

## Singapore Management University Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

6-2018

# Social stream classification with emerging new labels

Xin MU

*Nanjing University*

Feida ZHU

*Singapore Management University, fdzhu@smu.edu.sg*

Yue LIU

*Singapore Management University, yueliu@smu.edu.sg*

Ee-peng LIM


*Singapore Management University, eplim@smu.edu.sg*

Zhi-Hua ZHOU

*Nanjing University*

**DOI:** [https://doi.org/10.1007/978-3-319-93034-3\\_2](https://doi.org/10.1007/978-3-319-93034-3_2)

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)

 Part of the [Databases and Information Systems Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

### Citation

MU, Xin; ZHU, Feida; LIU, Yue; LIM, Ee-peng; and ZHOU, Zhi-Hua. Social stream classification with emerging new labels. (2018). *Advances in knowledge discovery and data mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, Australia, June 3-6, 2018: Proceedings*. 10937, 16-28. Research Collection School Of Information Systems.

**Available at:** [https://ink.library.smu.edu.sg/sis\\_research/4079](https://ink.library.smu.edu.sg/sis_research/4079)

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [libIR@smu.edu.sg](mailto:libIR@smu.edu.sg).

# Social Stream Classification with Emerging New Labels

Xin Mu<sup>1,2</sup>(✉), Feida Zhu<sup>2</sup>, Yue Liu<sup>2</sup>, Ee-Peng Lim<sup>2</sup>, and Zhi-Hua Zhou<sup>1</sup>

<sup>1</sup> National Key Laboratory for Novel Software Technology,  
Nanjing University, Nanjing 210023, China  
{mux,zhouzh}@lamda.nju.edu.cn

<sup>2</sup> School of Information Systems, Singapore Management University,  
Singapore, Singapore  
{fdzhu,yueliu,eplim}@smu.edu.sg

**Abstract.** As an important research topic with well-recognized practical values, classification of social streams has been identified with increasing popularity with social data, such as the tweet stream generated by Twitter users in chronological order. A salient, and perhaps also the most interesting, feature of such user-generated content is its never-failing novelty, which, unfortunately, would challenge most traditional pre-trained classification models as they are built based on fixed label set and would therefore fail to identify new labels as they emerge. In this paper, we study the problem of classification of social streams with emerging new labels, and propose a novel ensemble framework, integrating an instance-based learner and a label-based learner by completely-random trees. The proposed framework can not only classify known labels in the multi-label scenario, but also detect emerging new labels and update itself in the data stream. Extensive experiments on real-world stream data set from *Weibo*, a Chinese micro-blogging platform, demonstrate the superiority of our approach over the state-of-the-art methods.

**Keywords:** Stream classification · Emerging new labels  
Model update

## 1 Introduction

Social stream classification has attracted an ever-increasing level of attention from both academia and industry due to the recent boom of social media platforms such as Twitter, in which the user-generated contents (i.e., tweets) naturally form a data stream by chronological order. As each item could assume one or multiple labels based on its content, classifying tweets into their corresponding labels serves as the foundation for profiling both users and information diffusion processes, in turn contributing to many real-life applications including targeted marketing, customer relationship management and credit risk evaluation.

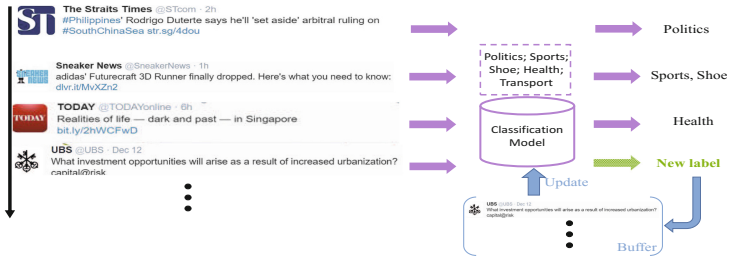


Fig. 1. An illustration of SSC-NL problem.

The basic process of social stream classification can be described as follows: with a set of social data which has been preprocessed and manually associated with concepts (labels), a classification method, such as SVMs or Random Forest, can be employed to train a model to predict labels for new incoming data. Despite the wealth of research efforts on social stream classification [1–3], most existing solutions, which are built with a fixed label set, face serious challenges when dealing with social data with emerging new labels due to its salient feature of topic novelty that is typical of social media content.

We therefore address in this paper a more challenging problem of Social Stream Classification with emerging New Labels (SSC-NL). Compared to previous social stream classification problems, the SSC-NL problem needs to accomplish three tasks simultaneously: (1) detecting emerging new labels; (2) classifying known labels in the multi-label scenario; and (3) updating the model with new labels identified. We illustrate this problem with a case for tweet stream classification in Fig. 1. We assume that the model is built initially with labels such as Politics, Sports, etc. This model is deployed in a tweet stream to classify each tweet with known labels, and correctly detect tweets with new labels as they emerge. These tweets with new labels are placed in a buffer, until the model update is triggered by some pre-defined criteria. Once the update is completed, the buffer is reset and the new model is ready for the next tweets in the stream.

To address the SSC-NL problem, we propose a novel ensemble framework named NL-Forest, which involves two cooperating models, an instance-based model and a label-based model, both composed of completely-random trees. NL-Forest can predict a ranking of known labels, and identify emerging new labels in the social streams. Furthermore, the models are to be updated once some criteria are met. We summarize some key contributions as follows: (1) The proposed method accomplishes three tasks simultaneously, including detection of new labels, classification for known labels, and model updating; (2) A straightforward approach for SSC-NL problem is to learn a known labels classifier and a new label detector like [4]. Compared to methods based on two distinct algorithm structures, completely-random trees are used as a single core to provide the solution to efficient prediction and updating as shown in Sect. 5.3. In addition, our model achieves better prediction performance and is more robust due to its ensemble strategy; (3) Experiments are conducted on both a real-world data

stream and simulated streams where new labels appear under different scenarios. Our framework outperforms existing state-of-the-art methods.

The rest of this paper is organized as follows: Sect. 2 examines the related work. We introduce the proposed framework in Sect. 4 and the experimental evaluation is detailed in Sect. 5. We conclude the paper in Sect. 6.

## 2 Related Work

Social stream classification has been extensively studied in recent years. Zubiaga and Spina [3] analyzed social features, and then performed classification experiments with Support Vector Machines (SVMs). In [1], a text-based classification method and a network-based classification method were proposed for classifying social data topics. Other than these supervised classification models, unsupervised learning was also widely used. For instance, topic modeling is effective in grouping documents into a pre-defined number of coarse clusters based on inter-document similarity or the co-occurrence patterns of terms [5]. However, existing algorithms normally employ a classifier with a fixed label set, thus are unable to address the problem of emerging new labels. Though some online setting methods, such as [6], are able to tackle this problem, each item needs to be manually labeled before update, making them unsuitable for real data streams.

Class-Incremental Learning (C-IL), which is a branch of incremental learning [7], has attracted much attention recently. The SSC-NL problem is actually a C-IL problem in social data stream context. In recent years, a number of algorithms [8–11] have been developed for classification under emerging new classes. For instance, the ECSMiner [12] tackled the novel class detection and classification problem by introducing time constraints for delayed classification. Learning with Augmented Class (LAC) [13] was proposed for identifying emerging new classes, assuming the availability of an unlabeled dataset to help identifying these new classes. In [10], an isolation-based idea was used for new class detection. However, above-mentioned methods are tailored to the single label problem, and face serious challenges in identifying new labels if instances are with multiple labels. Although, a new effort, MuENL [4], includes one classifier based on regularized SVMs and one detector based on tree structure which can tackle the SSC-NL problem, two strategies for model updating are required, resulting in a high computation cost and being hard to implement in real-time problem.

Other relevant approaches include tree-based methods to address the multi-label classification problem [14] and the anomaly detection problem [15]. Indeed, creating models to cope with environment changes [16], is widely studied in the machine learning and data mining community. Solving the SSC-NL problem can be seen as a preliminary step in social stream context.

## 3 Preliminaries

In general, social data stream is in the form of continuous streams of text data [17]. Text representation is a fundamental component to represent text



**Fig. 2.** An example of ANL and PNL. The initial label set contains four labels, the second tweet shows the ANL, and the third tweet shows PNL.

into an amenable form. Here, we denote  $\mathbf{x} \in \mathbb{R}^d$  as a vector representation for each text data by using a representation model like [18]. The SSC-NL problem therefore is defined as follows: given a set of social data as the training set  $D^T = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$ , where  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $\mathbf{y}_i \in \mathcal{Y} = \{-1, 1\}^c$  is the corresponding label vector,  $c$  is the number of labels.  $\mathbf{y}_{i,j} = 1$  iff the  $j^{\text{th}}$  label is assigned to the example  $\mathbf{x}_i$  and  $\mathbf{y}_{i,j} = -1$  otherwise. The streaming instance is from  $D^S = \{(\mathbf{x}_t, \mathbf{y}'_t)\}_{t=1}^\infty$ , where  $\mathbf{y}' \in \mathcal{Y}' = \mathcal{Y} \cup \mathcal{Y}_{\text{new}}$ ,  $\mathcal{Y}_{\text{new}} = \{-1, 1\}^a$ ,  $a > 0$ . The goal is to learn an initial model  $f$  with  $D^T$ , then  $f$  is used as a detector for emerging new labels ( $\mathcal{Y}_{\text{new}}$ ) and a classifier for known labels ( $\mathcal{Y}$ ) in the data stream. In addition,  $f$  can be updated when it maintains some criteria. The training set  $D^T$  is just used for building model at the beginning of the data stream and will then be discarded. Once update is completed, it is ready for the next instances in the data stream. Note that the model can detect instances of any number of emerging new labels, though they are grouped into one new meta-label.

Detecting emerging new labels in SSC-NL problem is a non-trivial task, because the instance with new labels is likely to contain known labels simultaneously. This is a distinct point of difference from the previous works [10, 12]. To specify the form of new label emergence in the multi-label scenario, we define two types of instances with new label as follows, and Fig. 2 shows an illustration.

**Definition 1.** [*Absolutely New Label (ANL)*]. Let  $\mathcal{Y}$  be the known label space and  $\mathcal{Y}_{\text{new}}$  be the new label space. An instance with absolutely new label is defined as  $(\mathbf{x}, \mathbf{y})$ , where  $\mathbf{y} \in \mathcal{Y}_{\text{new}}$  and  $\mathbf{y} \notin \mathcal{Y}$ .

**Definition 2.** [*Partially New Label (PNL)*]. Let  $\mathcal{Y}$  be the known label space and  $\mathcal{Y}_{\text{new}}$  be the new label space. An instance with partially new label is defined as  $(\mathbf{x}, \mathbf{y})$ , where  $\mathbf{y} \in \mathcal{Y}_{\text{new}}$  and  $\mathbf{y} \in \mathcal{Y}$ .

## 4 The Proposed Framework

In this section, we propose a novel framework named NL-Forest, which is composed of two cooperating forests. The instance-based forest (I-F) is built on the whole training data set, and the label-based forest (L-F) consists of multiple sub-forests by considering label information. The details are provided as follows.

---

**Algorithm 1.** NL-Forest construction

---

**Input:**  $D$  - input data,  $Z$  - number of trees in L-F,  
 $z$  - number of trees in L-F,  $\psi$ ,  $\phi$  - sample size.

**Output:** NL-Forest

```
1: initialize: L-F  $\leftarrow \{\}$ , L-F  $\leftarrow \{\}$ .
2: for  $i = 1, \dots, Z$  do
3:    $D_1 \leftarrow \text{sample}(D, \psi)$ 
4:   L-F  $\leftarrow$  L-F  $\cup$   $\text{Tree}(D_1)$ 
5: end for
6: for  $j = 1, \dots, c$  do
7:    $D_2 \leftarrow \{(x, y) | x \in D, y_{(j)} = 1\}$ 
8:   for  $k = 1, \dots, z$  do
9:      $D_3 \leftarrow \text{sample}(D_2, \phi)$ 
10:    L-F(j)  $\leftarrow$  L-F(j)  $\cup$   $\text{Tree}(D_3)$ 
11:  end for
12:  compute the threshold in L-F (j).
13: end for
14: return NL-Forest  $\leftarrow$  L-F  $\cup$  L-F
```

The function  $\text{Tree}(X)$

$X$  - input data,  $\text{MinSize}$  - minimum  
internal node size

```
1: if  $|X| < \text{MinSize}$  then
2:   return LeafNode $\{F[\cdot], \text{center}, v\}$ .
3: else
4:   let  $Q$  be a list of attributes in  $X$ . Randomly select an attribute  $q \in Q$  and randomly select a split point  $p$  from max and min values of attribute  $q$  in  $X$ .
5:    $X_L \leftarrow \text{filter}(X, q \leq p)$ 
6:    $X_R \leftarrow \text{filter}(X, q > p)$ 
7:   return inNode $\{\text{Left} \leftarrow \text{Tree}(X_L),$   
Right  $\leftarrow \text{Tree}(X_R)\}$ 
9: end if
```

---

### 4.1 NL-Forest: Training Process

The training process is detailed in Algorithm 1. Steps 2–5 in the left side of Algorithm 1 show the process of building L-F. The function  $\text{sample}(D, \psi)$  is defined as randomly sampling a subset with size  $\psi$  from the data set  $D$ . The function  $\text{Tree}(\cdot)$  as shown in the right side is defined as building a completely-random tree, where a partition is produced by randomly selecting an attribute and its cut-point between the minimum and maximum values in the sample. The splitting is stopped when the number of instances is less than  $\text{MinSize}$ . Note that in each node, we just record the mean of instances as “centre”, the label distribution  $F[\cdot]$ , and the average number of labels per instance as  $v$ . Steps 8–11 in the left side of Algorithm 1 show the L-F construction based on the label information, which is similar to building L-F.

In line 12, a threshold, which is used to measure new labels emerging in the data stream, is found in each sub-forest. The idea here is inspired by the model proposed in [15], wherein Liu et al. presented an isolation-based method. In the NL-Forest framework, each tree is actually built to isolate every instance from the rest of the instances in the input data set. Threshold determination is based on the fact that there exist “differences” between instances with new labels and original training instances, and thus instances with new labels are more susceptible to isolation than instances with only known labels. In other words, the instances with new labels will be isolated using fewer partitions<sup>1</sup> in a tree than instances with only known labels. To obtain threshold, because the sub-forest contains a subset of labels, we select the instances without this subset of labels in the training data to compute their average height in this sub-forest. The average height obtained will be finally used as the *threshold*.

---

<sup>1</sup> The fewer partitions means that instances with new labels are more likely to be of the shorter height in each tree.

## 4.2 NL-Forest: Deployment

Algorithm 2 describes the deployment of NL-Forest in a data stream. NL-Forest ( $\mathbf{x}$ ) produces a label vector  $(y_1, \dots, y_c, y_{new})$  in line 2 and is defined as:

$$\text{NL-Forest}(\mathbf{x}) = \begin{cases} y_1, \dots, y_c \leftarrow \text{I-F}(\mathbf{x}) \\ y_{new} \leftarrow \text{L-F}(\mathbf{x}, \mathbf{y}_{known}) \end{cases} \quad (1)$$

where  $c$  is the number of known labels and  $\mathbf{y}_{new}$  is new label predicted. In Eq. (1),  $\mathbf{x}$  falls into one node in each tree in I-F and the distribution  $F[\cdot]$  is recorded. The output of I-F is the average of label vectors in  $F[\cdot]$  as follow:

$$\text{I-F}(\mathbf{x}_{test}) = p(\mathbf{y}|\mathbf{x}_{test}) = \mathbb{E}\left[\sum_{i=1}^Z p(\mathbf{y}|\mathbf{x}_{test}, F[i])\right] \quad (3)$$

where  $p(\mathbf{y}|\mathbf{x}_{test}, F[i])$  is the output of  $i^{th}$  tree in I-F. I-F can also output an accurate number of labels by using the average results of  $v$  in each node. This is because previous works [19–21] have shown that, ensemble of completely-random trees can be successfully applied as a powerful classifier, and it is evident that the proposed method can be a classifier capable for classification task.

Equation (2) describes that L-F predicts a new label  $y_{new}$ . We first introduce a cooperating mechanism to detect instances with PNL. From the I-F outputs, we can obtain the probabilities of known labels in the form of a label vector, as indicated in Eq. (1). Thus, we generate a vector in descending order of known labels probabilities, denoted as  $\mathbf{y}_{known}$ . According to the order of labels in  $\mathbf{y}_{known}$ , we pass  $\mathbf{x}_{test}$  to the corresponding sub-forests. The function L-F( $\cdot$ ) is as follows:

$$\text{L-F}(\mathbf{x}_{test}, \mathbf{y}_{known}) = \mathbb{E}\left[\sum_{i=1}^u p(y_{new}|\mathbf{x}_{test}, y_i)\right], y_i \in \mathbf{y}_{known} \quad (4)$$

where  $p(y_{new}|\mathbf{x}_{test}, y_i)$  is the output in one sub-forest in L-F and is defined as:

$$p(y_{new}|\mathbf{x}_{test}, y_i) = \begin{cases} 1, & \text{if } \Theta(\mathbf{x}_{test}) < \text{threshold}_i \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where  $\Theta(\cdot)$  is the average height of the instance in  $i^{th}$  sub-forest. Note that each sub-forest is able to partition instances with the specific label. If  $\mathbf{x}_{test}$  contains new labels, it will be partitioned easier in this known label sub-forest, that is,  $\mathbf{x}_{test}$  will have shorter height in this sub-forest. In Eq. (4),  $\mathbf{x}_{test}$  is an instance with new label if the average height of instance  $\mathbf{x}_{test}$  in  $i^{th}$  sub-forest is less than the  $\text{threshold}_i$ . We finally use the top  $u$  labels in  $\mathbf{y}_{known}$  to predict whether new label is emerging in Eq. (3). We can use the predicted number of labels as a measure to guide the setup of  $u$ .

Because an instance with ANL is likely to differentiate from original instances in the training set, detecting ANL is equivalent to detecting new labels in single label setting. Hence, the instance with ANL can be directly isolated using fewer partitions in the I-F. Fortunately, some previous works employed random tree

---

**Algorithm 2.** NL-Forest deployment in the data stream

---

**Input:** NL-Forest,  $\mathcal{B}$  - buffer,  $s$  - buffer size.  
**Output:** prediction for each  $x$  in a data stream

```
1: while not end of data stream do
2:    $(y, y_{new}) \leftarrow \text{NL-Forest}(x)$ 
3:   if  $y_{new} > \frac{1}{2}$  then
4:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{x\}$ 
5:     NewLabel  $\leftarrow 1$ 
6:   else
7:     NewLabel  $\leftarrow 0$ 
8:   end if
9:   Output  $\{y_1, \dots, y_c, \text{NewLabel}\}$ 
10:  if  $|\mathcal{B}| \geq s$  then
11:    Update (NL-Forest,  $\mathcal{B}$ ) # detailed in Sect. 4.3
12:     $\mathcal{B} \leftarrow \text{NULL}$ 
13:  end if
14: end while
```

---

structures for new class detection and can naturally adapt to I-F. In this paper, we use the method in [10] to detect the ANL. In line 3–8 in Algorithm 2, NL-Forest outputs a positive decision if  $y_{new}$  is greater than a 0.5 threshold. This threshold corresponds intuitively to majority voting.

Model is updated when buffer  $\mathcal{B}$  is full ( $|\mathcal{B}| \geq s$ ) in line 10–13 in Algorithm 2. The size of buffer  $s$  is a user-defined parameter and can be set based on the memory space available<sup>2</sup>. Similar to [11], we only need to manually annotate instances with the true label in the buffer instead of labeling all instances in the data stream. In the following, we introduce two growing mechanisms.

### 4.3 NL-Forest: Model Update

**Growing a subtree in I-F.** Updating I-F is to update each leaf node in every tree using a random sample of size  $\lambda$  from  $\mathcal{B}$ . The update at each node involves either (1) a replacement with a simple update label distribution  $F[\cdot]$  to include the new label  $y_{c+1}$  or (2) a newly grown subtree if the total number of instances falling into the same leaf node exceeds the limit. At each node, growing a subtree needs to generate pseudo instances in each node which have the same attribute-values as “centre”. The number of pseudo instances is as recorded in  $F[\cdot]$ . The combined set of pseudo instances and the randomly selected instances which fall into this leaf node is used as input to build the subtree.

**Growing a new sub-forest in L-F.** A new sub-forest can be constructed using instances with the new label from  $\mathcal{B}$ . Once the new sub-forest is completed, a *threshold* is calculated as mentioned in Sect. 4.1 by using pseudo instances.

### 4.4 Model Complexity

In the training stage, the overall time complexity to construct random tree is  $O(Z\psi \log \psi + cz\phi \log \phi)$ . To predict an instance in the stream, it takes  $O(Z \log \psi +$

---

<sup>2</sup> This is a trade-off parameter, the larger means method needs more memory. In practise, we use the value which is greater than  $\psi$  to guide the setup of this parameter.



$uz \log \phi$ ) time to traverse each of the  $Z$  trees in l-F and  $z$  trees in  $u$  L-F. During the update, growing a subtree using a  $s$  size buffer takes  $O(Zs \log s)$  and growing a sub-forest takes  $O(z\phi \log \phi)$ . The total space required includes the buffer with size  $s$  and all centres and label distribution in leaf nodes in l-F. Thus, the space complexity is  $O(s + Zd\psi)$ .

## 5 Experiment

### 5.1 Experimental Setup

**Data Sets.** A summary of the data characteristics is provided in Table 1. The real streaming data is collected from *Sina Weibo*. This stream is about 220k items with 10 labels, and each item is preprocessed using *word2vec*<sup>3</sup> to produce a 300-dimension feature vector.

**Competing Algorithms.** Table 2 is a complete list of the methods used for new label detection and known label classification. It includes two multi-label supervised classifiers – binary relevance SVM (BR-SVM) [22], ML-KNN [23]; one supervised multi-label streaming classifier – SMART [6]; an existing solution for emerging new labels – MuENL [4]; an outlier detector as new label detector – iForest [15].

**Experiment Settings and Evaluation Metrics.** All methods are executed in the MATLAB environment with the following implementations: SVM is in the LIBSVM package [22]; MuENL, iForest and ML-KNN are the codes as released by the corresponding authors; SMART code is developed based on the original paper [6]. In NL-Forest, we set  $Z = 200$ ,  $z = 100$ ,  $\psi$  and  $\phi$  are set by  $0.6 * m$  and  $0.6 * n_i$ , where  $m$  and  $n_i$  are the sizes of  $D_1$  and  $D_2$  respectively.  $\lambda$  is set according to label balance in each tree. The trees stop growing when the total number of instances, which fall into a leaf node, exceeds the limit, e.g.,  $MinSize = 10$  in the simulated streams and  $MinSize = 100$  in the real stream. BR-SVM trains a linear classifier for each label independently and parameters are set according to cross validation. In ML-KNN,  $K$ , the number of nearest neighbors is set as 10. In SMART, the tree height is  $h = 30$ , and the number of

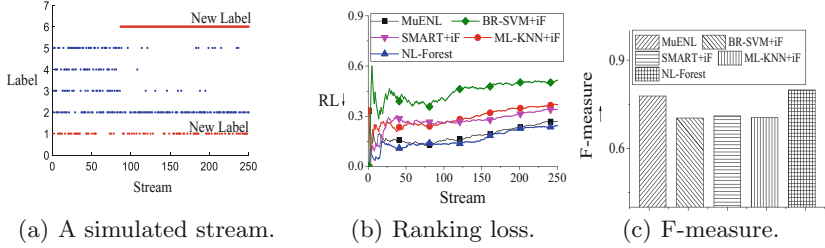
**Table 1.** A summary of data sets.

	Emotions	Yeast	Enron	Weibo
#Attributes	72	103	1001	300
#Labels	6	14	53	10
Volume	593	2417	1702	220K

**Table 2.** Methods used in the experiments.

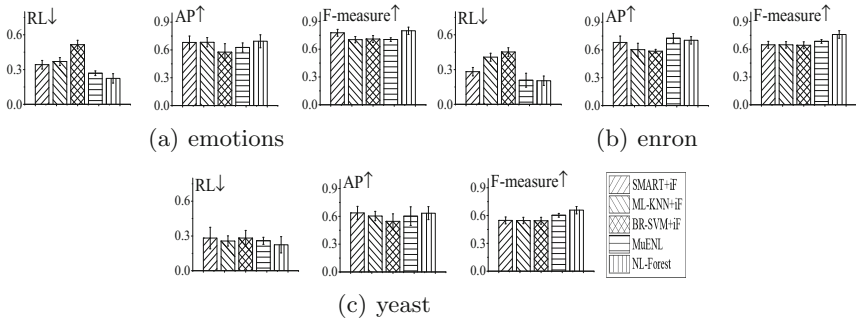
Method	Detector	Classifier
BR-SVM+iF	iForest	BR-SVM
ML-KNN+iF	iForest	ML-KNN
SMART+iF	iForest	SMART
MuENL	MuENLForest	MuENL <sub>MNL</sub>
NL-Forest	NL-Forest	

<sup>3</sup> <https://radimrehurek.com/gensim/index.html>.



**Fig. 3.** A simulated stream on *emotions* data set. (a) There are six labels, the blue points represent the known labels, and the red points represent the new labels. (b) and (c) the results of RL and F-measure. (Color figure online)

trees in the ensemble is  $n_t = 30$ . The number of trees in iForest is  $n_t = 50$ . In MuENL, the parameters in classification model are selected via cross validation, and the setup of detection model is same as iForest. We employed **Ranking Loss (RL ↓<sup>4</sup>)** and **Average Precision (AP ↑)** for classification performances and **F-measure (↑)** for detection results.



**Fig. 4.** Results of the simulated data streams.

## 5.2 Simulated Data Stream

To evaluate different scenarios under which new labels appear, we perform experiments in multiple simulated data streams which are generated from three benchmark multi-label data sets. In each data set, we first randomly select two labels as the new labels. The instances with any of these two labels are selected as set  $A$ , and the rest will be randomly divided into training set (80%) and testing set (20%). Then  $A$  is added to the testing set, and we simulate a steam by using the testing set. RL and AP are computed over the entire stream, F-measure is

<sup>4</sup> Here “↓” means the smaller the value, the better the performance; and “↑” means the larger the value, the better the performance.

computed when the buffer is full. An example is shown in Fig. 3. The simulation is repeated 30 times for each data set, the average results are reported in Fig. 4.

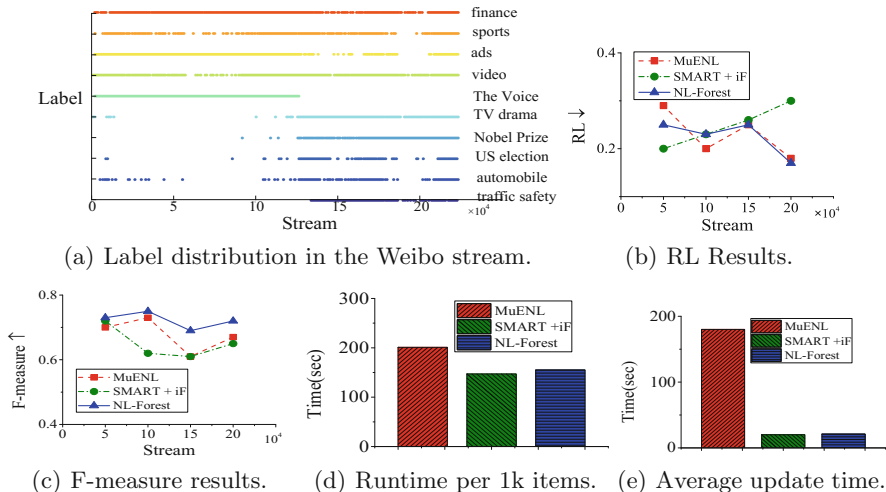
**Detailed Analysis.** The proposed NL-Forest has consistently produced better performance in all three data sets than any other methods for both emerging new labels detection and known labels classification. In terms of new label detection, NL-Forest produces higher F-measure than methods directly using an anomaly detector (i.e., iForest) which do not consider differences between new labels and anomaly when detecting the PNL. In terms of classification results (RL and AP), NL-Forest gives results comparable with state-of-the-art methods, e.g., MuENL<sub>MNL</sub> and ML-KNN. What has greatly contributed to the practical values of NL-Forest is the fact that it can be applied to a wide range of prediction problems and has fewer parameters to tune.

Compared to NL-Forest, MuENL consists of two independent models, i.e., a detector for new labels and a classifier for known labels. Despite its reasonable good classification and detection performances in simulations, it is not a good choice for the SSC-NL problem due to its high computational complexity in model update and practical difficulty in parameter determination. On the other hand, ML-KNN, BR-SVM and SMART are state-of-the-art multi-label classification methods for known labels, but they still require another framework to detect new labels. In addition, BR-SVM often comes at high computational costs in an extensive parameter search.

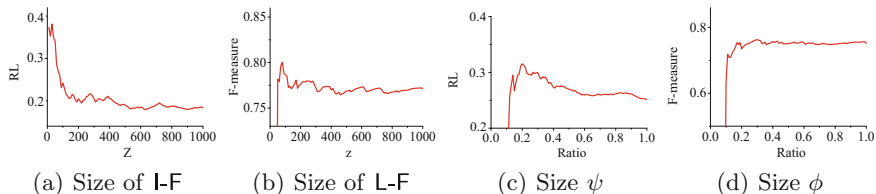
### 5.3 Real Data Stream

In this section, we conduct experiments on a social stream and compare the proposed method with SMART+iForest and MuENL which focus on the streaming data problem. Figure 5(a) indicates label distribution in the stream. For convenience, we use “1<sup>st</sup>” to “10<sup>th</sup>” to represent the category “traffic safety” to “finance”. We regard five labels (6<sup>th</sup> to 10<sup>th</sup>) as known labels and collect extra 15K instances with them to initialize the model. The 1<sup>st</sup> to 5<sup>th</sup> labels are regarded as new labels which occur in the different periods. To be specific, the 2<sup>nd</sup>, 3<sup>rd</sup> and 5<sup>th</sup> label emerge at around point 0 to 50K; at around point 50K to 100K, the 4<sup>th</sup> label appears; the 1<sup>st</sup> label emerges at last 100K points. Note that when the buffer is full, model will be updated using buffer data with true label. Evaluation metrics are computed at different time points as shown in Figs. 5(b) and (c).

Figure 5(c) shows NL-Forest outperforms other methods in detecting new labels, and NL-Forest gives comparable results with other methods in classification in Fig. 5(b). We also show the time of processing 1000 data items and the average update time in the stream. In Fig. 5(d), the proposed method achieves the shorter running time than MuENL for the real data stream, and is comparable to the state-of-the-art method SMART, which also employs completely random trees. Figure 5(e) shows the proposed method can be more efficient in deploying in the real application with the faster update.



**Fig. 5.** Results of the real data stream.



**Fig. 6.** Results of the sensitivity of parameters.

## 5.4 Sensitivity of Parameters

We study the influences of parameters in NL-Forest, i.e.,  $z$  and  $Z$  (the number of trees),  $\psi$  and  $\phi$  (the sampling sizes). We evaluate NL-Forest on the *emotions* data set with different settings of one parameter while the other parameters are fixed. Figures 6(a) and (b) show that the performance of NL-Forest is stable when we set the size of tree greater than 100. Therefore, in practice, model parameter setup can follow such guidelines. In Figs. 6(c) and (d), the X-axis represents a ratio between the sample size and original data size. Generally, the larger each random tree is, the better the performance is, but larger trees will consume memory. We observe that the RL or F-measure of NL-Forest converges at a small  $\psi$  or  $\phi$ . Hence, the ratio set by half is safe and recommended in practise. Note that similar results are also observed on the other data sets.

## 6 Conclusion

This paper introduces a novel framework with an instance-based model and a label-based model to address the SSC-NL problem. The strength of NL-Forest is

that the completely-random trees are used as a single core to effectively tackle emerging new labels detection and known labels classification, and provide the solution to efficient update. Evaluations on simulated streams and a real-world stream demonstrate the effectiveness of the proposed framework. In the future, the broader stream classification problem in real-world applications [24] including detection of concept drift, issues with outdated data, adaptation to the current state, and recurring contexts will be considered. It is also in our interest to explore the theoretical foundation for our model and extend the idea of this work to Multi-Instance Multi-Label learning (MIML) [25].

**Acknowledgement.** This research was supported by the National Research Foundation, Prime Minister’s Office, Singapore under its International Research Centres in Singapore Funding Initiative; the NSFC (61333014) and Pinnacle lab for analytics at Singapore Management University.

## References

1. Lee, K., Palsetia, D., Narayanan, R., Patwary, M.M.A., Agrawal, A., Choudhary, A.N.: Twitter trending topic classification. In: ICDM Workshops, pp. 251–258 (2011)
2. Tsai, M., Aggarwal, C.C., Huang, T.S.: Towards classification of social streams. In: SDM (2015) 649–657
3. Zubiaga, A., Spina, D., Martínez-Unanue, R., Fresno, V.: Real-time classification of twitter trends. *JASIST* **66**(3), 462–473 (2015)
4. Zhu, Y., Ting, K.M., Zhou, Z.H.: Multi-label learning with emerging new labels. In: ICDM, pp. 1371–1376 (2016)
5. Blei, D.M.: Probabilistic topic models. *Commun. ACM* **55**(4), 77–84 (2012)
6. Kong, X., Yu, P.S.: An ensemble-based approach to fast classification of multi-label data streams. In: 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom, pp. 95–104 (2011)
7. Zhou, Z.H., Chen, Z.Q.: Hybrid decision tree. *Knowl. Based Syst.* **15**(8), 515–528 (2002)
8. Al-Khateeb, T., Masud, M.M., Khan, L., Aggarwal, C., Han, J., Thuraisingham, B.: Stream classification with recurring and novel class detection using class-based ensemble. In: ICDM, pp. 31–40 (2012)
9. Mu, X., Zhu, F., Du, J., Lim, E.P., Zhou, Z.H.: Streaming classification with emerging new class by class matrix sketching. In: AAAI, pp. 2373–2379 (2017)
10. Mu, X., Ting, K.M., Zhou, Z.H.: Classification under streaming emerging new classes: A solution using completely-random trees. *IEEE TKDE* **29**(8), 1605–1618 (2017)
11. Haque, A., Khan, L., Baron, M.: Sand: Semi-supervised adaptive novel class detection and classification over data stream. In: AAAI, pp. 1652–1658 (2016)
12. Masud, M., Gao, J., Khan, L., Han, J., Thuraisingham, B.M.: Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE TKDE* **23**(6), 859–874 (2011)
13. Da, Q., Yu, Y., Zhou, Z.H.: Learning with augmented class by exploiting unlabeled data. In: AAAI, pp. 1760–1766 (2014)

14. Liu, F., Zhang, X., Ye, Y., Zhao, Y., Li, Y.: MLRF: multi-label classification through random forest with label-set partition. In: Huang, D.-S., Han, K. (eds.) ICIC 2015. LNCS (LNAI), vol. 9227, pp. 407–418. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-22053-6\\_44](https://doi.org/10.1007/978-3-319-22053-6_44)
15. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: ICDM, pp. 413–422 (2008)
16. Zhou, Z.H.: Learnware: on the future of machine learning. *Front. Comput. Sci.* **10**(4), 355–384 (2016)
17. Aggarwal, C.C.: Mining text and social streams: a review. *SIGKDD Explor.* **15**(2), 9–19 (2013)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
19. Fan, W., Wang, H., Yu, P.S., Ma, S.: Is random model better? on its accuracy and efficiency. In: ICDM, pp. 51–58 (2003)
20. Liu, F.T., Ting, K.M., Fan, W.: Maximizing tree diversity by building complete-random decision trees. In: Ho, T.B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 605–610. Springer, Heidelberg (2005). <https://doi.org/10.1007/11430919-70>
21. Zhou, Z.H.: *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, Boca Raton, FL, USA (2012)
22. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2**, 27:1–27:27 (2011). <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
23. Zhang, M.L., Zhou, Z.H.: ML-KNN: a lazy learning approach to multi-label learning. *Pattern Recogn.* **40**(7), 2038–2048 (2007)
24. Gama, J.: *Knowledge Discovery from Data Streams*. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press, Boca Raton (2010)
25. Zhou, Z.H., Zhang, M.L., Huang, S.J., Li, Y.F.: Multi-instance multi-label learning. *Artif. Intell.* **176**(1), 2291–2320 (2012)