Economically-efficient sentiment stream analysis

Conference Paper · July 2014

DOI: 10.1145/2600428.2609612

6 authors, including:

Roberto Lourenço Oliveira Júnior
Microsoft, Vancouver, Canada
6 PUBLICATIONS **94** CITATIONS
SEE PROFILE

Adriano Veloso
Federal University of Minas Gerais
220 PUBLICATIONS **4,526** CITATIONS
SEE PROFILE

Adriano C. M. Pereira
Federal University of Minas Gerais
169 PUBLICATIONS **2,923** CITATIONS
SEE PROFILE

Wagner Meira Jr.
Federal University of Minas Gerais
617 PUBLICATIONS **12,398** CITATIONS
SEE PROFILE

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 [20]. 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 [19,22,28]. 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 [28], 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 [42] and concept drift (i.e., data which nature or distribution change over time). Žliobaitė [35] 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. [27] 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 [5], each tree is limited to a small subset of attributes. Gama et al. [17] proposed a mechanism to discard old information based on sliding windows. Bifet et al. [6, 7] proposed an adaptive sliding window algorithm, called ADWIN, suitable for data streams with sudden drifts. The approach presented in [24] suggests that a time-based forgetting function, which makes more recent observations more significant, provides adaptiveness to the classifier. Klinkenberg [23] compares example selection, often used in windowing approaches with example weights. Experiments show that both approaches are effective. In [30] 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. [31] 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. [16] extracted the concept from each data block using feature similarity probabilities. Masud et al. [25] 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. [41] employed active learning to produce a classifier ensemble that selects labeled instances from data streams to build classifiers. Also, in [37, 38] active learning approaches are presented for data streams that explicitly handle concept drifts. They are based on uncertainty [21],

dynamic allocation of labeling efforts over time, and randomization of the search space. Žliobaitė et al. [36] proposed a system that implements active learning strategies, extending the Massive Online Analysis (MOA) framework [8].

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 [12, 14], and this is problematic because producing classifiers on-the-fly would be unacceptably costly. In such circumstances, alternate classification strategies may become more convenient [33].

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 [32,34], being thus well-suited for sentiment stream analysis, as shown in [30].

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 [40]. There are several smart support-counting strategies [1,18,40], and many fast implementations [3] that can be used. We employ the vertical counting strategy, which is based on the use of inverted lists [39]. 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 [30]. For instance, it becomes possible to efficiently extract rules from $\mathcal{D}_n$ without per-

forming 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 [2], 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 [15]. 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.

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 [26, 29].

There is an alternative, less stringent notion of efficiency, which is based on the principle of compensation [13]. Under new arrangements in the society, some may be better off while others may be worse off. Compensation holds if those made better off under the new set of conditions could compensate those made worse off. Next we discuss algorithms that exploit these two notions of economic efficiency in order to select messages to compose the training sets.

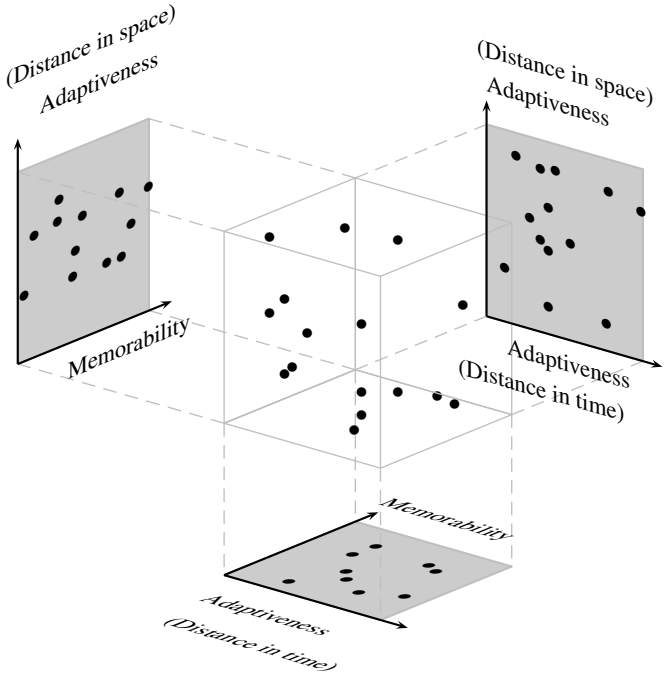


Figure 1: Illustrative example. The 3D utility space.

Pareto Frontier

Messages that are candidate to compose the training set at time step n are placed in a 3-dimensional space, according to their utility measures, as shown in Figure 1. Thus, each message a is a point in such utility space, and is given as $< U_s(a), U_t(a), U_r(a) >$.

Definition 3. Message a is said to dominate message b iff both of the following conditions are hold:

- $U_s(a) \geq U_s(b)$ and $U_t(a) \geq U_t(b)$ and $U_r(a) \geq U_r(b)$
- $U_s(a) > U_s(b)$ or $U_t(a) > U_t(b)$ or $U_r(a) > U_r(b)$

Therefore, the dominance operator relates two messages so that the result of the operation has two possibilities as shown in Figure 2 (Left): (i) one message dominates another or (ii) the two messages do not dominate each other.

Definition 4. Training set $\mathcal{P}_n = \{d_1, d_2, \ldots, d_m\}$ is said to be Pareto-efficient at time step n, if $\mathcal{P}_n \subseteq \mathcal{D}_n$ and there is no pair of messages $(d_i, d_j) \in \mathcal{P}_n$ for which d_i dominates d_j.

Messages that are not dominated by any other message, lie on the Pareto frontier [28]. Therefore, by definition, the Pareto-efficient training set at time step n, $\mathcal{P}_n$, is composed by all the messages lying in the Pareto frontier that is built from $\mathcal{D}_n$. There are efficient algorithms for building and maintaining the Pareto frontier, and we employed the algorithm proposed in [11] which ensures $O(|\mathcal{D}_n|)$ complexity. We denote the process of exploiting Pareto-efficient training sets as Pareto-Efficient Selective Sampling, or simply PESS. Figure 2 (Middle) shows an illustrative example of a Pareto frontier built from arbitrary points in the utility space.

Kaldor-Hicks Region

The PESS strategy follows a stringent criterion, which tends to select only few messages to compose the training sets. As a result, the training sets may become excessively small and prone to noise. The Kaldor-Hicks criterion, on the other hand, follows a cost-benefit analysis and circumvents the small training set problem by stating that efficiency is achieved if those that are made better off could in theory compensate those that are made worse off. Thus, under the Kaldor-Hicks criterion, an utility measure can compensate other utility measures, and therefore, this criterion selects messages that are located inside a region which is below the Pareto frontier. To define this region we must first define the overall utility of a message.

Definition 5. Assuming that all measures are equally important, the overall utility of an arbitrary message $d_i \in \mathcal{D}_n$ is:

$$U(d_i) = U_s(d_i) + U_t(d_i) + U_r(d_i) \tag{6}$$

That is, the overall utility of a message is given as the sum of its utility measures. Also, the baseline message, which is denoted as d^*, is defined as:

$$d^* = \{d_i \in \mathcal{P}_n | \forall d_j \in \mathcal{P}_n : U(d_i) \leq U(d_j)\} \tag{7}$$

That is, the baseline is the message lying in the frontier for which the overall utility assumes its lowest value.

The Kaldor-Hicks region is composed of messages for which the overall utility is not smaller than the baseline overall utility. Such baseline utility is the utility associated with the message lying in the Pareto frontier for which the overall utility is the lowest.

Definition 6. Training set $\mathcal{K}_n = \{d_1, d_2, \ldots, d_m\}$ is said to be Kaldor-Hicks-efficient at time step n, if $\mathcal{P}_n \subseteq \mathcal{K}_n \subseteq \mathcal{D}_n$, and there is no message $d_i \in \mathcal{K}_n$ such that $U(d^*) > U(d_i)$.

We denote the process of exploiting Kaldor-Hicks-efficient training sets as Kaldor-Hicks-Efficient Selective Sampling, or simply KHSS. Figure 2 (Right) shows an illustrative example of a Kaldor-Hicks region built from arbitrary points in the utility space.

4. EXPERIMENTAL EVALUATION

In this section we empirically analyze the performance of our classifiers. We employ the mean squared error (MSE) as the basic evaluation measure in our experiments, since we are primarily interested in evaluating sentiment scoring given by Equation 2. The MSE measure is given as:

$$MSE = \frac{1}{|\mathcal{T}|} \sum_{\forall t_i \in \mathcal{T}} (1 - \hat{p}(s_i|t_i))^2 \tag{8}$$

where s_i is the correct sentiment associated with message $t_i \in \mathcal{T}$, and $\hat{p}(s_i|t_i)$ is the sentiment score assigned by the classifier to message $t_i \in \mathcal{T}$.

To evaluate the amount of computing resources used as the stream evolves, we employ the RAM-Hours measure [9], where every RAM-Hour equals a GB of RAM deployed for 1 hour of execution. We also evaluate the amount of training resources used over time, as the number of messages labeled during the process. We used Hoeffding Adaptive Trees [4, 10] (abbreviated as HAT), Active Classifier [37, 38] (abbreviated as AC), and Incremental Lazy Associative Classifier [30] (abbreviated as ILAC) as baselines. All datasets

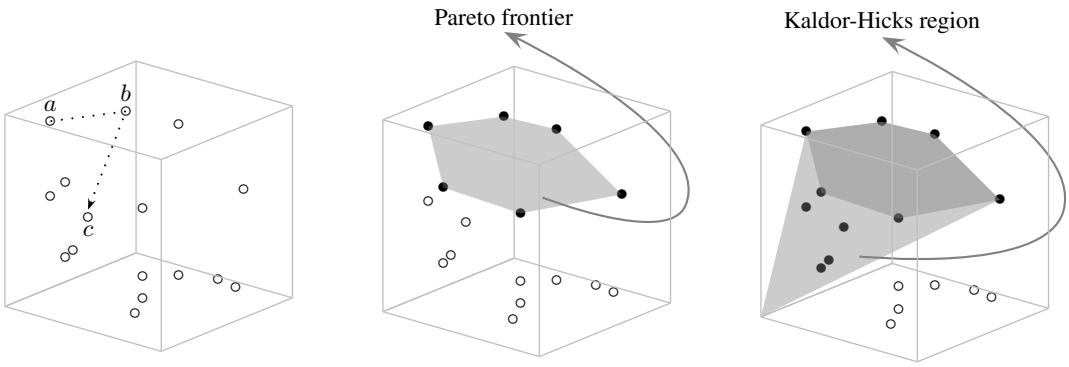


Figure 2: Illustrative example. (Left) The dominance operator: neither a or b dominates each other, but b dominates c. (Middle) Points lying in the Pareto frontier. (Right) Points inside the Kaldor-Hicks region.

used in our experiments were manually labeled by three to five human annotators. We expended significant time, effort, and resources to obtain high quality (labeled) data from Twitter streams, which shall be made available at publication time.

All experiments were performed on a 1.93 GHz Core i7 machines with 8GB of memory, using the MOA system [8], an environment for running experiments with evolving data streams. Our evaluation follows the Test-Then-Train methodology, in which each individual message in $\mathcal{T}$ is used to test the classifier and then it becomes available for training. Finally, we consider three possible settings:

- **Instance Processing** – Once processed, message t_n is included into D_{n+1}, and then a new classifier is built. Under this setting, message t_n is mandatorily labeled.

- **Batch Processing** – After a batch of b messages $B = \{t_n, t_{n+1}, \ldots, t_{n+b}\}$ is processed, only a subset of B is included into D_{n+b}, since some messages in B may be similar to each other. Under this setting, not all messages in B need to be labeled, since only a subset of B is included into D_{n+b}. Therefore, there is a trade-off between batch size and labeling effort, and a similarity threshold, denoted as δ, controls the messages in the batch that must be labeled.

In the following we describe our evaluation scenarios and discuss the performance of the classifiers.

4.1 Brazilian Presidential Elections

The presidential election campaigns were held from June to October 2010. the candidate Dilma Rousseff launched a Twitter page during a public announcement, and she used Twitter as one of the main sources of information for her voters. The campaign attracted more than 500,000 followers and as a result Dilma was the second most cited person on Twitter in 2010. The election came to a second round vote, and Dilma Rousseff won the runoff with 56% of the votes.

Dilma Rousseff Election Campaign

We collected 66,643 messages in Portuguese referencing Dilma Rousseff in Twitter during her campaign. We labeled these messages in order to track the population sentiment of approval during this period. As shown in Figure 3 (a), approval varied significantly over the time due to several polemic statements and political attacks, and our goal is to score approval during her campaign.

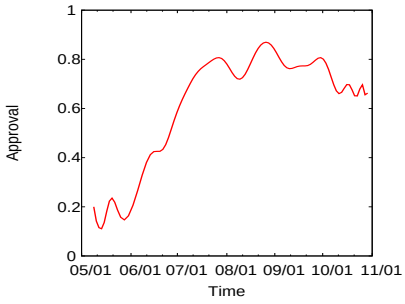
Figure 3 (b) shows the results in terms of MSE obtained for the evaluation of the classifiers in this dataset. All classifiers evaluated in this experiment operate on an instance basis. The x-axis represents different time steps (i.e., each message that passes in the stream), while the y-axis shows the MSE so far. As it can be seen, a better approximation is obtained using our proposed algorithms, namely PESS and KHSS. AC and ILAC were very competitive during all the campaign. Both PESS and KHSS algorithms started much better than the other competing algorithms, but slowly converges to the baseline numbers as the stream evolves.

Figure 3 (c) shows results concerning the proposed algorithms when operating in batch mode. The figure shows the number of messages that were labeled during the process as a function of δ, the minimum similarity threshold discussed in Section 3.1. Basically, we calculate the Jaccard coefficient associated with each possible pair of messages in the batch, and if the coefficient is greater than δ, the corresponding messages are merged into a new one. The process continues merging similar messages until no pair of messages are similar enough, and the process stops. At the end, only the merged messages are labeled. Clearly, higher values of δ implies that less messages are merged, and thus incurring more labeling effort. Further, as the figure shows, the dependence between labeling effort and δ tends to be linear.

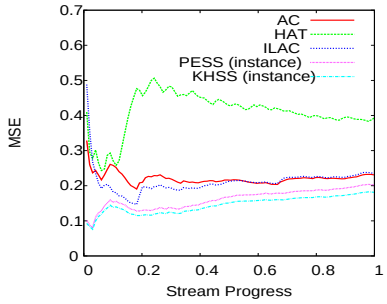
By varying δ, we also study the trade-off between labeling effort and MSE. As shown in Figure 3 (d), MSE decreases if more labeling effort is spent during the process. Specifically, best results are achieved when about 40% of the messages in the stream are labeled during the process. Although both PESS and KHSS require the same amount of training resources, KHSS provides slightly better MSE numbers. Furthermore, smaller batch sizes incur in less labeling effort for this dataset.

We assume that HAT requires only the target message for updating its tree model, and thus we consider that the training set is composed only by the target message. The AC algorithm requires much more messages within each training set. An abrupt decrease in the number of training messages is always observed after drifts. The proposed PESS algorithm requires very small training sets, since the Pareto frontier at each time step is composed by few messages, but these messages are still able to make the classifier robust to drifts as the stream evolves. Further, despite being less stringent than PESS, the proposed KHSS algorithm also requires small training sets, as shown in Figure 3 (e).

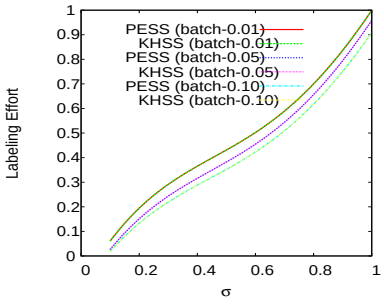
Figure 3 (f) shows RAM-Hours numbers for the algorithms. AC, as well as PESS (instance) and KHSS (instance), are clearly the



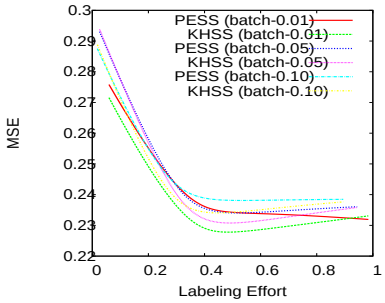
(a) Approval over Dilma Rousseff's campaign. Approval sentiment varied greatly from 05/2010 to 11/2010.



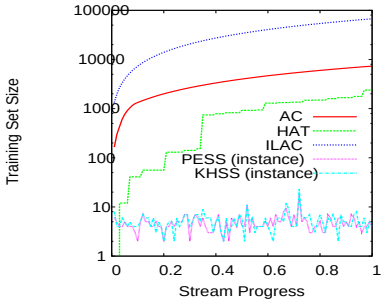
(b) MSE numbers as the stream evolves.



(c) X-Y scatter plot correlating the minimum similarity threshold α and labeling effort.



(d) X-Y scatter plot correlating labeling effort and MSE numbers.



(e) Size of the training sets as the stream evolves.

Figure 3: Brazilian Presidential Elections. Tweets are in Portuguese.

best performers in terms of amount of computing resources required. Also, resources required during the process significantly increases when PESS and KHSS operate on batch mode, but still, ILAC is the worst performer.

4.2 TIME's Person of the Year

Every year, TIME magazine selects the person (or a group of persons) that has mostly influenced during the year. The chosen person for 2010 was Mark Zuckerberg. The reader choice, however, was Julian Assange, with an overwhelming superiority of votes.

Zuckerberg and Assange

We collected 5,616 messages in English referencing Julian Assange and Mark Zuckerberg from 1-15-2010 to 12-21-2010. We labeled them in order to track diverse sentiments regarding the magazine's decision. Sentiments include (dis)approval, surprise (since the reader choice was pointing to Julian Assange), and even fury. The dataset contains 7,294 distinct terms, and a message is posted every 45 seconds, on average.

Figure 4 (a) shows the results in terms of MSE. As it can be seen, a better approximation is obtained by HAT and ILAC. For this dataset, AC was not effective in the first time steps. At the end of the process, both PESS (instance) and KHSS (instance) algorithms achieved competitive numbers when compared against the best performers.

Figure 4 (b) shows the trade-off between labeling effort and MSE. Again, MSE numbers decrease as more labeling effort is spent during the process. This trend is particularly evidenced for smaller batch sizes. Further, the KHSS algorithms shows a better trade-off between labeling effort and MSE. Finally, Figure 4 (c) shows RAM-Hours numbers for the evaluated algorithms. The AC algorithm, as well as PESS (instance) and KHSS (instance) are, again, extremely competitive in terms of amount of computing resources required. Further, the amount of resources required during the process significantly increases when PESS and KHSS operate on batch mode, but still, ILAC is the worst performer.

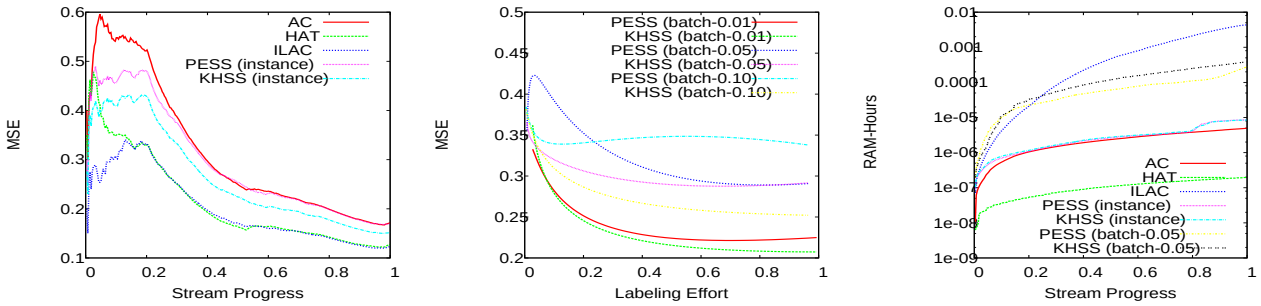
4.3 FIFA World Cup

The 2010 Soccer World Cup involved 32 teams. The Brazilian team was defeated by the Dutch team on 07-02-2010, after a controversial match. The Brazilian team scored first, but soon after the Dutch team scored twice and won the match. A specific player, Felipe Melo, had decisive participation (for better and worse) in all three goals. Specifically, Figure 5 (a) shows how the appreciation for Felipe Melo varied during the match.

The Brazilian Defeat

We collected 3,214 messages in Portuguese referencing Felipe Melo that were posted in Twitter as the match was happening. We labeled them in order to track the appreciation for the participation of Felipe Melo.

Figure 5 (b) shows the results in terms of MSE.As it can be seen, the AC algorithm achieved the worst MSE numbers for this dataset. On the other hand, HAT, ILAC, as well as PESS (instance) and KHSS (instance) showed extremely competitive numbers. This is expected, since this dataset contains three sudden drifts (as shown in Figure 5 (a)), and HAT, ILAC, PESS (instance) and KHSS (instance) were all able to ensure adaptiveness. For this dataset, memorability is not mandatory (as the sentiment distribution never returns to a pre-drift distribution), and thus PESS (instance) and KHSS (instance) were not able to provide significant improvements, although being the best performers overall.



(a) MSE numbers as the stream evolves.

(b) X-Y scatter plot correlating labeling effort and MSE numbers.

(c) RAM-Hours as the stream evolves.

Figure 4: Person of the Year. Tweets are in English.

Figure 5 (c) shows a X-Y scatter plot correlating δ and labeling effort. The correlation is almost linear. The trade-off between labeling effort and MSE is shown in Figure 5 (d). Clearly, MSE decreases with the effort spent to label messages. Figure 5 (e) shows the number of messages composing the training set at each time step. As in previous cases, AC and ILAC require much more training resources than other competing algorithms. PESS (instance) as well as KHSS (instance) require much less training messages, again, showing that the selective sampling strategy is effective in producing small and effective sets at each time step.

Finally, Figure 5 (f) shows RAM-Hours numbers. In this case, AC, as well as PESS (instance) and KHSS (instance), are clearly the best performers in terms of amount of computing resources required. Further, the amount of resources required during the process significantly increases when PESS and KHSS operate on batch mode, but still, as in other datasets, ILAC is the worst performer.

5. CONCLUSIONS

This paper focused on sentiment analysis on Twitter streams. We have introduced new algorithms for active training-set formation, which we denote as Pareto-Efficient Selective Sampling (PESS) and Kaldor-Hicks Selective Sample (KHSS). The proposed algorithms provide the resulting classifier with memorability and adaptiveness. We formalized the selective sampling process as a multi-objective optimization procedure, which finds a proper balance between adaptiveness and memorability. Adaptiveness is assessed by computing the distance in time and space between the target message and the candidate ones. Also, candidate messages are randomly shuffled, thus providing memorability to the resulting classifier. The message utility space is composed by such dimensions, and we compute the Pareto Frontier in this space in order to pick up messages satisfying the Pareto improvement condition, finding a proper balance between adaptiveness and memorability. The Kaldor-Hicks criterion enables memorability to compensate adaptiveness, or vice-versa. A systematic evaluation involving recent events demonstrated the effectiveness of our algorithms.

As future work, we intend to extend our strategies for algorithms that do not depend on manual labeling.

6. ACKNOWLEDGMENTS

Adriano Veloso, Adriano Pereira, Wagner Meira Jr., and Renato Ferreira would like to acknowledge grants from CNPq, CAPES, Fapemig, Finep, and InWeb – the Brazilian National Institute of Science and Technology for the Web. Srinivasan Parthasarathy would like to acknowledge NSF grant IIS 1111118 and a Google research award. Roberto Oliveira Jr. would like to acknowledge that some aspects of this work was conducted while he was a visiting researcher in Srinivasan Parthasarathy's lab at Ohio State University.

7. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216. ACM, 1993.

[2] R. Baeza-Yates and B. R-Neto. *Modern Information Retrieval*. Addison-Wesley-Longman, 1999.

[3] R. Bayardo, B. Goethals, and M. Zaki, editors. *Workshop on Frequent Itemset Mining Implementations*, volume 126, 2004.

[4] A. Bifet and E. Frank. Sentiment knowledge discovery in twitter streaming data. In *Disc. Science*, pages 1–15, 2010.

[5] A. Bifet, E. Frank, G. Holmes, and B. Pfahringer. Ensembles of restricted hoeffding trees. *TIST*, 3(2):30:1–30:20, 2012.

[6] A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *SDM*, 2007.

[7] A. Bifet and R. Gavaldà. Adaptive learning from evolving data streams. In *IDA*, pages 249–260, 2009.

[8] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive online analysis. *JMLR*, 11:1601–1604, 2010.

[9] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank. Fast perceptron decision tree learning from evolving data streams. In *PAKDD*, pages 299–310, 2010.

[10] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavaldà. Detecting sentiment change in twitter streaming data. *JMLR*, 17:5–11, 2011.

[11] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

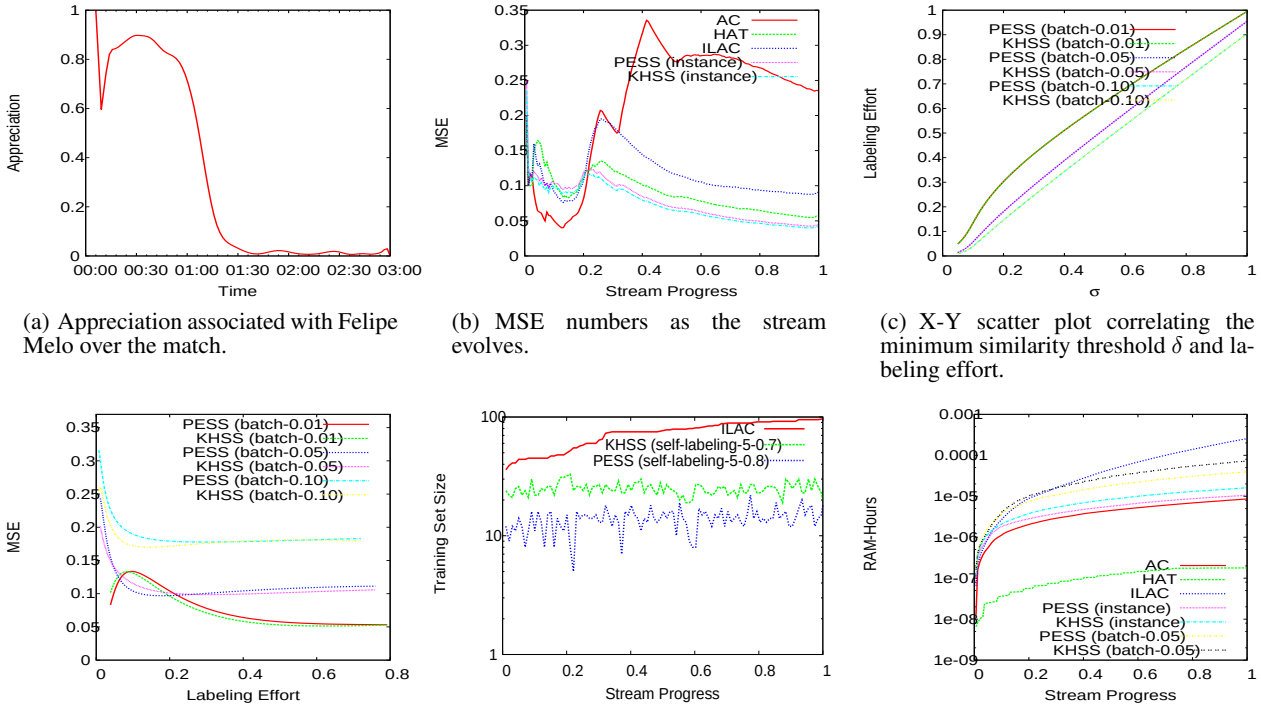
[12] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and regression trees. *Wadsworth Intl.*, 1984.

[13] J. Chipman. Compensation principle. In S. N. Durlauf and L. E. Blume, editors, *The New Palgrave Dictionary of Economics*. Palgrave Macmillan, 2008.

[14] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[15] R. Durstenfeld. Algorithm 235: Random permutation. *Commun. ACM*, 7(7):420, 1964.

[16] L. Feng, F. Chen, and Y. Yao. A concept similarity based data stream classification model. *Journal of Information & Computational Science*, 10(4):949–957, 2013.



(a) Appreciation associated with Felipe Melo over the match.

(b) MSE numbers as the stream evolves.

(c) X-Y scatter plot correlating the minimum similarity threshold δ and labeling effort.

(d) X-Y scatter plot correlating labeling effort and MSE numbers.

(e) Size of the training sets as the stream evolves.

(f) RAM-Hours as the stream evolves.

Figure 5: The Brazilian Defeat. Tweets are in Portuguese.

[17] J. Gama, R. S. ao, and P. Rodrigues. Issues in evaluation of stream learning algorithms. In *SIGKDD*, page 329, 2009.

[18] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining Knowledge Discovery*, 8(1):53–87, 2004.

[19] J. Hicks. The foundations of welfare economics. *The Economic Journal*, 49(196):696–712, 1939.

[20] R. Hof. Real-time advertising has arrived, thanks to Oreo and The Super Bowl, April 2013. `www.forbes.com/`.

[21] C. Jin, K. Yi, L. Chen, J. Yu, and X. Lin. Sliding-window top-k queries on uncertain streams. *VLDB J.*, 19(3):411–435, 2010.

[22] N. Kaldor. Welfare propositions in economics and interpersonal comparisons of utility. *The Economic Journal*, 49(195):549–552, 1939.

[23] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intell. Data Anal.*, 8(3), 2004.

[24] I. Koychev. Gradual forgetting for adaptation to concept drift. In *ECAI*, pages 101–106, 2000.

[25] M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *ICDM*, pages 929–934, 2008.

[26] M. Moreira, J. dos Santos, and A. Veloso. Learning to rank similar apparel styles with economically-efficient rule-based active learning. In *ICMR*, pages 361–369, 2014.

[27] M. N. nez, R. Fidalgo, and R. Morales. Learning in environments with unknown dynamics: Towards more robust concept learners. *JMLR*, 8, 2007.

[28] F. Palda. *Pareto's Republic and the new Science of Peace*. Cooper-Wolfling, 2011.

[29] M. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *RecSys*, pages 19–26, 2012.

[30] I. Santana, J. Gomide, A. Veloso, W. M. Jr., and R. Ferreira. Effective sentiment stream analysis with self-augmenting training and demand-driven projection. In *SIGIR*, pages 475–484. ACM, 2011.

[31] D. Torres, J. Ruiz, and Y. Sarabia. Classification model for data streams based on similarity. In *IEA*, pages 1–9, 2011.

[32] A. Veloso, W. M. Jr., and M. Zaki. Lazy associative classification. In *ICDM*, pages 645–654, 2006.

[33] A. Veloso, W. Meira Jr., M. Gonçalves, H. de Almeida, and M. Zaki. Calibrated lazy associative classification. *Inf. Sci.*, 181(13):2656–2670, 2011.

[34] A. Veloso, M. Otey, S. Parthasarathy, and W. Meira Jr. Parallel and distributed frequent itemset mining on dynamic datasets. In *HiPC*, pages 184–193, 2003.

[35] I. Žliobaitė. Learning under concept drift: an overview. *CoRR*, abs/1010.4784, 2010.

[36] I. Žliobaitė, A. Bifet, G. Holmes, and B. Pfahringer. MOA concept drift active learning strategies for streaming data. *JMLR*, 17:48–55, 2011.

[37] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with evolving streaming data. In *Machine Learning and Knowledge Discovery in Databases*, volume 6913, pages 597–612. 2011.

[38] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *IEEE Trans. on Neural Networks and Learning Systems*, PP(99):1–1, 2013.

[39] M. Zaki and K. Gouda. Fast vertical mining using diffsets. In *SIGKDD*, pages 326–335, 2003.

[40] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *SIGKDD*, pages 283–286, 1997.

[41] X. Zhu, P. Zhang, X. Lin, and Y. Shi. Active learning from stream data using optimal weight classifier ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(6):1607–1621, 2010.

[42] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *KDD*, pages 336–345, 2003.