Towards Classification of Social Streams

Min-Hsuan Tsai* Charu C. Aggarwal† Thomas S. Huang‡

Abstract

Social streams have become very popular in recent years because of the increasing popularity of social media sites such as *Twitter*, and *Facebook*. Such social media sites create huge streams of data, which can be leveraged for a wide variety of applications. In this paper, we will focus on the classification problem for social streams. Unfortunately, such streams are extremely noisy, and contain large volumes of information, with information about *network linkages* between the participants exchanging messages. This is additional social information, associated with the text stream, which can be very helpful for classification. We combine an LSH method with an incremental SVM model in order to design an effective and efficient social context-sensitive streaming classifier for this scenario. The LSH model is used for learning the social context, and the SVM model is used for more effective classification within this context. We will present experimental results, which show the effectiveness of our techniques over a wide variety of other methods.

Keywords: Text Mining; Data Streams; Social Streams

1 Introduction

Social streams have become ubiquitous in recent years because of the enabled of a wide variety of user-centered applications in social networks, which can create continuous streams of text data. Some specific examples are as follows:

- In social networks, users continuously communicate with one another with the use of text messages. This results in massive volumes of text streams which can be leveraged for a variety of mining and search purposes in the social network. This is because the text messages are reflective of user interests.
- In chat and email networks, users constantly send messages to one another, which can cause large streams of data. The volume of such streams can be extremely large, because of the very large number of communicating users.
- Many media-sharing sites enable users to make interactive comments about media content. Such data can also be considered social streams.

Classification algorithms need to be implemented very *efficiently* to work with *one pass of the data*, because of the massive volume of the data which must be processed in online fashion. Therefore, the training needs to be performed in a fast incremental fashion in order to ensure the best results. Typically, the stream may experience *concept drift*, because of which the key patterns in the underlying stream may change over time. This means that the training models may become outdated over time, and may constantly need to be updated, in order to ensure accurate results for classification. Therefore, social stream models need to be continuously updated over time. A major issue with social streams is that they are associated with linkages between participants, who constantly exchange messages. This information is very useful for the knowledge discovery process.

Each message in the social stream is associated with a sender and a receiver, which corresponds to an edge in the underlying social network. Typically, the content exchanged in a message is highly related to the location of this edge in the underlying social network. In this paper, we will examine the problem of classification in social streams, in which additional information about the social context of the underlying documents is available. We note that the presence of social information provides a considerable amount of feedback to the classifier, because it provides an understanding of how the different social actors are related to the different classes. Since the nature of the social actors may be closely related to the label, this information can presumably help in the classification process. At the same time, it also creates an additional challenge for the classification process, because the social information needs to be used effectively for classification.

Another additional challenge with social streams is that they are *typically very noisy*, and often contain many incorrectly labeled instances. For example, in *Twitter*, the hash tags provided by the users can label some of the documents for a particular topic, but this information is often quite noisy. A user may incorporate a hash tag for a tweet, for which the content may not necessarily belong to that topic. At the same time, a tweet which does not contain a specific hash tag may also belong to a relevant topic. Thus, social streams typically fall in the category of very *noisy* classification problems, in which it is very challenging to relate the class behavior of the test instance to the content of the social

*Google, Mountain View, CA, mhtsai@google.com
†IBM Research, Yorktown Heights, NY, charu@us.ibm.com
‡UIUC, Urbana, IL, huang@ifp.uiuc.edu

stream. When combined with the fact that social streams need to be classified very fast with incremental and online methods, this creates a very challenging scenario for the classification process.

We note that the social senders and recipients of a given message provide an effective social *context* for a particular topic. The classification behavior of a given text document is likely to be somewhat different within different contexts. For example, a message which is sent from a user who has frequently exchanged sports related messages with a group of other similar users, is more likely to be about the same topics. At the same time, it is extremely important to not overfit into a particular topic with the use of social context. In order to achieve this goal, we will use a *locality sensitive hashing (LSH)* technique to efficiently partition the incoming documents based on the context implied by their position in the social network. Different classifiers are constructed for different contexts in order to maximize the classification accuracy. Multiple contextual partitionings are constructed with different hash functions in order to maximize accuracy, and avoid over-fitting within any particular context.

This paper is organized as follows. The remainder of this section discusses the related work. In section 2, we will present an algorithm for classification of social streams. The experiments are presented in section 3. Section 4 contains the conclusions and summary.

1.1 Related Work. The problem of text stream classification arises in the context of a number of different IR tasks, such as *news filtering* [15], and email spam filtering [7]. Text streams have been widely studied, both in the context of the problems of clustering and classification [6, 12, 14, 29]. The problem of classification of data streams has been widely studied by the data mining community in the context of different kinds of data [1, 5, 23, 26]. Various problems, such as clustering, classification, and event detection have been studied recently in the context of social streams [2, 3, 4, 11, 20].

The methods discussed in [13, 18] studies methods for classifying text streams in which the classification model may evolve over time. Some recent work [8, 22] discusses methods for incremental classification, although these methods do not leverage all the available information in a social stream. Another related work is that of one-class classification of text streams [27], in which only training data for the positive class is available, but there is no training data available for the negative class. This is quite common in many real applications in which it easy to find representative documents for a particular topic, but it is hard to find the representative documents in order to model the background collection. The method works by designing an ensemble of classifiers in which some of the classifiers corresponds to a recent model, whereas others correspond to a long-term model.

A number of *neural network methods* [19, 24] have also been adapted to the stream scenario. In these methods, the classifier starts off by setting all the weights in the neural network to the same value. The incoming training example is classified with the neural network. In the event that the result of the classification process is correct, then the weights are not modified. On the other hand, if the classification is incorrect, then the weights for the terms are either increased or decreased depending upon which class the training example belongs to. A Bayesian method for classification of text streams is discussed in [9]. The method in [9] constructs a Bayesian model of the text which can be used for online classification. The key components of this approach are the design of a Bayesian online perceptron and a Bayesian online Gaussian process, which can be used effectively for online learning.

2 Socially Sensitive Stream Classification

In this section, we will present our algorithm for socially sensitive stream classification. Before discussing the algorithm in detail, we will introduce some notations and definitions, as well as the problem setting. We assume that the records which are received by the data stream are denoted by $\overline{T_1} \ldots \overline{T_i} \ldots$. Each record $\overline{T_i}$ is essentially a text document, which can be used for classification purposes. We assume the most difficult case of a massively multi-label scenario, in which we are tracking a set of m binary labels on the documents simultaneously. These m binary labels could correspond to the m different topics in the social stream that we are trying to track. For example, in a social stream the m different labels could correspond to *sports*, *current events*, *politics* etc. These m different labels are not necessarily mutually exclusive, and a subset of them may be simultaneously present in the data. For example, a post which belongs to the category of "politics" could also belong to the category of "current events". The value of the m different binary labels associated with the ith document are denoted by $l_i^1 \ldots l_i^m$. Each of these labels is drawn from $\{0, 1\}$ depending upon whether or not the content belongs to that category. Such multi-label scenarios are quite common in social streams, and it is necessary to track the different labels simultaneously in order to perform the most effective classification. We assume that the multi-label vector for a given training data point is denoted by $(l_i^1 \ldots l_i^m)$.

In addition, the originator of the content is drawn from a set of N social participants which are indexed by $\{1 \ldots N\}$. Therefore, we assume that the following additional pieces of information about the text document are available:

- The origination node $q_i \in \{1 \ldots N\}$ which is the sender of the message T_i to other nodes.
- The object S_i contains a *set of one or more* receiver nodes R_i, which correspond to all recipients of the

message T_i from node q_i. Thus, the message T_i is sent from the origination node q_i to each node $p \in R_i$.

Clearly, the author and recipients of a social post provide very important information about the *social context* of the post. Therefore, it is critical to learn how we can leverage this context for a more effective classification process. In this paper, we will use a locality sensitive hashing approach in order to partition the data space into a set of social contexts, so that more effective classifiers can be constructed within these contexts.

2.1 Contextual Training of Social Stream. In this section, we will present the approach for contextual training of the social stream. The incoming stream is partitioned into two different groups in multiple ways, depending upon the social context of the users. Each of the models can be considered a *socially local* model, which is designed to be more efficient than a purely global model. Therefore, for an incoming data point, we perform the following operations, depending upon whether it is a training data point, or test data point:

- If it is a training data point, then the training model is incrementally updated during the process, with the addition of the new data point. We use an SVM classifier, which is updated incrementally with the addition of the new data point. This operation is performed for each of the relevant pairs of socially locally models. For a given pair, only one of the socially local models needs to be updated during the process.
- If it is a test data point, we classify the test instance with each of its relevant socially local models. The average class label from these different test instances are reported.

Therefore, our description will focus on the following aspects:

- The design of the LSH model for data partitioning.
- The design of an online approach for the training of each socially local model.
- The process for actual categorization of a given test instance.

The online algorithm dynamically partitions the solution space on the basis of the class behavior of the social patterns in the underlying data. This provides an effective social context in which the classifier can be more effective. In order to examine the relative behavior of the classification patterns, the algorithm dynamically maintains the mean statistics of the number of arrivals of each class type. Specifically, at the time of the arrival of the n-th (training) text document, the algorithm dynamically maintains the value of $(\sum_{i=1}^{n} l_i^1 \ldots \sum_{i=1}^{n} l_i^m)$. This can be easily achieved by simple additive operations over the incoming training records. Correspondingly, we can divide the afore-mentioned vector by n in order to determine the mean number of arrivals of each class, which is denoted by $(\mu^1 \ldots \mu^m)$.

At the same time, we maintain these statistics for the messages posted by *each actor* in the underlying data stream. Thus, for the i-th actor in $\{1 \ldots N\}$, and the k-th class in $\{1 \ldots m\}$, it is assumed that the mean value is denoted by $\mu^k(i)$. These statistics are maintained in exactly the same way as the global statistics. We note that the value of $\nu^k(i) = \mu^k(i) - \mu^k$ may be either positive or negative, depending upon whether the i-th actor has above-average or below-average involvement with class k, and its weighted mean value over all actors is equal to 0. The *class characteristic vector* for any particular actor i is denoted $\overline{\nu(i)} = (\nu^1(i) \ldots \nu^m(i))$. This vector provides an idea of the profile of the actor, as it relates to the different classes. The concept of a class characteristic vector can be generalized to that of a particular post $\overline{T_j}$ with a sender q_j and recipients R_j.

DEFINITION 1. (CLASS CHARACTERISTIC VECTOR) *The class characteristic $\overline{\gamma(q_j, R_j)}$ vector of a post $\overline{T_j}$ with sender q_j and recipient set R_j, is the average of the sender characteristic vector and the recipient characteristic vector. In other words, we have:*

$$\overline{\gamma(q_j, R_j)} = \frac{1}{2} \left(\overline{\nu(q_j)} + \frac{1}{|R_j|} \cdot \sum_{p \in R_j} \overline{\nu(p)} \right) \tag{2.1}$$

We perform the LSH-based partitioning in r different ways, with the use of r random projection vectors denoted by $\overline{\eta_1} \ldots \overline{\eta_r}$. The value of r is typically much less than m. Consider a training record T_j with sender q_j and recipient set R_j. For the i-th random projection (i in $\{1 \ldots r\}$), the training record T_j is assigned to one of two partitions, depending on the value of $sgn(\overline{\eta_i} \cdot \overline{\gamma(q_j, R_j)}) \in \{-1, 1\}$. Thus, this creates two partitions with different social contexts, within which the training can be performed more effectively. We denote the two partitions for the ith random projection by $\mathcal{P}_i^{+1}$ and $\mathcal{P}_i^{-1}$. A separate set of m training models is constructed for each of the $2 \cdot r$ partitions, one for each of the m different binary labels in the data. Thus, a total of $2 \cdot r \cdot m$ models are constructed and maintained for the training data stream. For each incoming training record, we first determine which partition it might belong to with the use of the LSH function. Once, this has been determined, we update the SVM model with the use of the incoming data point. Next, we describe the process of updating an SVM model with the addition of a training data point.

2.2 Incremental SVM learning with decaying sample weights. In this section, we introduce an incremental binary SVM learning algorithm that takes the decay factors on the seen samples over time. We first define a time-sensitive weight function, denoted by $f(t) = 2^{-dt}$. This is also referred to as the fading function, which regulates the importance of different stream samples over time. This is the exponential decay function, which is used commonly in streaming scenario. The parameter d is the decay rate of the samples, which is also the inverse of the half life of the data stream. The same fading parameter d is used across all samples.

Next, we will describe the binary SVM model for a particular combination of the j-th binary class label and partition identifier. We will describe a single model for one of combination of multi-labels and partitions, though all the SVM models for the different multi-labels and partitions need to be maintained simultaneously. Let us consider the case, where we have n previously seen documents, which are denoted by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathbb{R}^p$ is the set of features of the i-th document, and $y_i \in \{-1, +1\}$ is the label indicating whether it belongs to j-th class. Let the i-th document of this partition correspond to the kth document $\overline{T_k}$ in the original data stream that has been assigned to this partition based on LSH. We note that, we can convert the binary $\{0, 1\}$ label to $\{-1, 1\}$ with the use of the transformation $y_i = 2 \cdot l_k^j - 1$. This representation is more convenient for mathematical modeling and analysis. Then, we will formulate a binary SVM classification problem with soft margin. For separation, we use the hyperplane defined by $w^T x - b = \pm 1$, along with nonnegative slack variables ξ_i for data point i, which allow for some degree of misclassification of the i-th data point. The larger the value of ξ_i, the greater the misclassification of the i-th instance. The minimization of $\frac{1}{2}\|w\|^2$ maximizes the margin between the two classes, and the slack variables are penalized in the objective function.

$$\begin{aligned} (2.2) \qquad \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \\ \text{subject to} \quad & y_i(w^T x_i - b) \geq 1 - \xi_i,\ \xi_i \geq 0 \end{aligned}$$

We note that $w^T x - b = 0$ is the maximum-margin hyperplane for making the binary decision about the class label. The constant C represents the relative importance of allowing some misclassification (through slack variables), and the maximization of the margin between the two hyperplanes. In order to solve this problem effectively, we can further rewrite (2.3) in the Lagrangian formulation as a convex quadratic problem, with the introduction of the lagrange multiplier α:

$$(2.3) \qquad \min_{0 \leq \alpha_i \leq C} W = \frac{1}{2}\alpha^T Q \alpha - e^T \alpha + b\, y^T \alpha$$

In order to incorporate decay factors into the formulation, we rewrite (2.3) with sample weight $f_i(t)$ for sample i:

$$\begin{aligned} (2.4) \qquad \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + \frac{C}{\sum_i f_i(t)}\sum_i f_i(t)\xi_i \\ \text{subject to} \quad & y_i(w^T x_i - b) \geq 1 - \xi_i,\ \xi_i \geq 0 \end{aligned}$$

Note that the incorporation of the decay weights will impact the optimal SVM hyperplane found by the approach. The dual formulation of the afore-mentioned optimization problem can be derived as follows:

$$\begin{aligned} (2.5) \qquad \min_{\alpha} \quad & W = \frac{1}{2}\alpha^T Q \alpha - e^T \alpha + b\, y^T \alpha \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C s_i(t) \end{aligned}$$

In the afore-mentioned formulation, Q is the kernel matrix whose (i, j)-th entry is $Q_{ij} = y_i y_j K(x_i, x_j)$ and $s_i(t) = \frac{f_i(t)}{\sum_i f_i(t)}$ is the *normalized sample weights* for the i-th sample. The Karush-Kuhn-Tucker (KKT) conditions of the optimization problem above may be defined as follows:

$$(2.6) \qquad g_i = \frac{\partial W}{\partial \alpha_i} = Q_{i\cdot}\alpha + b y_i - 1$$

$$(2.7) \qquad \frac{\partial W}{\partial b} = \sum_j y_j \alpha_j = 0$$

We note that the value of g_i and the corresponding coefficients $\{\alpha_i, b\}$ can be used to partition the samples into three categories: $\mathcal{S}$ as the set of support vectors, $\mathcal{E}$ as the set of error support vectors which are located outside the margin, and $\mathcal{R}$ as the remaining samples. In particular, the conditions that the sample falls into each set are:

$$\begin{aligned} (2.8) \qquad \mathcal{E} &= \{i | g_i \leq 0;\ \alpha_i = C s_i(t)\} \\ \mathcal{S} &= \{i | g_i = 0;\ 0 < \alpha_i < C s_i(t)\} \\ \mathcal{R} &= \{i | g_i \geq 0;\ \alpha_i = 0\} \end{aligned}$$

We then follow [8] to introduce the incremental weighted SVM learning that considers the incremental steps to maintain the equilibrium of the KKT conditions with small margin vector coefficients change. In particular, we write the incremental KKT conditions as follows:

$$(2.9) \qquad \Delta g_i = Q_{ic}\Delta\alpha_c + \sum_{j \in S} Q_{ij}\Delta\alpha_j + y_i \Delta b$$

$$(2.10) \qquad 0 = y_c \Delta\alpha_c + \sum_{j \in S} y_j \Delta\alpha_j$$

Then, the increments of $\Delta\alpha_i$ and Δb can be written in terms of $\Delta\alpha_c$:

$$(2.11) \qquad (\Delta b, \Delta\alpha_{s_1}, \ldots, \Delta\alpha_{s_n})^T = -\Delta\alpha_c \mathbf{P}^{-1}\mathbf{r}$$

Algorithm 1 Incremental SVM learning with decaying sample weights

input $(x_c,\ y_c)$ of the new sample T_c, previous seen dataset $\mathcal{D}^{(t_1)}$, sample weights $s_i^{(t_1)}$, previous SVM model $\alpha_i^{(t_1)}$,$b^{(t_1)}$,$\mathcal{S}^{(t_1)}$, $\mathcal{E}^{(t_1)}$, $\mathcal{R}^{(t_1)}$

1. Initialize normalized sample weight rate $\beta_i \equiv s_i^{(t)}/s_i^{(t_1)} = \frac{1}{\eta} f(t-t_1)$, where $\eta = f(t-t_1)\sum_i s_i^{(t_1)}+1$,
2. Update $\alpha_i \leftarrow \beta_i \alpha_i$
3. Set $\alpha_c = 0$, compute g_c with α_c using (2.6)

repeat

if $g_c > 0$ **then**

4. $\mathcal{R}^{(t)} = \mathcal{R}^{(t_1)} \cup \{T_c\}$; terminate;

else

5. find $\Delta\alpha_c$ that satisfies one of the conditions:
   (i) $g_c = 0$, then $\mathcal{S}^{(t)} = \mathcal{S}^{(t_1)} \cup \{T_c\}$, terminate
   (ii) $\alpha_c = C$, then $\mathcal{E}^{(t)} = \mathcal{E}^{(t_1)} \cup \{T_c\}$, terminate
   (iii) Make any of previous seen samples migrate across $\mathcal{S}^{(t_1)}$, $\mathcal{E}^{(t_1)}$, $\mathcal{R}^{(t_1)}$.

end if

6. Compute $\Delta\alpha_i$ and Δg_i from $\Delta\alpha_c$ using (2.11) and (2.14)
7. $\alpha_i \leftarrow \alpha_i + \Delta\alpha_i$ and $g_i \leftarrow g_i + \Delta g_i$

until Termination condition is met.

output $\alpha_i^{(t)}$, $b^{(t)}$, $\mathcal{D}^{(t)} = \mathcal{D}^{(t_1)} \cup \{T_c\}$, $\mathcal{S}^{(t)}$, $\mathcal{E}^{(t)}$, $\mathcal{R}^{(t)}$

where

$$\mathbf{P} = \begin{pmatrix} 0 & y_{s_1} & \cdots & y_{s_n} \\ y_{s_1} & Q_{s_1 s_1} & \cdots & Q_{s_1 s_n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{s_n} & Q_{s_n s_1} & \cdots & Q_{s_n s_n} \end{pmatrix} \tag{2.12}$$

and

$$\mathbf{r} = \begin{pmatrix} y_c \\ Q_{s_1 c} \\ \vdots \\ Q_{s_n c} \end{pmatrix} \tag{2.13}$$

Therefore (2.9) becomes

$$\Delta g_i = \left(Q_{ic} - [y_i,\ Q_{i\cdot}^T]\mathbf{P}^{-1}\mathbf{r}\right)\Delta\alpha_c \tag{2.14}$$

for $\forall i \in \mathcal{S}$

It is worth noting that the above model assumes that every sample shares the same decay factor d. Under this assumption, the model won't change over time even with weight decaying samples, due to the use of normalized sample weights. For example, there are three samples with weight {1,1,0.5}, or normalized weight {0.4,0.4,0.2}. After some time their weights decay all by 0.3 (as they have the same d) and become {0.3,0.3,0.15}. However, their normalized sample weights keep the same {0.4,0.4,0.2}, thus the model won't change. So the model only changes when new samples added, and it will reflect the decaying time at that time. For example, when the new sample added the decaying factor is 0.7 or 0.2 gives the sample weights {0.7,0.7,0.35,1} or {0.2,0.2,0.1,1} .

This incremental update process allows us to continuously maintain the incremental model for each partition and binary label. We note that the models for all the different class labels and corresponding LSH partitions are maintained simultaneously. We note that the updates can be performed very efficiently, and therefore, the model can be maintained in real time.

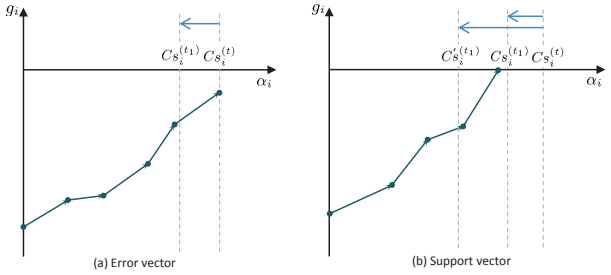


Figure 1: Decaying effect on the support and error vectors.

2.3 Decaying effect on the model. In this section we will discuss how the decaying factor of the samples will affect the current model. First, we note that the decaying effect would take place only when a new sample is added, because we assume that all the previously seen samples decay at the same rate (thus the normalized sample weights s_i keep the same). In the case when there is an unsee sample coming in, all the normalized sample weights of the previously seen samples will reduce (at different degrees), and the new model needs to take into account this change. We illustrate the possible changes of the model from the previously seen samples aspect in Fig. 1. In the case of the previously seen sample being an error vector, shown in Fig. 1(a), the reduce of the normalized sample weight would not change the state of that sample being an error vector since g_i cannot reach 0. In the case of the support vectors in the old model, shown in Fig. 1(b), there will be two possibilities: first is that the reduction of the normalized sample weight is small ($Cs_i^{(t)}$ to $Cs_i^{(t_1)}$) such that the g_i is still 0 and the sample remains as the support vector. On the other hand, if the reduction is large ($Cs_i^{(t)}$ to $C's_i^{(t_1)}$) and the g_i of that sample is no longer 0, the sample will become an error vector in the new model.

2.4 Test Instance Classification. The training model discussed in the previous section can be used for effective classification of test instances. We socially-sensitive local partition can be used for classification. The first step is to assign the test instance to one of the partitions with the use of the same LSH approach. Thus, for a test instance T_k with sender q_k and recipients R_k, we compute the values of $sgn(\overline{\eta_i} \cdot \overline{\gamma(q_k, R_k)}) \in \{-1, 1\}$ in order to assign a partition to the test instance for each $i \in \{1 \ldots r\}$. We note that the vectors η_i are always maintained throughout the training and test process. For each of these partitions, the relevant SVM classifier is used for the purposes of classification. The different ways of partitioning the data may provide different class labels for the test instance. The largest occurring class label from the different votes is reported as the relevant classification for the test instance.

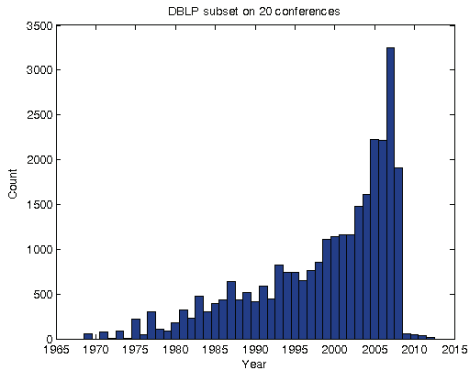


Figure 2: Number of publications per year in DBLP dataset

3 Experimental Results

In this section, we evaluate the effectiveness and efficiency of our proposed algorithm, and compare it with a baseline method which uses pure text classification with the use of SVM. The approach was tested for both effectiveness and efficiency. The results were tested on two publicly available data sets.

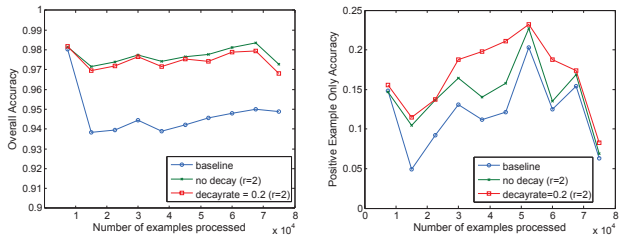
3.1 Data Sets. Two real data sets were used for the purpose of experimental evaluation. Both data sets had a temporal component, which was used to convert them into a social stream. The data sets used are described below.

- **Enron Email Dataset.** The raw Enron email corpus[1] contains more than 500,000 messages belonging to 158 senior management personnel of the Enron enterprise. We used a cleaned subset of the corpus consisting around 75,000 emails between 7,500 users and arranged them in a chronological order based on the received time of the email. The messages were split into 10 sessions, and for each session we further partition the messages into a training set and a testing set in a 60:40 ratio. As a result, the testing set contains around 30,000 messages. In order to generate the labels, we manually selected the following 19 event labels: *meeting, agreement, gas, energy, power, report, update, request, conference, letter, deal, credit, california, trading, contract, project, presentation, houston, announcement.* We used the subject of the message to determine whether the message belonged to any of these 19 events. This resulted in a multi-label setting for each message. In the *Enron* data, we represent each email with bag-of-words features via a dictionary of 10,000 terms which are the most frequent terms appears in the corpus (with stemming and stop words removes). We used the standard class-characteristic vector in this case.

- **DBLP Dataset.** The DBLP database is a collection of bibliographic information on major computer science publications. There are more than 10 attributes available from the original DBLP database, and we only retrieve four attributes in our experiment, namely, paper title and abstract, authors, venue, and publish date. Also, we only use a subset of the DBLP papers with four data mining related areas (database, data mining, information retrieval and artificial intelligence) which were published between 1969 and 2012. There are in total 28,569 documents (209 without published year), 28,702 authors and 20 conferences. Fig. 2 shows the number of publications per year of this data set. In this data set, each paper is represented by a bag of words that appeared in the abstract and title of the paper. The dictionary used has 11,771 unique terms. Note that, instead of using the *mean* characteristic vector of all the authors of the paper, we use a *max* pooling strategy to obtain the class characteristic vector. That is, we use the characteristic vector from the author with highest number of publication as the class characteristic vector for that paper.

3.2 Baseline Classifier. We compared our approach against the conventional online *SVM* classifier for text data, where the incremental and decremental SVM learning from Cauwenberghs and Poggio [8] is adopted. The model uses an approach of training with incoming samples by retaining only the support vectors. The support vectors, which are samples on or within the margin, do not fade out. The SVM model is incrementally maintained with these support vectors.

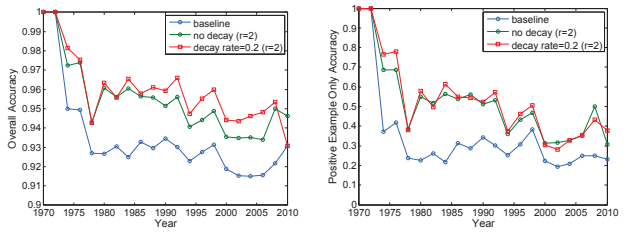
[1] http://www.cs.cmu.edu/~enron/



(a) (b)

Figure 3: Classification accuracy for *Enron* dataset

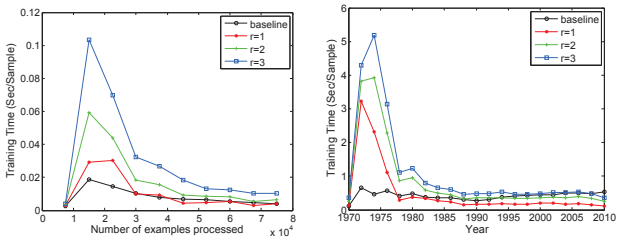
3.3 Experimental Setting. All results reported were performed on machines each equipped with two Intel Quad Core Xeon 2.4GHz CPUs and 16 GB memory. In the streaming scenario, we consider the samples in a chronological order based on their arrival time (i.e., the received time for the *Enron* data set and the published year for the DBLP data set). In the case of the *DBLP* data set, ties are broken randomly. We then split the samples into sessions. Each session contains both a training and testing part where the training part always comes before the testing part in terms of arrival time. The models are trained in an online fashion and continuously updated over time. In addition, each testing sample was converted into a training sample *after* its classification had already been performed, and then the ground truth was added to the test sample. In our proposed approach there are two important parameters: the number of LSH hyperplanes (r) and the decay rate (d). We will study the impact of each of these parameters on the classification accuracy in detail.



(a) (b)

Figure 4: Classification accuracy for *DBLP* dataset

3.4 Effectiveness Results. The performance results are reported in terms of classification accuracy. We present classification accuracy with progression of stream for the *Enron* and *DBLP* data sets are illustrated in Fig. 3 and Fig. 4 respectively, where X-axis is the stream progression in terms of either the session processed or the year processed and Y-axis is the classification accuracy in the last session. Note that the classification accuracy are measured both in terms of *Overall* accuracy and the *Positive Examples Only*, as the amount of positive and negative examples are highly imbalanced hence the overall accuracy sometimes does not reflect the data set behavior very well. Furthermore, the negative sample is usually contaminated with members of different classes, and therefore, the accuracy on the negative class is quite noisy. Our proposed socially sensitive approaches (with or without decay factor) outperform the baseline SVM classifier by about 25% and 50% in terms of positive examples only accuracy for both the *Enron* and *DBLP* data streams. The variation in the classification accuracy over the course of the stream was because of the local distributions of the labels.

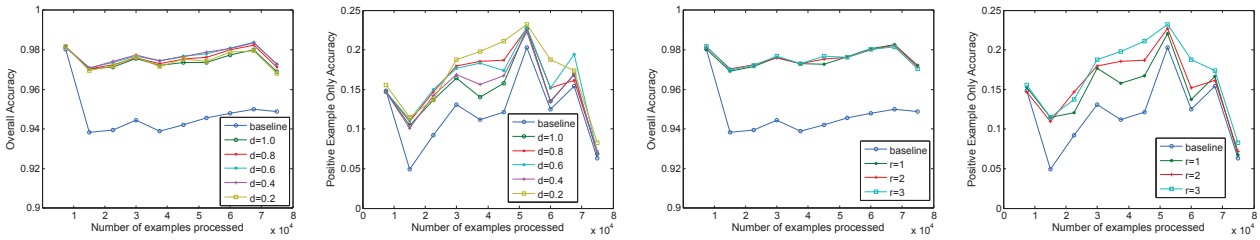


(a) Enron data set (b) DBLP data set

Figure 7: Training efficiency with various number of LSH hyperplanes. For the Enron data set, the decay rate is 0.2, while for the DBLP data set, the decay rate is 0.8.

Fig. 5 and Fig. 6 further demonstrate the variation of classification accuracy with effect of contextual partitioning and the decay factor on each data set. As can be seen from the figures, the performance increases with the number of LSH partitioning hyperplanes as more hyperplanes give better and finer partitioning on the social contexts. However, finer partitioning means more models need to be trained so the training time will also increase. The behavior of the streams over different decay rates is somewhat different, and depends upon the inherent concept drift in the underlying stream. This will be discussed in more detail in the sensitivity analysis section. Next, we will present efficiency results.

3.5 Efficiency Results. On the other hand, Fig. 7 shows the efficiency with progression of the data stream. Here the efficiency is defined as the training update time for each data point. It can be seen that at the beginning, the proposed method needs much more processing time than the baseline method. This is because of the special nature of the initial training period. However, the processing time drops quickly when some of the old examples decay out, and the number of effective examples (with modest values of the decay weights) becomes steady. The processing time of our approach is almost linear in the number of LSH hyperplanes. For smaller number of hyperplanes, our approach is faster in terms of the steady state speed after the initial training period. When the number of the hyperplanes increases to 3, the processing time for each example is slightly slower than the baseline



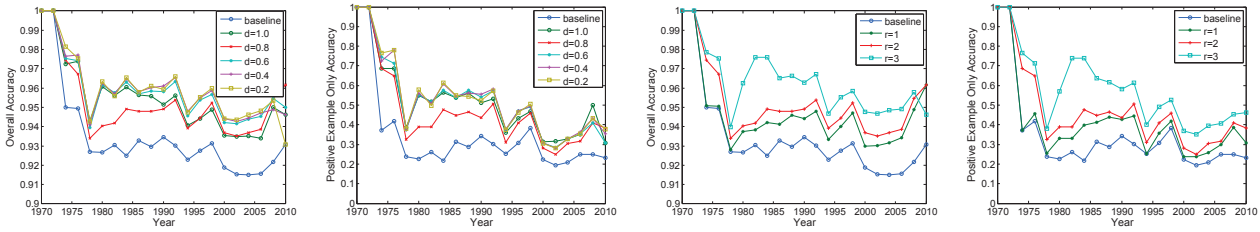
(a) Overall accuracy with various decay rates (r=2)

(b) Positive examples only accuracy with various decay rates (r=2)

(c) Overall accuracy with various values of r ($d = 0.2$)

(d) Positive examples only accuracy with various values of r ($d = 0.2$)

Figure 5: Classification accuracy based on various parameter settings for *Enron* data set



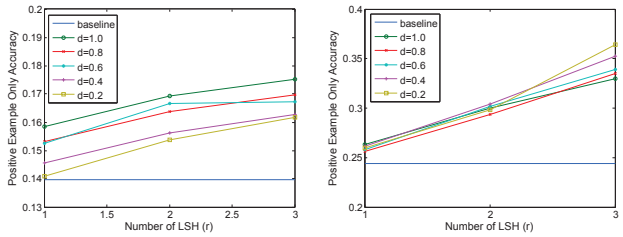
(a) Overall accuracy with various decay rate (r=2)

(b) Positive examples only accuracy with various decay rates ($r = 2$)

(c) Overall accuracy with various values of r ($d = 0.8$)

(d) Positive examples only accuracy with various values of r ($d = 0.8$)

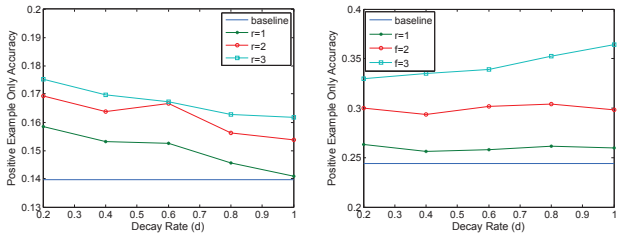
Figure 6: Classification accuracy based on various parameter settings for *DBLP* data set



(a) *Enron* data set

(b) *DBLP* data set

Figure 8: Sensitivity of positive example only classification accuracy on number of LSH hyperplanes (r)



(a) Enron data set

(b) DBLP data set

Figure 9: Sensitivity of positive example only classification accuracy on decay rate (d)

method in terms of the steady state speed. Nevertheless, the processing times were comparable in terms of the steady state processing speeds. It is also important to remember that our approach performs better than the baseline even in cases, where only one hyperplane is used. Therefore, our approach has clear advantages, which evaluated in terms of the tradeoff between accuracy and efficiency.

3.6 Sensitivity Analysis In the experiments, the sensitivity of the approach is tested with respect to the decay rate (d) and number of LSH hyperplanes (r). These parameters are selected from $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ and $\{1, 2, 3\}$ respectively for testing purposes. Fig. 8 and Fig. 9 summarize the sensitivity analysis in terms of positive example only accuracy on these two parameters. When the number of LSH hyperplane (r) increases, the classification accuracy also increases (Fig. 8). It is evident from the result that more social contextual based hyperplanes enable construction of multiple classifiers, each of which can effectively classifies the documents with certain social context. A larger number of hyperplanes corresponds to a greater number of samples, and therefore also leads to higher effectiveness. Therefore, the overall effectiveness would always increase with the number of hyperplanes r.

We also tested the sensitivity of the method with respect to the decay rate (d). In this case, the results are different for the two data sets. It is evident that a decreasing trend exists with the increase of the decay rate on the *Enron* data set. On the other hand, a (slightly) increasing trend with the increase of the decay rate on the *DBLP* data set (Fig. 9) is evident. This implies that the decay rate (d) is more data dependant. For a data set with higher concept drift, a lower decay rate

(d) is preferred. Recall that the decay rate is the inverse of the half life of the data, thus the lower decay rate the faster the data will decay out. Therefore, it is reasonable that the *Enron* data set provides optimal results at a lower decay rate as compared to the *DBLP* data set. At the same time, it is also important to remember that out socially sensitive classification approach is better than the baseline across the entire spectrum of decay values.

4 Conclusions

Social streams have become ubiquitous because of the increasing prevalence of social media sites, which allow user feedback. This is because the communication between participants can be viewed as continuous streams of data, which need to be processed in the context of a variety of applications. This paper presents an approach for incorporating social information into the classification process with the use of an incremental and socially contextual SVM-approach. The results show that the incorporation of social context greatly improves the overall classification accuracy, because it results in classifiers which are appropriate for their specific social context. The experimental results show the effectiveness of the method over state-of-the-art techniques for stream classification without social context.

Acknowledgments

Research was sponsored by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] C. C. Aggarwal. Data Streams: Models and Algorithms, *Springer*, 2007.

[2] C. C. Aggarwal, K. Subbian. Evolutionary Network Analysis: A Survey, *ACM Computing Surveys*, 47(1), 10, July 2014.

[3] C. C. Aggarwal. Mining Text and Social Streams: A Review, *ACM SIGKDD Explorations*, 15(2), pp. 9–19, December 2013.

[4] C. C. Aggarwal, and K. Subbian. Event Detection in Social Streams, *SDM Conference*, 2012.

[5] C. C. Aggarwal, J. Han, J. Wang, P. Yu. On Demand Classification of Data Streams, *KDD Conference*, 2004.

[6] C. C. Aggarwal, P. S. Yu. On Clustering Massive Text and Categorical Data Streams, *Knowledge and Information Systems Journal*, 24(2), pp. 171–196, 2010.

[7] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, C. D. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. *ACM SIGIR Conference*, 2000.

[8] G. Cauwenberghs, T. Poggio. Incremental and Decremental Support Vector Machine Learning, *NIPS Conference*, 2000.

[9] K. Chai, H. Ng, H. Chiu. Bayesian Online Classifiers for Text Classification and Filtering, *ACM SIGIR Conference*, 2002.

[10] K. Crammer, Y. Singer. A New Family of Online Algorithms for category ranking, *ACM SIGIR Conference*, 2002.

[11] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, J. Huang. StreamCube: Hierarchical Spatio-temporal Hashtag Clustering for Event Exploration over the Twitter Stream. *IEEE ICDE Conference*, 2015.

[12] D. Fisher. Knowledge Acquisition via incremental conceptual clustering. *Machine Learning*, 2: pp. 139–172, 1987.

[13] G. P. C. Fung, J. X. Yu, H. Lu. Classifying text streams in the presence of concept drifts. *PAKDD Conference*, 2004.

[14] J. H. Gennari, P. Langley, D. Fisher. Models of incremental concept formation. *Journal of Artificial Intelligence*, 40: pp. 11–61, 1989.

[15] D. Lewis. The TREC-4 filtering track: description and analysis. *Proceedings of TREC-4, 4th Text Retrieval Conference*, pp. 165–180, 1995.

[16] D. Lewis, R. E. Schapire, J. P. Callan, R. Papka. Training algorithms for linear text classifiers. *ACM SIGIR Conference*, 1996.

[17] H. T. Ng, W. B. Goh, K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. */ SIGIR Conference*, 1997.

[18] T. Salles, L. Rocha, G. Pappa, G. Mourao, W. Meira Jr., M. Goncalves. Temporally-aware algorithms for document classification. *ACM SIGIR Conference*, 2010.

[19] H. Schutze, D. Hull, J. Pedersen. A comparison of classifiers and document representations for the routing problem. *ACM SIGIR Conference*, 1995.

[20] K. Subbian, C. Aggarwal, J. Srivastava. Content-centric Flow Mining for Influence Analysis in Social Streams, *ACM CIKM Conference*, 2013.

[21] A. Surendran, S. Sra. Incremental Aspect Models for Mining Document Streams. *PKDD Conference*, 2006.

[22] C.-Y. Tseng and M.-S. Chen. Incremental SVM Model for Spam Detection on Dynamic Email Social Networks, *ICCSE*, 2009.

[23] H. Wang, W. Fan, P. Yu, J. Han, Mining Concept-Drifting Data Streams with Ensemble Classifiers, *KDD Conference*, 2003. .

[24] E. Wiener, J. O. Pedersen, A. S. Weigend. A Neural Network Approach to Topic Spotting. *SDAIR*, pp. 317–332, 1995.

[25] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. T. Archibald, X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, 1999.

[26] K. L. Yu, W. Lam. A new on-line learning algorithm for adaptive text filtering. *ACM CIKM Conference*, 1998.

[27] Y. Zhang, X. Li, M. Orlowska. One Class Classification of Text Streams with Concept Drift, *ICDMW Workshop*, 2008.

[28] Q. Zhao, P. Mitra. Event Detection and Visualization for Social Text Streams, *ICWSM*, 2007.

[29] S. Zhong. Efficient Streaming Text Clustering. *Neural Networks*, Volume 18, Issue 5-6, 2005.