Positive Unlabeled Learning for Data Stream Classification

Xiao-Li Li[*] Philip S. Yu[†] Bing Liu[‡] See-Kiong Ng[§]

Abstract

Learning from positive and unlabeled examples (PU learning) has been investigated in recent years as an alternative learning model for dealing with situations where negative training examples are not available. It has many real world applications, but it has yet to be applied in the data stream environment where it is highly possible that only a small set of positive data and no negative data is available. An important challenge is to address the issue of concept drift in the data stream environment, which is not easily handled by the traditional PU learning techniques. This paper studies how to devise PU learning techniques for the data stream environment. Unlike existing data stream classification methods that assume both positive and negative training data are available for learning, we propose a novel PU learning technique LELC (PU Learning by Extracting Likely positive and negative micro-Clusters) for document classification. LELC only requires a small set of positive examples and a set of unlabeled examples which is easily obtainable in the data stream environment to build accurate classifiers. Experimental results show that LELC is a PU learning method that can effectively address the issues in the data stream environment with significantly better speed and accuracy on capturing concept drift than the existing state-of-the-art PU learning techniques.

1 Introduction.

Traditionally, supervised learning techniques are proposed to build accurate classifiers that require a large number of labeled training examples from the predefined classes for learning. In practice, this paradigm can be problematic because collecting and labelling large sets of training examples can be expensive and tedious. The alternative approach of Positive and Unlabeled (PU) learning has been investigated in recent years. PU learning reduces the amount of labeled training data by developing classification algorithms that can learn from a set of labeled positive examples augmented with a set of unlabeled examples. In other words, given a set P of positive examples of a particular class and a set U of unlabeled examples (which contains both hidden positive and hidden negative examples), we build a classifier using P and U to classify the data in U as well as future test data. Several PU learning techniques (e.g. [12], [13], [14], [22]) have recently been proposed to solve the PU learning problem in document classification domain with promising results.

The key difference of the PU learning techniques from the traditional classification approaches is that we can still learn an accurate classifier even without the negative training data. The dynamic data stream environment is one such environment where it is highly possible that only a small set of positive data and no negative data is available in practice. However, while PU learning has been applied successfully in static data mining scenarios, it has yet to be applied in the dynamic data stream environment. An important challenge is to address the issue of *concept drift* in the data stream environment, which is not easily handled by the traditional PU learning techniques. In this paper, we study how to devise PU learning techniques in the data stream environment.

With the advent of advanced data streaming technologies [1], we are now able to continuously collect large amounts of data in various application domains, e.g., daily fluctuations of stock market, traces of dynamic processes, credit card transactions, web click stream, network traffic monitoring, position updates of moving objects in location-based services and text streams from news etc [2]. Due to its potential in industry applications, data stream mining has been studied intensively in the past few years. In particular, much research has been focused on classifying data streams [3]-[9]. The general approach is to first learn one or multiple classification models from the past records of the evolving data, and then use a selected model that best matches the current data to predict the new data records. All the existing data stream classification techniques assume that at each time stamp there are both large amounts of positive and negative training data available for learning.

In practice, however, this assumption is often violated, i.e. we are unlikely to have the necessary negative training data for learning a classifier in a data stream environment. This is because stream data typically has the nature of being bursty and at the same

[*]Institute for Infocomm Research.
[†]University of Illinois at Chicago.
[‡]University of Illinois at Chicago.
[§]Institute for Infocomm Research.

time evolving rather rapidly. For example, a user may be interested in the recent burst of documents on a particular topic from the news data stream (e.g., today's breaking news from a news stream) and want to get similar documents about it subsequently. In such situations, while there is the availability of positive training examples (even if only a small number), it is not realistic to expect also the negative training examples. The ad hoc nature of the event makes it impossible to prepare pre-labeled (negative) training examples, and manually labelling some negative examples (typically a labour-intensive and time-consuming process, even in a non-data stream environment) on the spot is unlikely to be acceptable due to the highly dynamic and reactive nature of the tasks. Furthermore, the training data from the historical records may not be valid anymore since the documents of interest are likely to continue to evolve over time. This is referred to as *concept drift* [8].

Additionally, data stream classification requires real-time response. This means that obtaining an adequately large number of positive examples in a timely fashion can also be rather difficult in many real applications. As such, not only do we have to deal with the absence of negative data, we also have to make do with a fairly small set of positive data.

The problem is thus the following: can we still be able to build an accurate classifier with small positive data P and unlabeled data U in a concept drifting stream environment? This problem is a challenging one not only because we need to detect reliable negative examples (like other PU learning techniques) from the unlabeled examples, but also because the small positive set may not even adequately represent the whole positive class. On top of that, there is concept drifting in the data stream. We propose a novel PU learning technique to extract both likely positive examples and likely negative examples from unlabeled set U in the concept drifting stream environment. Here the positive data P is the small positive data, which represents a user's new interest in the current time stamp. The unlabeled data U is the historical data in previous time stamp, which consists of previous positive data *Pold* and previous negative data *Nold*, i.e. $U = Pold \cup Nold$. Note that not all the data in *Pold* (*Nold*) are examples for the positive (negative) data in the current time stamp because the labels might have changed due to the change of users' interests. Our proposed PU learning technique for data stream classification is called LELC (PU Learning by Extracting Likely positive and negative *micro-Clusters*). To address the problems of small positive data and the lack of negative data, after extracting the reliable negative documents from unlabeled set U, LELC focuses on extracting high-quality likely positive and negative *micro-clusters* from U. Unlike some previous works that extract likely positive data (document) individually, our proposed LELC extracts likely positive and negative *micro-clusters*, which is more robust than the existing techniques. The main contributions in this paper can be summarized as follows:

- For the first time, the PU learning problem is formally defined in the data stream environment. This enables the use of PU learning techniques for data stream classification tasks where it is common that no negative data but only a small positive dataset and unlabeled data are available for training.
- A novel classification algorithm LELC has been designed to extract high-quality positive and negative *micro-clusters* from data streams. The main innovation is in the technique for selecting likely positive and negative instances from the unlabeled set, which also exploits the characteristics of data streams. In contrast to the existing PU learning methods, which extract likely negative/positive instances point by point, the new method extracts them by small clusters (or micro-clusters [3]). The micro-clusters are formed by utilizing labels in the past streams (which are treated as unlabeled data in the new interval) to produce good clustering. It is important to note that although the past labels may no longer be of interest, data points of the same labels still form homogeneous groups. We call this technique *label consistent micro-clustering*. It is reasonable to assume that data points close together change their labels together under concept drift rather than each changing its label randomly.
- Experimental results of LELC algorithm show that such a PU learning method can effectively address the challenging issues in the data stream environment, significantly outperforming the existing state-of-the-art PU learning methods.

2 Related Work.

PU learning has been studied in the recent years to address the lack of negative training data in practice. A theoretical study of PAC learning from positive and unlabeled examples under the statistical query model was first reported in [10]. Muggleton [11] followed by studying the problem in a Bayesian framework where the distribution of functions and examples are assumed known. [12] reported sample complexity results and provided theoretical elaborations on how the problem could be solved.

Subsequently, a number of practical PU learning algorithms were proposed [12], [13], [14]. These PU learning algorithms all conformed to the theoretical results presented in [12] by following a common two-step strategy, namely: (1) identifying a set of reliable negative documents from the unlabeled set; and then (2) building a classifier using EM or SVM iteratively. The specific differences between the various algorithms in these two steps are as follows.

The S-EM method proposed in [12] was based on naïve Bayesian classification and the EM algorithm [15]. The main idea was to first use a spy technique to identify some reliable negative documents from the unlabeled set, and then to run EM to build the final classifier. The PEBL method [13] uses a different method (1-DNF) for identifying reliable negative examples and then runs SVM iteratively for classifier building [16]. More recently, [14] reported a technique called Roc-SVM. In this technique, reliable negative documents were extracted by using the information retrieval technique Rocchio [17]. Again, SVM is used in the second step. A classifier selection criterion is also proposed to catch a good classifier from iterations of SVM. Despite the differences in algorithmic details, the above methods all focused on extracting reliable negative instances from the unlabeled set.

Some of the recent works also looked into the extraction of likely positive data [23]. In [18], a method called PN-SVM was proposed to deal with the case when the positive set is small. PU learning has been used to classify electronic products [21] and identify unexpected instances in the test set [19].

Note that the problem could potentially be modelled as a one-class classification problem. For example, in [20], a one-class SVM that uses only positive data to build a SVM classifier was proposed. Such approaches are different from our method in that they do not use unlabeled data for training. As previous results reported in [14] have already showed that they were inferior for text classification, we do not consider them in this work.

We notice that all the current PU learning methods have been devised for static data environments. In this paper, we explore the application of PU learning in the dynamic data environments such as stream data classification. Such data environments have an inherent lack of negative examples and small positive examples where PU learning seemed to be naturally designed for such scenarios. However, as mentioned, there are challenging issues such as concept drift in the data stream environment which is not easily handled by the traditional PU learning techniques.

A number of techniques have been proposed to classify data stream [3]-[9]. The existing work can be divided into two classes according to how they deal with the historical records [24]. One class focuses on how to effectively update the classification model when stream data flows in [5] [3], by discarding the historical records after a certain period of time or gradually decreasing their weights as time elapses [24]. The other class devised methods for choosing historical records which can match well with the current data to help train a better model instead of just using the most recent data alone [25] [9] [6]. Some other related techniques for handling stream data include discovering high-order models from evolving data [26] to stop chasing the evolving trends and dealing with high dimensional classification through a resource adaptive approach [4].

These existing techniques all required both labeled positive and negative data at each time instance for their learning processes. They are not designed for solving the problem of classification with only positive and unlabeled data. On the other hand, traditional PU learning techniques do not address the issue of concept drift which occurs frequently in the data stream environment. It is thus worthwhile to explore the possibility of devising novel PU learning techniques for the data stream environment to address the lack of negative data and insufficient positive data in such environments.

3 Problem Definition.

Let us now describe how data stream classification can be modelled as a PU learning problem. Suppose we have a data stream D_1, D_2, $\ldots$, D_m, where each D_i (i = 1, 2, $\ldots$, m) denotes the data that arrived between time t_{i-1} and t_i. D_i can be split into positive data D_{ip} and negative data D_{in}, i.e. $D_i = D_{ip} \cup D_{in}$. In the current time t_{m+1}, we are given the current interest as the positive data D_C , and the aim of our data stream classification is to train a classifier based on D_C and the historical data D_i. The classifier can then be used to classify the future data stream.

Since a user's interests may change over time, one should not treat all the positives in the historical data, namely $\cup_{i=1}^{m} D_{ip}$, as the current positive data (similarly, we should not treat all the negatives in the historical data, namely $\cup_{i=1}^{m} D_{in}$, as the current negative data). To address the potential concept drift, we treat all historical data as unlabeled data. In other words, we only have (i) the *positive* data D_C which describes a user's current interest, and (ii) the *unlabeled data* D_i (i = 1, 2, $\ldots$, m) which describe a user's past interests. In this way, the data stream classification problem with concept drift can be modelled as a positive and unlabeled learning problem.

Clearly, it is impractical to use the entire historical data D_i (i = 1, 2, $\ldots$, m) as the unlabeled data for training a classifier because the massive and potentially obsolete historical data which may result in unacceptably slow and possibly inaccurate response to a user's requirements (recall that most data stream classification applications demand real-time responses), not to mention the large space needed to store the huge amount of data stream. As such, in this work, we focus on using only the most recent data D_m for learning. In other words, our PU learning problem is defined as follows: given the current positive data P (D_C), and the most recent data $D_m = D_{mp} \cup D_{mn}$ treated as the unlabeled data U, we train an accurate classifier that can identify the hidden positive data from the future data stream (or test set T). In this paper, we focus on the document classification task for this work.

4 The Proposed Technique.

Typically, a large number of labelled positive and negative training examples are required to learn a good classification model. In our case, we only have a small positive data P and an unlabeled data U. To build an accurate classifier, our proposed technique LELC has to extract trustworthy positive and negative examples by using P and U.

The LELC employs a *progressive* two stage approach. In stage one, a conservative ensemble strategy is used to extract very reliable negative examples from unlabelled data. In stage two, a novel approach exploring label consistency within small clusters (or micro-clusters) is developed to further extract likely positive and negative clusters from the unlabelled data.

Since we do not have any negative examples initially, our first task is to extract some reliable negative examples. We propose to use an ensemble strategy that integrates the two state-of-the-art PU learning extraction techniques, namely Spy extraction and Rocchio extraction, to extract a high quality reliable negative set RN (Section 4.1). Both extraction methods have relatively low error rates and independent in the production of their errors. By integrating them, we minimize the potential bias of individual methods and the expected errors in RN extracted by the ensemble classifier can be expected to be greatly reduced [27].

We notice that for our document classification task, typically $|RN| >> |P|$ and that the topics in P are usually related and the set of reliable negative documents in RN are from diverse topics. In order to balance the positive and negative data [28], we first cluster the negative documents in RN into several homogenous groups. A set of more accurate one-versus-one Rocchio classifiers [29] are subsequently built using P and each RN cluster (Section 4.2).

A potentially large number of positive and negative examples may still be in the remaining unlabeled data (U - RN) that can be exploited for building our classifier. We therefore continue our efforts to extract likely positive and negative data. To do so, we cluster the positive and negative documents in the past streams (after removing the documents in RN) by using their labels to produce good micro-clusters. Although the past labels may no longer be of interest in the current time interval, data points of the same labels still form homogeneous groups. Section 4.3 first describes this innovative clustering step which we called label consistent *micro-clustering.*

We are now ready to extract further likely negative/positive examples for building our accurate stream data classifier. We use the set of Rocchio classifiers to classify each micro cluster to decide if they are likely positive and negative clusters. In contrast to conventional PU learning methods which extract likely positive/negative instances point by point, our approach is novel in extracting likely positive/negative *micro-clusters* from the data streams. This innovative approach turns out to be a robust approach as we are able to exploit the fact that data points close together tend to change their labels together under concept drift rather than each changing its label randomly.

With the given positive set P, extracted reliable negative set RN, and the high-quality likely positive micro-clusters and likely negative micro-clusters, our LELC algorithm has adequate positive and negative data to build a robust classifier for data stream classification (Section 4.4). Since the ability to provide rapid response is often the key to most data stream classification applications, we also analyze the efficiency of our LELC algorithm in Section 4.5.

4.1 Extracting Reliable Negative Documents.

Supervised learning methods require both positive and negative training sets to learn a classification model. In PU learning, given that only the positive data P and the unlabeled data U are available, there is a need to extract some initial negative documents, i.e. *reliable* negative documents from unlabeled set U. The key requirement for this extraction step is that the identified negative documents from the unlabeled set U should be reasonably reliable or pure, i.e., with no or very few positive documents (or false negatives). This is because false negatives will affect the performance of the subsequent steps of extracting the likely positives, and the likely negatives, and the final step to build SVM classifier which is also very sensitive to noise. Currently, there are two existing PU learning techniques, i.e. Roc-SVM (Rocchio extraction) [14] and S-EM (Spy extraction) [12] that can be used for extracting large quantity of reliable negative documents from U (other techniques, such as 1-DNF technique in PEBL [13] can only extract a very small number of reliable negative documents).

Our proposed algorithm LELC uses an effective ensemble strategy [30] that integrates both the Spy extraction and the Rocchio extraction algorithms to extract the most reliable unbiased negatives. Documents are classified as negative class only if both Spy extraction and Rocchio extraction agree that they are negative documents. In this way, we minimize the potential bias of the individual methods and reduce the possibility of extracting the false negative documents [27].

Algorithm 1 shows the details of this process. The set RN is used to store the reliable negative instances identified. Step 1 initializes RN to the empty set, while Steps 2-3 initialize the positive set P (current positive set D_C) and the unlabeled set U (the union of the positive set D_{mp} and negative set D_{mn} in the last time instance). Steps 4 and 5 build the Spy extraction classifier F_{S-EM} [12] and the Rocchio extraction classifier $F_{Roc-SVM}$ [14] respectively. For each document d in unlabeled set U, steps 6-10 classify it into reliable negative set RN if both classifiers F_{S-EM} and $F_{Roc-SVM}$ classify them into negative class. The sets P' and U' are used to store the remaining documents of D_{mp} and D_{mn} respectively in Steps 11 and 12, excluding the reliable negative documents already identified in RN. Note that due to potential concept drift in the data stream environment, it is possible that some reliable negative documents are from the previous positive set D_{mp} (Step 11). Finally, Step 13 outputs the unbiased reliable negative set RN, the remaining positive set P', as well as the remaining negative set N'. Note that set $P' \cup N'$ becomes the new unlabeled set.

Algorithm 1 Extract Reliable Negative Documents

```
1:  $RN = \emptyset$;
2:  $P = D_C$;
3:  $U = D_{mp} \cup D_{mn}$;
4:  Build a Spy extraction classifier $F_{S-EM}$ using $P$ and $U$;
5:  Build a Rocchio extraction classifier $F_{Roc-SVM}$ using $P$ and $U$;
6:  for each document $d \in U$ do
7:      if $((F_{S-EM}(d) = -1) \&\& (F_{Roc-SVM}(d) = -1))$ then
8:          $RN = RN \cup \{d\}$;
9:      end if
10: end for
11: $P' = D_{mp} - RN$;
12: $N' = D_{mn} - RN$;
13: Output $RN$ , $P'$ and $N'$
```

4.2 Building representative positive and negative prototypes. At this point, we have a positive set P and a reliable negative set RN. Ideally, we can use them directly to build a classifier using a supervised classification technique to classify any other uncertainty documents. However, we noticed that in practice, while the topics in P are usually related, the set of negative documents in RN typically consists of a large quantity of documents of diverse topics. As such, it is good to first cluster the negative documents in RN into several homogenous groups, i.e. RN_1, RN_2, ..., and RN_r. Based on the positive data P and each homogenous negative data RN_i (i = 1, 2, ..., r), we can then build a more accurate classifier by constructing the corresponding positive and negative representative prototypes for one-versus-one classification.

The detailed algorithm is shown in the Algorithm 2. In step 1 of the algorithm, each document d is first represented as a vector [31] $\vec{d} = (q_1, q_2, \ldots, q_n)$. Each element q_i in $\vec{d}$ represents a word feature w_i which is calculated as the combination of term frequency (tf) and inverse document frequency (idf), namely, $q_i = tf_i * idf_i$, where tf_i is the number of times that the word w_i occurs in d, while idf_i is computed as

$$idf_i = \log \frac{|D|}{df(w_i)}$$

Here $|D|$ is the total number of documents and $df(w_i)$ is the number of documents where the word w_i occurred at least once.

After representing each document into a vector, the second step of the algorithm (Step 2) is to cluster the documents in RN into r groups such that documents within each group are more similar with each other. In this work, we employ the k-means clustering algorithm [32] as an efficient and effective clustering technique to cluster RN. In k-means clustering, *cosine similarity* [31] is used to evaluate the similarity between any two documents $\vec{d} = (q_1, q_2, \ldots, q_n)$ and $\vec{d'} = (p_1, p_2, \ldots, p_n)$:

$$cos(\vec{d}, \vec{d'}) = \frac{\Sigma_{i=1}^{n} q_i * p_i}{\sqrt{\Sigma_{i=1}^{n} q_i^2} * \sqrt{\Sigma_{i=1}^{n} p_i^2}}$$

In steps 3 to 6, we construct the positive and negative representative prototypes $\vec{p_i}$ and $\vec{n_i}$ (i = 1, 2, ..., r) which can then be used to classify the documents in the unlabeled set $P' \cup N'$. Note that α and β are two user-customisable parameters for adjusting the relative impact of positive and negative training examples. In this work, we use α= 16 and β= 4 which were recommended in [33].

4.3 Extracting Likely Positive and Negative Micro-Clusters. In the data stream environment, the number of positive documents in P is often insufficient,

Algorithm 2 Building representative positive and negative prototypes

```
1: Represent each document in $P$ and $RN$, using TFIDF representation;
2: Partition the reliable negative $RN$ into $r$ clusters $RN_1$, $RN_2$, …, and $RN_r$, where the number of clusters $r$ is set as $r = |RN|/(|P'|+|N'|+|RN|)*k$;
3: for $i = 1$ to $r$ do
4:     $\vec{p_i} = \alpha\frac{1}{|P|}\Sigma_{\vec{d}\in P}\frac{\vec{d}}{||\vec{d}||} - \beta\frac{1}{|RN_i|}\Sigma_{\vec{d}\in RN_i}\frac{\vec{d}}{||\vec{d}||}$;
5:     $\vec{n_i} = \alpha\frac{1}{|RN_i|}\Sigma_{\vec{d}\in RN_i}\frac{\vec{d}}{||\vec{d}||} - \beta\frac{1}{|P|}\Sigma_{\vec{d}\in P}\frac{\vec{d}}{||\vec{d}||}$;
6: end for
```

while the reliable negative documents in RN_i may be far away from the boundary between the actual positive and negative documents. It is thus quite likely that the classifier built by using only P and RN_i may not be accurate enough. As such, an important step in our LELC method is to extract the more *likely* positive documents LP as well as the more *likely* negative documents LN from the new unlabeled set $P' \cup N'$. The new sets LP and LN can then be used to enhance the original small positive set P and reliable negative training set RN to build accurate classifiers.

With the positive and negative prototype vectors $\vec{p_i}$ and $\vec{n_i}$ ($i = 1, 2, \ldots, r$) built using the Algorithm 2, we may build a Rocchio classifier to classify each document d from $P' \cup N'$ by checking if d is nearer to the positive or negative prototype vectors based on cosine similarity. However, since Rocchio is a linear classifier, when the decision boundary is non-linear or does not conform to the separating plane resulted from cosine similarity, Rocchio may inaccurately classify some documents into false positives or false negatives. This will lead to further damage in our final classifier. Figure 1 shows a possible scenario when we learn a Rocchio classifier with P and RN_i. In the figure, $\vec{p_i}$ and $\vec{n_i}$ represent the positive and negative prototypes (or prototype vectors) respectively. H is the decision hyperplane produced by Rocchio classifier. It has the exactly same distance (similarity) to $\vec{p_i}$ and $\vec{n_i}$. If a document$d$ is located on the right-hand-side of H, i.e. *sim* $(d, \vec{p_i}) <$ *sim* $(d, \vec{n_i})$, it is classified as negative; otherwise it is classified as positive where *sim* $(d, \vec{p_i}) \geq$ *sim* $(d, \vec{n_i})$. However, the positive and negative classes in the data cannot be separated by H well in Figure 1, and some positive documents in the regions 1 and 3 are misclassified as negative documents (false negatives) while the negative documents within the region 2 are misclassified as positive documents (false positives).

From the discussion above, it is clear that classification based on individual document classification may

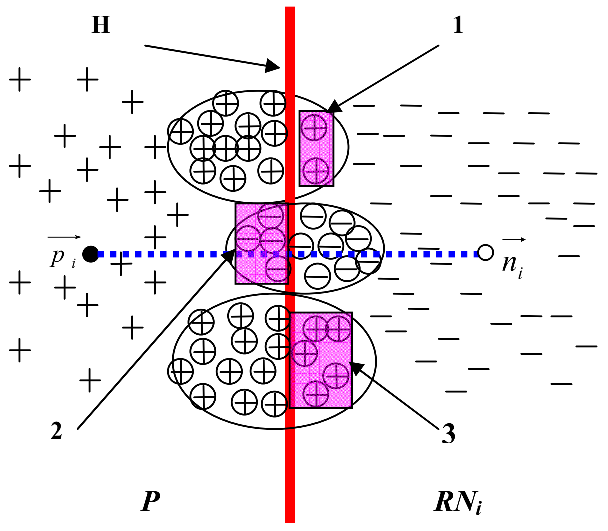

Figure 1: False positives and false negatives classified by using a Rocchio classifier

not work well. As such, we investigate how to extract likely positive and negative micro-clusters instead using a novel approach based on *label consistent micro-clustering*. First, we cluster P' into micro-clusters P_1, P_2, …, P_m and N' into micro-clusters N_1, N_2,…, N_n using the k-means clustering. Instead of directly deciding the label of each individual document in $P' \cup N'$, we classify each micro-cluster P_i and N_j ($i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$). Since the documents in each micro-cluster are similar, they should have higher possibility to belong to same class. Thus, for each micro-cluster, we can first classify each document in the micro-cluster to give them temporary labels. We then employ a voting strategy to decide the micro-cluster's category and use it to decide the final label for each document (namely, a document final label is assigned by using its micro-cluster's category). This is a robust technique since it not only considers the similarities of each individual document to the positive and negative prototype vectors, but it also takes the relationships between documents in the micro-clusters (i.e. similar documents in one micro-cluster should belong to the same class) into account when assigning the final labels for the documents. In the example shown in Figure 1, the likely positive set LP would consist of all the documents within the entire micro-clusters, namely regions 1 and 3, which are classified as belonging to the positive class because most of the documents in the micro-cluster were classified as positives. Similarly, the documents in region 2 will be classified as negatives together as a micro-cluster, and stored into the likely negative set LN, since most of the documents in the micro-cluster were classified as negatives. Note that some micro-cluster P_i can become likely negative (LN) and micro-cluster N_i may become likely positive (LP) due to the concept drift in data stream environment.

Algorithm 3 Extracting the likely positive and negative micro-clusters

```
1: Partition $P'$ and $N'$ into micro-clusters $C_P = \{P_1, P_2, \ldots, P_m\}, C_N = \{N_1, N_2, \ldots, N_n\}$ respectively. The number of micro-clusters for $P'$ is $m = |P'|/(|P'| + |N'| + |RN|) * k$, The number of micro-clusters for $N'$ is $n = |N'|/(|P'| + |N'| + |RN|) * k$;
2: $LP = \emptyset, LN = \emptyset$;
3: for each micro-cluster $C_i \in C_P = \{P_1, P_2, \ldots, P_m\} \cup C_N = \{N_1, N_2, \ldots, N_n\}$ do
4:   $pos\_vote = 0$;
5:   $neg\_vote = 0$;
6:   for each document $d \in C_i$ do
7:     if $(\max_{i=1}^{r} sim(d, \vec{p}_i) > \max_{i=1}^{r} sim(d, \vec{n}_i))$ then
8:       $pos\_vote + +$;
9:     else
10:      $neg\_vote + +$;
11:    end if
12:  end for
13:  if $(pos\_vote > neg\_vote)$ then
14:    $LP = LP \cup C_i$;
15:  else
16:    $LN = LN \cup C_i$;
17:  end if
18: end for
```

Algorithm 3 shows the detailed procedure. In step 1, to exploit label consistent micro-clustering, we cluster separately the past positive set P' into m groups P_1, P_2, …, P_m and the past negative set N' into n groups N_1, N_2,…, N_n. The number of the micro-clusters for m (for P') and n (for N') is proportioned to the size of RN, P' and N' and is decided by a parameter k (similarly r for RN in Algorithm 2). From our experimental results in Section 5, we observe that the choice of k does not affect classification results much as long as it is not too small or too big.

Next, the likely positive set LP and the likely negative set LN are initialized as empty set in Step 2. Then, from Step 3 to Step 18, we decide the class label of the each micro-cluster C_i in C_P and C_N. We initialize the two variables *pos_vote* (count the number of positive support for micro-cluster C_i) and *neg_vote* (count the number of negative support for micro-cluster C_i) as zero. In Steps 6 to 12, we classify each document d in micro-cluster C_i, by comparing its biggest positive similarity (the highest similarity with all the positive prototype vectors) with the biggest negative similarity (the highest similarity with all the negative prototype vectors) and update the *pos_vote* and *neg_vote* according to d's temporary label. Finally, in steps 13 to 17, we classify the entire micro-cluster C_i based on the values of *pos_vote* and *neg_vote* – if *pos_vote* is bigger than *neg_vote*, then the whole micro-cluster C_i and its corresponding documents will be classified as positive documents and put into the likely positive set LP; otherwise they are classified as negative documents and put into the likely negative set LN.

4.4 Building the Final Classifier.

Finally, we build an SVM classifier with the positive set P and the reliable negative set RN, enhanced by the likely positive set LP and the likely negative set LN.

Algorithm 4 Building the final SVM classifier

```
1: $P = P \cup LP$;
2: $N = RN \cup LN$;
3: Build a final SVM classifier $F_{SVM}$ using $P$ and $N$.
```

In Steps 1 and 2 of Algorithm 4, we update the positive set P by its union with LP, and build a negative set N by the union of the reliable negative set RN with the likely negative set LN. We finally build an SVM classifier by using the resulting positive set P and negative set N. We use SVM in this work because of its superior performance in text classification [16] [34]; the other classification techniques could also be applied, if preferred.

4.5 Analysing the Efficiency of LELC algorithm.

The main steps for our LELC algorithm include 1) integrating Rocchio (using Rocchio classifier) with Spy extraction (using naïve Bayesian classifier) to extract reliable the negative documents, 2) k-means clustering to partition the reliable negatives RN and the remaining uncertainty unlabeled set P' and N', 3) Rocchio classifier for extracting the likely positive and negative micro-clusters, and 4) employing SVM to build the final classifier. Since the steps from 1) to 3) can be performed using efficient algorithms implemented in linear time, the main time complexity is the time used to build the final SVM classifier. While current PU learning techniques such as S-EM, Roc-SVM and PEBL all need to iteratively build their SVM or EM classifiers, our proposed LELC technique is much more efficient than these state-of-the-art PU learning techniques since it only needs to build a single SVM classifier. This is very important in the data stream environment where the need to quickly respond to a user's request is often expected.

5 Empirical Evaluation.

In this section, we evaluate the proposed LELC technique under different experimental settings and compare it with three main existing methods, namely S-EM [12], PEBL [13] and Roc-SVM [14]. Both Roc-SVM and S-EM are available on the Web as a part of the LPU system (http://www.cs.uic.edu/~liub/LPU/LPU-download.html). We implemented PEBL ourselves as it is not publicly available. We do not compare with other current data stream classification techniques as they cannot be applied to the scenario that we are considering in this work, namely having only a small set of positive documents and no negative data available for training.

We use the 20 Newsgroups text collection [35], which is commonly used in evaluating text classification methods, to simulate the data stream environment by generating different data at different time stamps. The 20 Newsgroups collection contains documents from 20 different UseNet discussion groups, which are also categorized into 4 main categories, i.e. computer, recreation, science, and talk. For a fair comparison, we have removed all the UseNet headers (thereby discarding the potentially telling subject line) in our experiments.

We use the *F-measure* to evaluate the performance of the final classifier to identify or retrieve positive documents from the future data stream. The *F-measure* is the harmonic mean of precision (p) and recall (r), and it is defined as $F = 2*p*r/(p+r)$. In other words, the *F-measure* reflects an average effect of both precision and recall. When either of them (p or r) is small, the value will be small. Only when both of them are large, *F-measure* will be large. This is suitable for our purpose to accurately identify all the positive documents from the future data stream. Having either too small a precision or too small a recall is unacceptable and would be reflected by a low *F-measure*. Note that the *F-measure* is a commonly used metric for evaluating the performance of many classification systems; the reader may refer to [36] for further details of its theoretical bases and practical advantages.

5.1 Experiment settings . Our objective is to evaluate whether our technique can handle scenarios where there is a concept drift in the data stream. Let us describe how we can use the benchmark 20 Newsgroups data to simulate such data streams. Specifically, we describe how to generate the positive data P, unlabeled data U and test set T at different time stamps.

First, we define a super positive set SP which consists of several related newsgroups (classes) out of the 20 newsgroups. The corresponding base negative set BN is defined as the remaining newsgroups of all the 20 Newsgroups. For example, suppose SP is a set of 5 classes of computer related documents, i.e. SP = {pc.hardware, mac.hardware, windows.x, ms-windows, comp.graphics}. Then, the base negative set BN would comprise the remaining 15 newsgroups, i.e. BN = {rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, alt.atheism, misc.forsale, soc.religion.christian, talk.politics.guns, talk.politics.mideast, talk.politics.misc, talk.religion.misc}. Table 1 shows all the 4 possible super positive sets SP used in this paper (BN is not shown in table 1).

Table 1: Super positive set SP

Categories	**Super positive set SP**
Computer	pc.hardware, mac.hardware, windows.x, ms-windows, comp.graphics
Recreation	rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey
Science	sci.crypt, sci.electronics, sci.med, sci.space
Talk	talk.politics.guns, talk.politics.mideast, talk.politics.misc, talk.religion.misc

Given a super positive set SP and the corresponding base negative set BN, we generate positive data, unlabeled data and test data that simulate the data stream at different time stamps as follows. Suppose at time instance T_{i-1} we have the data $D_{i-1} = D_{i-1,p} \cup D_{i-1,n}$ where the $D_{i-1,p}$ is the set of positive documents generated from some classes PC, $PC \subseteq SP$ and $D_{i-1,n}$ is the set of negative documents generated from the classes in BN. We need to generate the new positive set in time instance T_i – note that we do not need to generate any negative documents here, since only positive data are available at the current time T_i. Figure 2 illustrates how we simulate a concept drift in the data stream environment. Given the positive and negative data in time T_{i-1}, the data generation process for time T_i consists of two random steps. First, we randomly select either an "Add" or a "Remove" action (i.e. each with the same 50% probability). "Add" represents the event that the user has got a new class of interest besides the original ones. "Remove" indicates that the user has dropped a class of interest that he/she currently has. Then, we randomly decide a class P_i from SP - PC to add or a class P_j to remove from PC, depending on the action selected. If we do not have any class in T_{i-1}, then we do not remove any class in T_i. Instead, we add one new class P_i. If we already have all positives class in T_{i-1}, then in T_i we do not add any class but we remove an old class P_j from PC instead.

Once we have selected to "add" or "remove" a class, let us now describe how to generate the documents for the positive data set, unlabeled data set, and test data set in time stamp T_{i-1} and T_i:

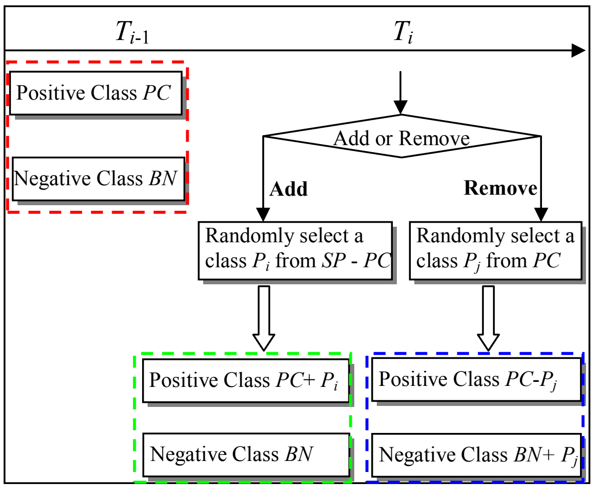
$T_{i\text{-}1}$
T_i
Positive Class *PC*
Negative Class *BN*
Add or Remove
Add
Remove
Randomly select a class P_i from *SP* - *PC*
Randomly select a class P_j from *PC*
Positive Class *PC*+ P_i
Negative Class *BN*
Positive Class *PC*-P_j
Negative Class *BN*+ P_j

Figure 2: Simulate the concept drift in data stream environment

<u>*Case*1</u>. We have selected to "add" a class P_i. For each class in *PC*, 30% of documents are randomly selected as test positive set in T. Out of the remaining 70% of documents, a small number of documents, say s documents for each class in *PC*, is randomly selected and put into the new positive document set $D_{i,p}$ in T_i. The remaining positive documents are placed in the positive set $D_{i-1,p}$ in T_{i-1}.

For the newly added positive class P_i, we randomly select 30% of the documents as the positive documents in the test set T. Also, we randomly select s documents from P_i and place them in the new positive document set $D_{i,p}$ in T_i. The remaining documents are ignored so that we can keep a similar proportion of documents for each class in the test set and the unlabeled set (or D_{i-1}). Since the number of positively labeled documents available in practice can be quite small (either because there were few documents to start with, or it is simply too tedious and expensive to hand-label the training examples on a large scale), to reflect this constraint, we experiment with various small numbers of (randomly selected) positive documents, namely, s = 30, 50, or 70.

<u>*Case*2</u>. We have selected to "remove" a class P_j. In this case, for each class in *PC*-P_j, we randomly select 30% of the documents as the test positive set in T. Out of the remaining 70% of documents, a small number of the documents, say s documents for each class in *PC*-P_j, is randomly selected and placed in the new positive document set $D_{i,p}$ in T_i. The remaining positive documents are all put into the positive set $D_{i-1,p}$ in T_{i-1}.

For the positive class P_i that is to be removed, we randomly select 30% documents as the negative documents in the test set T, since the user is no longer interested in this class of documents. Also, we randomly select s documents from P_i and remove them. The remaining documents are placed in the old positive document set $D_{i-1,p}$ in T_{i-1}.

In both Case 1 and Case 2, we also randomly select 30% of the documents from the base negative set *BN* as the negative test set in T. The remaining 70% documents are placed in $D_{i-1,n}$ in T_{i-1}.

For every entry (super positive set *SP*) in Table 1, we randomly select an "add" or a "remove" action as well as the class to "add" or "remove". We perform the data generation step 100 times to simulate a data stream with concept drift in 100 time instances, from T_1 to T_{100}. Because of the random nature of the experiments, the results reported below are the average values for the results from the 100 randomly assembled positive, unlabeled and test sets.

5.2 Experimental Results.

We performed various evaluation experiments to show the effectiveness of our LELC method. First of all, let us present the comparative results on the different data sets.

1) *Comparing overall performance of various techniques.* To illustrate the relative performance of various techniques when the number of available positively labeled documents is quite small, we show the results obtained by using s = 50, i.e. each positive class having only 50 documents. For completeness, we have also experimented with two different numbers of positive documents, namely s = 30 and s = 70; the results will be shown later (Figure 4).

Table 2: The overall performance of various techniques

Data Sets	**S-EM**	**PEBL**	**Roc-SVM**	**LELC**
Computer	59.3	4.6	31.3	83.9
Recreation	83.2	5.8	34.9	90.8
Science	71.2	5.2	20.5	78.2
Talk	66.8	4.7	35.7	79.6
Average	70.1	5.1	30.6	83.1

Table 2 shows the classification results of various techniques in terms of F-measure with the cluster parameter k = 30 (note that we also experimented with different k in the later part of the experiments). The first column lists the 4 different data sets (categories), each having 4 to 5 classes of related documents as *SP* (See Table 1). Columns 2 to 4 show the results of the three existing PU techniques S-EM [12], PEBL [13] and Roc-SVM [14] respectively. Column 5 gives the corresponding results of our proposed LELC technique. As mentioned, each result in the table is the average result of 100 random runs where we randomly select the "Add" or "Remove" a class at each time stamp to simulate the data stream scenario with concept drift. Due to the space limitation, we can only show the average results for each data set. Table 2 shows that LELC produces the best results consistently across all the data sets, achieving an F-measure of 83.1% on average, which is 13.0%, 78.0% and 52.5% higher than the F-measures of the three existing techniques S-EM, PEBL and Roc-SVM respectively. We notice S-EM was the best of the three existing techniques but PEBL performed poorly in this setting. This observation of the current technique is consistent with the previous results reported in [18] [14] where there were also only small positive example sets available. However, it is clear that LELC is much better suited for accurate data stream classification than the current PU learning methods.

2)*Extracting Reliable Negative Documents.* A key step of PU learning is to extract reliable negative documents from the unlabeled set U. To better understand the overall results presented above, we also compared the first step of the four techniques S-EM, PEBL, Roc-SVM and LELC that extracts the reliable negative documents.

Table 3: The first step of PU learning techniques to extract reliable negative examples in data set *computer*

Techniques	**ExtN**	**TN**	**FN**	**RFN**
S-EM	8371.4	8142.9	228.5	2.7
PEBL	173.6	170.6	3.0	1.7
Roc-SVM	10474.5	10198.4	276.1	2.6
LELC	8117.6	8028.3	89.3	1.1

Table 3 compares the performance of the four techniques for extracting the negative documents in the data set *computer*, again using cluster parameter $k = 30$. Columns 2 to 5 show the number of extracted reliable negatives (ExtN), the number of extracted true negatives (TN), the number of extracted false negatives (FN), and the ratio of extracted false negatives (RFN) respectively. We observe that S-EM, Roc-SVM and our proposed LELC extracted 8371.4, 10474.5 and 8117.6 reliable negative documents on average, while PEBL (using 1-DNF) can only extract 173.6 negative documents (column 2). From the column 3 and 4, we can see that S-EM and Roc-SVM extracted 228.5 and 276.1 false negatives on average, while LELC only extracted 89.3 false negatives, which is significantly smaller than both S-EM and Roc-SVM. This indicates that the design of our LELC method was effective in minimizing the bias of each method for extracting very pure negative examples. Most importantly, LELC can extract a large number of reliable negatives (higher recall) and its number of false negatives is much smaller than the individual S-EM and Roc-SVM (higher precision). We can clearly see from column 5 that LELC is lowest in terms of the ratio of extracted false negatives, i.e. only 1.1%. This is crucial for our algorithm since the false negatives will affect the accuracy of the remaining steps and the final classification results. We notice that while PEBL has a relatively lower ratio of extracted false negatives than S-EM and Roc-SVM, it still performed poorly since it managed to extract only a very small number of likely negatives.

Table 4: Comparison of extracting individual likely positive/negative documents and micro-clusters

Data Sets	**Positive**		**Negative**	
	Individual	Cluster	Individual	Cluster
Computer	1159.5	1219.6	2586.5	2875.3
Recreation	1276.0	2557.4	3563.7	4057.7
Science	899.5	1021.7	4877.1	5668.2
Talk	1122.1	3139.0	4838.3	5433.6
Average	1114.3	1984.4	3966.4	4508.7

3)*Extracting Likely Positive & Negative Micro-Clusters.* Recall that instead directly deciding the individual document's class labels, our LELC algorithm (as shown in Algorithm 3) partitions the unlabeled documents into micro-clusters and then decide the category of each micro-cluster (and subsequently the labels of the documents in the micro-clusters) by a voting strategy. Table 4 shows the detailed results that compare using "individual" document extraction with using "micro-cluster" extraction for determining the positive and negative training data across the four evaluation data sets. Using our LELC algorithm with "micro-cluster" extraction on the four data sets, we can extract, on average, 1984.4 positive documents and 4508.7 negative documents correctly (with all the documents in the micro-cluster classified as positive if the cluster's positive voting is larger than negative voting, and vice versa). If we adopt the "individual" document labelling strategy (i.e. using the positive and negative prototype vectors to classify each individual document directly), we can only correctly extract 1114.3 positive documents and 3966.4 negative documents. In other words, on average, our proposed LELC algorithm can correctly classify 870.1 (1984.4-1114.3) more positive documents and 542.3 (4508.7-3966.4) more negative documents, with the use of the novel micro-cluster assignment approach. By being able to extract larger sets of likely positive and negative documents with less false positive and false negative documents, our LELC is therefore able to build more accurate classifiers than existing techniques, as we have already shown in the beginning of this subsection.

4)*Evaluating the effectiveness of the parameters.* To study the sensitivity of the number of the micro-clusters for our LELC algorithm, we also performed a series of experiments using different numbers of micro-clusters k from 10 to 100 with a step of 10. The results on the four evaluation data sets are shown in Figure 3. We observe that LELC's performance is quite stable across the different values of k. The choice of k is not crucial to obtain the best results as long as the k is not too small or too big.

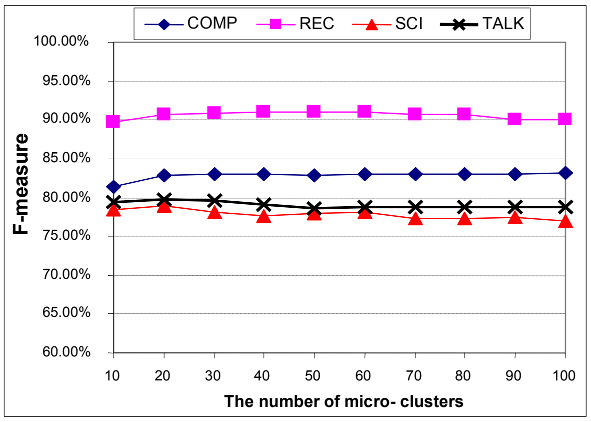

Figure 3: The sensitivity of the number of micro-clusters

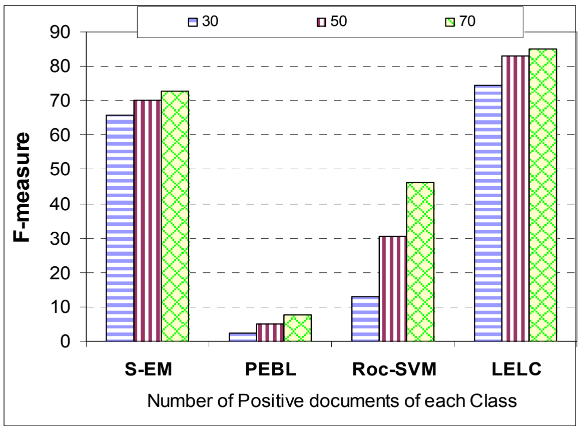

Figure 4: F-measures of different techniques with different numbers of positive documents of each class in P

We also experimented with different number of positive documents for each class in P , i.e. s = 30, 50 and 70 (Figure 4) for the different techniques S-EM, PEBL, Roc-SVM and LELC. The purpose of these experiments is to investigate the relative effectiveness of various techniques for different number of positive documents.

We observe from Figure 4 that our proposed technique LELC consistently performed much better than the three current methods S-EM, PEBL and Roc-SVM when positive data is small e.g. s=30, 50 and 70. As expected, all techniques improved their results when the size of positive data increases (especially Roc-SVM increases much faster than other techniques). However, since in the data stream environment with concept drift, it is more often than not that the positive data is small, it is important for the data stream classification method to perform well with small positive data, and our LELC method is clearly the current best PU learning method for such classification tasks.

6 Conclusions.

Learning from positive and unlabeled examples (PU learning) has been investigated in recent years as an alternative learning model to handle the common situations where negative training examples are not available. PU learning has many real world applications but it has yet to be applied in the data stream environment. This research is the first attempt to propose the application of PU learning in the data stream environment.

An important challenge in data stream classification is to address the issue of concept drift, one which the current PU learning techniques have not been able to handle well. This paper studies how to devise novel techniques to enable robust PU learning in such data stream environment. We have proposed a novel PU learning technique called LELC (PU Learning by Extracting Likely positive and negative micro-Clusters). As a PU learning method, LELC is fundamentally different from existing data stream classification methods which assume both positive and negative training data are available for learning. In fact, our LELC method only requires a small set of positive documents and a set of unlabeled documents to build accurate classifiers. This is important for the data stream environment where it is often the case that not only the negative training examples are absent, but the number of positive examples available for learning can also be fairly limited.

Our proposed LELC technique works well because it can extract reliable negative documents much more effectively than existing techniques, as shown by our experimental results. In addition, LELC technique has been designed to extract high-quality positive and negative micro-clusters from unlabelled data. The main innovation is in the technique for selecting likely positive and negative instances from the unlabeled set, which exploits the characteristics of data streams, i.e. data points close together change their labels together under concept drift rather than each changing its label randomly. In contrast to the existing PU learning methods, which extract likely negative/positive instances point by point, the new method extracts them by micro-clusters. The micro-clusters are formed by utilizing labels in the past streams to produce good clustering, i.e. *label consistent micro-clustering.*

Since LELC can automatically extract high-quality positive and negative micro-clusters from data streams, the limitations associated with the original positive set

P, such as its limited size, does not have a great impact on our algorithm. Augmented by the high quality likely positive set LP and likely negative set LN that resulted, our LELC algorithm is thus able to build a robust classifier for data stream classification. The experimental results showed that LELC can be used in the data stream environment much more effectively than the current PU learning methods, with significantly better speed and accuracy.

References

[1] S. B. B. Babcock, M. Datar, R. Motwani, and J. Widom, *Models and issues in Data Stream Systems*, PODS, 2002.

[2] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, *Querying and Mining of time Series Data: Experimental Comparison of Representations and Distance Measures*, VLDB, 2008.

[3] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, *On Demand Classification on Data Streams*, SIGKDD, 2004.

[4] C. C. Aggarwal and P. S. Yu, *LOCUST: An Online Analytical Processing Framework for High Dimensional Classification of Data Steams*, ICDE, 2008.

[5] G. Hulten, L. Spencer, and P. Domingos, *Mining time-changing data streams*, SIGKDD, 2001.

[6] W. N. Street and Y. Kim, *A streaming ensemble algorithm (SEA) for large-scale classification*, SIGKDD, 2001.

[7] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, *Multi-dimensional regression analysis of time-series data streams*, VLDB, 2002.

[8] H. Wang, J. Yin, J. Pei, P. S. Yu, and J. X. Yu, *Suppressing model overfitting in mining concept-drifting data streams*, SIGKDD, 2006.

[9] H. Wang, W. Fan, P. S. Yu, and J. Han, *Mining concept-drifting data streams using ensemble classifiers*, SIGKDD, 2003.

[10] F. Denis, *PAC Learning from Positive Statistical Queries*, ALT, 1998.

[11] S. Muggleton, *Learning from Positive Data*, in Proceedings of the sixth International Workshop on Inductive Logic Programming: Springer-Verlag, 1997, pp. 358-376.

[12] B. Liu, W. S. Lee, P. S. Yu, and X. Li, *Partially Supervised Classification of Text Documents*, ICML, 2002.

[13] H. Yu, J. Han, and K. C. -C. Chang, *PEBL: Positive Example Based Learning for Web Page Classification Using SVM*, SIGKDD, 2002.

[14] X. Li and B. Liu, *Learning to Classify Texts Using Positive and Unlabeled Data*, IJCAI, 2003.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of the Royal Statistical Society, 1977.

[16] V. N. Vapnik, *The nature of statistical learning theory*, Springer, 1995.

[17] J. Rocchio, *Relevance feedback in information retrieval.* In G. Salton (ed.). The SMART Retrieval System: Experiments in Automatic Document Processing, 1971.

[18] G. P. C. Fung, J. X. Yu, H. Lu, and P. S. Yu, *Text Classification without Negative Examples Revisit*, IEEE TKDE, vol. 18, pp. 6-20, 2006.

[19] X. Li, B. Liu, and S. -K. Ng, *Learning to Identify Unexpected Instances in the Test Set*,IJCAI, 2007.

[20] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, *Estimating the Support of a High-Dimensional Distribution*, Neural Comput, vol. 13, pp. 1443-1471, 2001.

[21] X. Li and B. Liu, *Learning from Positive and Unlabeled Examples with Different Data Distributions*, ECML, 2005.

[22] B. Liu, Y. Dai, W. S. Lee, P. S. Yu, and X. Li, *Building Text Classifiers Using Positive and Unlabeled Examples*, ICDM, 2003.

[23] X. Li, B. Liu and S. -K. Ng, *Learning to Classify Documents with Only a Small Positive Training Set*, ECML, 2007.

[24] J. Gao, W. Fan, and J. Han, *On Appropriate Assumptions to Mine Data Streams: Analysis and Practice*, ICDM, 2007.

[25] G. Widmer and M. Kubat, *Learning in the presence of concept drift and hidden contexts*, Machine Learning, vol. 23, pp. 69-101, 1996.

[26] S. Chen, H. Wang, S. Zhou, and P. S. Yu, *Stop Chasing Trends: Discovering High Order Models in Evolving Data*, ICDE, 2008.

[27] L. Hansen and P. Salamon, *Neural network ensembles*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, pp. 993-1001, 1990

[28] F. Provost, *Machine learning from imbalanced data sets*, presented at Proceedings of the AAAI'2000 Workshop on Imbalanced Data Sets, 2000.

[29] C. -W. Hsu and C. -J. Lin, *A comparison of methods for multi-class support vector machines*, IEEE. Transactions on Neural Networks, vol. 13, pp. 415-425, 2002.

[30] T. G. Dietterich, *Ensemble methods in machine learning*, vol. LNCS Vol. 1857: Springer 2001.

[31] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, 1986.

[32] P. S. Bradley and U. M. Fayyad, *Refining Initial Points for K-Means Clustering*, ICML, 1998.

[33] C. Buckley, G. Salton, and J. Allan, *The effect of adding relevance information in a relevance feedback environment*, SIGIR, 1994.

[34] T. Joachims, *Making large-scale support vector machine learning practical*, in Advances in kernel methods: support vector learning, 1999, pp. 169-184.

[35] K. Lang, *NewsWeeder: Learning to Filter Netnews*, ICML, 1995.

[36] J. William M. Shaw, *On the foundation of evaluation*, American society for information science,1986.