathcal{R} = \{\mathcal{R}(t_1) \cup \mathcal{R}(t_2) \cup \ldots \cup \mathcal{R}(t_n)\}$. With online rule extraction, $\mathcal{R}$ is extended dynamically as messages $t_i \in \mathcal{T}$ are processed. Initially $\mathcal{R}$ is empty; a classifier $\mathcal{R}_{t_i}$ is appended to $\mathcal{R}$ every time a message t_i is processed. Producing a classifier $\mathcal{R}(t_i)$ involves extracting rules from the corresponding training-set. This operation has a significant computational cost, since it is necessary perform multiple accesses to $\mathcal{D}_i$. Different messages in $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ may demand different classifiers $\{\mathcal{R}_{t_1}, \mathcal{R}_{t_2}, \ldots, \mathcal{R}_{t_m}\}$, but different classifiers may share some rules (i.e., $\{\mathcal{R}_{t_i} \cap \mathcal{R}_{t_j}\} \neq \emptyset$). In this case, memorization is very effective in avoiding work replication, reducing the number of data access operations. Thus, before extracting rule $\mathcal{X} \rightarrow s_i$, the classifier first checks whether this rule is already in $\mathcal{R}$. If an entry is found, then the rule in $\mathcal{R}$ is used instead of extracting it from the training-set. If it is not found, the rule is extracted from the training-set and then it is inserted into $\mathcal{R}$.

3.3 Utility Space and Selective Sampling

Our approach to sentiment stream analysis is based on selecting high-utility messages to compose the training set at each time step. Training sets must provide adaptiveness and memorability to the corresponding classifiers. Improving adaptiveness and memorability simultaneously, however, is a conflicting-objective problem. Instead, our approaches create training sets that balance between adaptiveness and memorability. Specifically, at each time step, candidate messages are placed into an n-dimensional space, in which each dimension corresponds to a utility measure which is either related to adaptiveness or memorability.

Utility Measures

At each time step, the classifier must score sentiments that are expressed in the target message. Some of the utility measures we are going to discuss next are based on the distance to the target message. By minimizing such distance we are essentially maximizing adaptiveness, since the selected messages are similar to the target message. As for memorability, we are going to discuss a utility measure based on randomly shuffling candidate messages:

- **Distance in space** − The similarity between the target message t_n and an arbitrary message t_j is given by the number of rules in the classifier $\mathcal{R}_a(t_n)$ that are also applicable to t_j. Differently from traditional measures such as cosine and Jaccard [**?**], the rule-based similarity considers not only isolated terms, but also combination of terms. Thus, the utility of message t_j is given as:

$$U_s(t_j) = \frac{|\mathcal{R}_a(t_n) \cap \mathcal{R}_a(t_j)|}{|\{\mathcal{R}_a(t_n)\}|} \tag{3}$$

- **Distance in time** − Let $\gamma(t_j)$ be a function that returns the time in which message t_j arrived. The utility of message t_j is given as:

$$U_t(t_j) = \frac{\gamma(t_j)}{\gamma(t_n)} \tag{4}$$

- **Memorability** − In order to provide memorability, the training set must contain messages posted in different time periods. A simple way to force this is to generate a random permutation of the candidate messages, that is, randomly shuffling the candidate messages [**?**]. Let $\alpha(t_j)$ be a function that returns the position of message t_j in the shuffle. The utility of message t_j is given as:

$$U_r(t_j) = \frac{\alpha(t_j)}{|\mathcal{D}_n|} \tag{5}$$

Each candidate message is judged based on these three utility measures. The need to judge one situation better than another motivates much of Economics, and next we discuss concepts from Economics and how they can be applied to select messages to compose the training set.

Figure 1: Illustrative example. The 3D utility space.

3.4 Economic Efficiency

When the society is economically efficient, any changes made to assist one person would harm another. The same intuition could be exploited for the sake of selecting messages to compose the training set at each time step. In this case, a training set is economically efficient if it is only possible to improve memorability at the cost of adaptiveness, and vice-versa [**?**, **?**].

There is an alternative, less stringent notion of efficiency, which is based on the principle of compensation [**?**]. Under new arrangements in the society, some may be better off while others may