Continual Rare-Class Recognition with Emerging Novel Subclasses

Hung Nguyen ✉ Xuejian Wang Leman Akoglu

Carnegie Mellon University
Heinz College of Information Systems and Public Policy
{hungnguy, xuejianw, lakoglu}@andrew.cmu.edu

Abstract. Given a labeled dataset that contains a rare (or minority) class of *of-interest* instances, as well as a large class of instances that are *not* of interest, how can we learn to recognize future *of-interest* instances over a continuous stream? We introduce RaRecognize, which (*i*) estimates a *general* decision boundary between the rare and the majority class, (*ii*) learns to recognize individual rare subclasses that exist within the training data, as well as (*iii*) flags instances from previously unseen rare subclasses as newly emerging. The learner in (*i*) is general in the sense that by construction it is dissimilar to the *specialized* learners in (*ii*), thus distinguishes minority from the majority without overly tuning to what is seen in the training data. Thanks to this generality, RaRecognize ignores all future instances that it labels as majority and recognizes the recurrent as well as emerging *rare* subclasses only. This saves effort at test time as well as ensures that the model size grows moderately over time as it only maintains specialized minority learners. Through extensive experiments, we show that RaRecognize outperforms state-of-the art baselines on three real-world datasets that contain corporate-risk and disaster documents as rare classes.

1 Introduction

Given a labeled dataset containing (1) a rare (or minority) class of *of-interest* documents, and (2) a large set of *not-of-interest* documents, how can we learn a model that can effectively identify future *of-interest* documents over a continuous stream? Different from the traditional classification setup, the stream might contain *of-interest* (as well as *not-of-interest*) documents from *novel subclasses that were not seen in the training data.* Therefore, the model is required to continually recognize both the recurring as well as the emerging instances from the underlying rare class distribution.

Let us motivate this setting with a couple of real-world examples. Suppose we are given a large collection of social media documents (e.g. Twitter posts). A subset of the collection is labeled as *risky*, indicating posts that constitute (financial, reputational, etc.) risk to a corporation. The rest (majority) of the collection is *not-risky.* The goal is then to learn a model that can continually identify future posts that are *risky* over the social-media stream. Here, the rare class contains *risky* documents of a few known types, such as bankruptcy, corruption, and spying. However, it is unrealistic to assume that it contains examples from all possible risk types—given the large spectrum, labeling effort, and potentially evolving nature of risk.

Consider another case where the training set consists of news articles. A subset of the articles belongs to the rare class of *disasters*, indicating news about natural or man-made disasters. The rest are *not-disaster* articles. Similar to the first case, the rare class might contain articles about floods, earthquakes, etc. however it is hard to imagine it would contain instances from all possible types of disasters. The goal is to learn to continually recognize future articles on disasters.

In both examples above, the model needs to learn from and generalize beyond the labeled data so as to recognize future rare-class instances, both from *recurring* (i.e., seen in the training data) as well as from *novel subclasses*; for instance sexual assault, cyber attack, etc. in risk domain and explosions, landslides, etc. in disasters domain. In machine learning terms, this is a very challenging setup in which the learner needs to generalize not only to unseen instances but also to *unseen distributions*. In other words, this setting involves test data that has a related yet different distribution than the data the model was trained on.

The stream classification problem under emerging novel classes has been studied by both machine learning and data mining communities. The area is referred to under various names including open-world classification [13,14], life-long learning [1], and continual learning [12]. In principle, these build a "never-ending learner" that can (1) assign those recurring instances from known old classes to their respective class, (2) recognize emerging classes, and (3) grow/extend the current learner to incorporate the new class(es). The existing methods differ in terms of accuracy-efficiency trade-offs and various assumptions that they make. (See Section 5 for detailed related work.) A common challenge that all of them face is what is known as *catastrophic forgetting*, mainly due to model growth. In a nutshell, the issue is the challenge of maintaining performance on old classes as the model is constantly grown to accommodate the new ones.

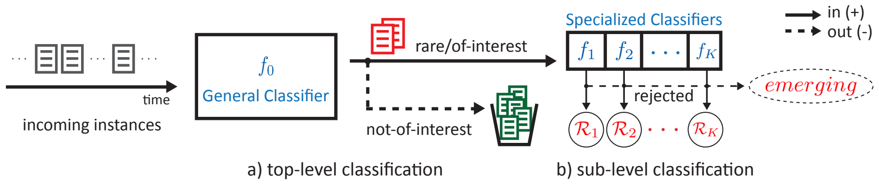

time
incoming instances
f_0
General Classifier
rare/of-interest
not-of-interest
Specialized Classifiers
f_1 f_2 $\cdots$ f_K
rejected
emerging
$\mathcal{R}_1$ $\mathcal{R}_2$ $\cdots$ $\mathcal{R}_K$
in (+)
out (-)
a) top-level classification
b) sub-level classification

Fig. 1: An illustration of the recognition flow in our proposed model.

Our work is different from all prior work in one key aspect: our goal is not to recognize *any and every* newly emerging class—but only those (sub)classes related to the rare class *of-interest*. That is, our primary goal is to recognize rare-class instances. *Not-of-interest* instances, as long as they are filtered out accurately, are ignored—*no matter they are recurrent or novel*, as depicted in Fig. 1. This way, we carefully avoid the aforementioned issue that current models face. Our model grows slowly, only when novel *rare* subclasses are recognized.

Thanks to a moderate model size (by definition, rare subclasses are far fewer), our model is not only less prone to catastrophic forgetting but also (a) is faster at test time, and (b) requires much less memory.

We summarize the main contributions of this work as follows.

- **Problem and Formulation:** We address the problem of recognizing instances from a rare, *of-interest* class over a stream continually. The setting differs from traditional (binary) classification in that the data distribution (for both rare and majority class) might change over time, where novel subclasses emerge. We formulate a new model called RaRecognize that *simultaneously* learns (*i*) a separate specialized classifier (SC) that recognizes an individual rare subclass, as well as (*ii*) a general classifier (GC) that separates rare instances from the majority. While being discriminative, GC is constructed to be dissimilar to the individual SCs such that it can generalize without overly tuning to seen rare subclasses in the training data.
- **Efficient Algorithm:** Our proposed solution exhibits two key properties: runtime and memory efficiency; both essential for the stream setting. Given a new instance that GC labels as belonging to the majority class, we simply do nothing—no matter it is recurrent or emerging. By not processing the majority of the incoming instances, we achieve *fast response time*. Moreover RaRecognize remains compact, i.e. *memory-efficient*, as it requires space linear in the number of *rare* subclasses which only grows slowly.
- **Applications:** Recognizing recurrent as well as novel instances that belong to a certain class *of-interest* is a broad problem that finds numerous applications, e.g. in monitoring and surveillance. For example, such instances could be production-line items with the goal to continually recognize faulty ones where novel fault types might emerge over time. They could also be public documents, such as social media posts, where the goal is to recognize public posts *of-interest* such as bullying, shaming, disasters, threat, etc.

Reproducibility: We share the source code for RaRecognize and our public-domain datasets at https://github.com/hungnt55/RaRecognize.

2 Problem Setup and Preliminary Data Analysis

Problem Setup and Overview. We start by introducing the problem statement more formally with proper notation. As input, a labeled training dataset $\mathcal{D} = \mathcal{R} \cup \mathcal{N} \in \mathbb{R}^{n\times d}$ containing n d-dimensional instances is provided. The set $\mathcal{R} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_{n_0}, y_{n_0})\}$ consists of $|\mathcal{R}| = n_0$ instances belonging to the *of-interest* rare class where $y_i = +1$ for $i = 1, \ldots, n_0$ and the set $\mathcal{N} = \{(\mathbf{x}_{n_0+1}, y_{n_0+1}), \ldots, (\mathbf{x}_n, y_n)\}$ consists of $|\mathcal{N}| = (n - n_0)$ instances from the *not-of-interest* class where $y_i = -1$ for $i = (n_0 + 1), \ldots, n$. Without loss of generality, we will refer to the data instances as documents and to the rare class as the *risk* class in the rest of this section to present our ideas more concretely.

Given $\mathcal{D}$, the goal is to recognize future *risk* documents, *either recurring or newly emerging*, over a stream (or set) of new documents $\mathbf{x}_{n+1}, \mathbf{x}_{n+2}, \ldots$ (here, each document has a vector representation denoted by $\mathbf{x}$ such as bag-of-words, embedding, etc.). The new documents may associate with recurring risk, i.e., belong to known/seen risk subclasses $1, \ldots, K$ in $\mathcal{R}$. They may also be emerging, i.e., from previously unknown/unseen new risk subclasses $(K+1), (K+2), \ldots$; which differentiates our setup from the traditional classification problem.

Therefore, we start by decomposing $\mathcal{R}$ into known risk subclasses, $\mathcal{R} = \bigcup_{k=1}^{K} \mathcal{R}_k$, where $\mathcal{R}_k$ contains the documents that belong to the kth risk subclass. Given $\{\{\mathcal{R}_1, \ldots, \mathcal{R}_K\}, \mathcal{N}\}$ our approach involves simultaneously training the following two types of classifiers:

1. A *general* classifier (GC) f_0 to separate $\mathcal{R}$ and $\mathcal{N}$ that can generalize to unseen subclasses of $\mathcal{R}$,
2. A *specialized* classifier (SC) f_k, $k = 1 \ldots K$, to separate $\mathcal{R}_k$ and $\mathcal{R} \backslash \mathcal{R}_k$.

At test time, we first employ f_0. Our goal is not to recognize every emerging novel class, but only the novel risk subclasses (in addition to recurring ones), thus our first step is to recognize risk. If f_0 labels an incoming document $\mathbf{x}$ as -1 (i.e., not-risk), we discard it. Otherwise, the incoming document is flagged as risky. For only those labeled as $+1$, we employ f_k's to further identify the type of risk. Among the f_k's that accept $\mathbf{x}$ as belonging to the kth risk subclass, we assign it to the subclass that is $\arg\max_k f_k(\mathbf{x})$. If all f_k's reject, then $\mathbf{x}$ is considered to be associated with a new type of emerging risk. (See Fig. 1.)

The classifier models. Our risk detector is f_0 which we learn using the entire labeled dataset $\mathcal{D}$. As such, it is trained on a few known risk subclasses in $\mathcal{R}$ but is desired to be *general* enough to recognize other types of future risk.

To achieve this generality, our main idea is to avoid building f_0 on factors that are too specific to any known risk subclass (such that f_0 is not overly fit to existing or known risk types) but rather, to identify broad factors about risk that are *common* to all risk subclasses (such that f_0 can employ this broader view to spot risk at large).

In fact, factors specific to the known risk subclasses are to be captured by the corresponding f_k's. Then, f_0 is to identify discriminative signals of risk that are sufficiently different from those used by all f_k's. Moreover, each f_k should differ from other $f_{k'}$'s, $k' \neq k$, to ensure that they are as *specialized* as possible to their respective risk types. Such dependence among the models is exactly why we train all these $(K+1)$ classifiers *simultaneously*, to enforce the aforementioned constraints conjointly. We present our specific model formulation and optimization in Section 3.

Preliminary Data Analysis. Before model formulation, we perform an exploratory analysis on one of our real-world datasets containing documents labeled as risky and not-risky. The goal of the analysis is to see if our hypothesized ideas get realized in the data.

In particular, we aim to find out if there exists (1) factors that are specific to each risk subclass, as well as (2) factors beyond those specific ones that are still discriminative of risk. For simplicity and interpretability, we use the bag-of-words representation of the data in this section, thus factors correspond to individual words. However, our proposed model can handle other document vector representations in general.

To this end, we formulate a constrained optimization problem to find word sets that cover or characterize different document sets. Here, we define a word to *cover* a document if the word appears in it at least once. Given the set of unique words $\mathcal{V}$, $|\mathcal{V}| = d$, we look for a set of words $\mathcal{V}_k \subset \mathcal{V}$ that covers all the documents in $\mathcal{R}_k$ but as few as those in $\mathcal{R} \backslash \mathcal{R}_k$ for all $k = 1 \dots K$ (i.e., specific words for each risk subclass), and another set of words $\mathcal{V}_0 \subset \mathcal{V}$ that covers all risk documents in $\mathcal{R}$ but as few as those in $\mathcal{N}$. We restrict the word sets to be *non-overlapping*, i.e., $\mathcal{V}_k \cap \mathcal{V}_{k'} = \emptyset \;\; \forall k, k' \in \{0, \dots, K\}$, such that each word can only characterize either one of $\mathcal{R}_1, \dots, \mathcal{R}_K$ or $\mathcal{R}$ at large. Under these conditions, if we could find a set $\mathcal{V}_0$ that shares no words with any $\mathcal{V}_k$'s while still being able to cover the risky documents but only a few (if at all) not-risky ones, then we can conclude that broad risk terms exist and a *general* f_0 can be trained.

Our setup is a constrained mixed-integer linear program (MILP) as follows:

$$
\begin{aligned}
\min_{\boldsymbol{\Phi}} \quad & |\mathbf{v}_0| + \sum_{k=1}^{K} |\mathbf{v}_k| + o + \alpha + \beta \\
\text{s.t.} \quad & \mathbf{z}_k + \mathbf{R}_k \cdot \mathbf{v}_k \geq \mathbf{1} \quad \forall k = 1 \dots K && \triangleright \text{ risk subclass coverage with exoneration} \\
& \mathbf{z}_0 + \sum_{k=1}^{K} \mathbf{R}_k \cdot \mathbf{v}_0 \geq \mathbf{1} && \triangleright \text{ risk coverage with exoneration} \\
& |\mathbf{z}_0| + \sum_{k=1}^{K} |\mathbf{z}_k| \leq o && \triangleright \text{ \# unexplained documents less than } o \\
& \mathbf{v}_0 + \sum_{k=1}^{K} \mathbf{v}_k \leq \mathbf{1} && \triangleright \text{ each word used for at most 1 set} \\
& \sum_{i \in \mathcal{N}} \mathbf{N}_i \cdot \mathbf{v}_0 \leq \alpha && \triangleright \text{ cap on cross-coverage of not-risk} \\
& \sum_{k=1}^{K} \sum_{i \in \mathbf{R}_k} \sum_{k' \neq k} \mathbf{R}_{k,i} \cdot \mathbf{v}_{k'} \leq \beta && \triangleright \text{ cap on cross-coverage among risk subclasses}
\end{aligned}
$$

The program is parameterized by $\boldsymbol{\Phi} = \{\{\mathbf{v}_k\}_{k=0}^{K}, \{\mathbf{z}_k\}_{k=0}^{K}, o, \alpha, \beta\}$. $\mathbf{R}_k \in \mathbb{R}^{n_k \times d}$ denotes the data matrix encoding the word occurrences for n_k documents in risk subclass $k = 1 \dots K$, and $\mathbf{N} \in \mathbb{R}^{(n - n_0) \times d}$ is the corresponding data matrix for the not-risk documents. $\mathbf{v}_0 \in \mathbb{R}^d$ and $\mathbf{v}_k \in \mathbb{R}^d$'s depict (binary) variables to be estimated that capture the word assignments to the sets $\mathcal{V}_0$ and $\mathcal{V}_k$'s respectively. (e.g., jth entry of $\mathbf{v}_0$ is set to 1 if word j is assigned to $\mathcal{V}_0$ and to 0 otherwise.)

Ideally all of o, α, and β are zero; that is, all documents are covered without any exoneration and no cross-coverage exists. However, that yields no feasible solution. Instead, we define them as scalar upper-bound variables added to our minimization objective toward setting them to as small values as possible. Finally, our objective aims to find the smallest-size possible word sets. This ensures that the most important words are selected which also facilitates interpretability.

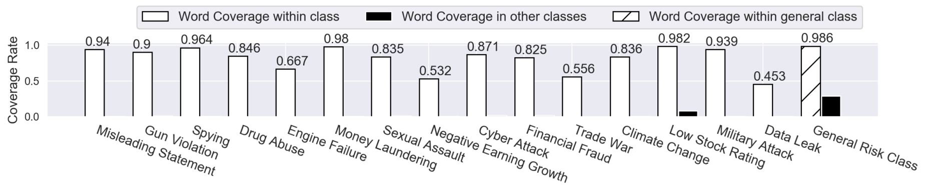
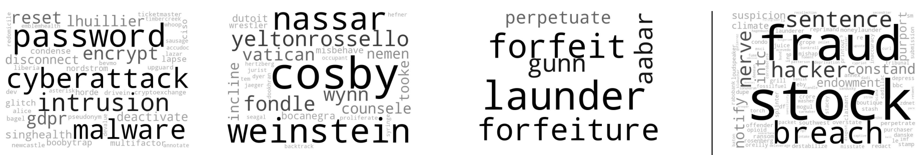


Fig. 2: Within- and cross-coverage rates of $\mathcal{V}_k$'s and $\mathcal{V}_0$ (resp.) for $\mathcal{R}_k$'s and $\mathcal{R}$.



($\mathcal{V}_1$) Cyber attack ($\mathcal{V}_2$) Sexual assault ($\mathcal{V}_3$) Money laundering ($\mathcal{V}_0$) General risk

Fig. 3: Wordclouds representing 3 example risk subclasses and the overall risk class. Notice that the former are quite specific, and the latter are broader.

We provide an exploratory analysis on a dataset containing corporate risk documents as the *of-interest* class. It contains 15 risk subclasses as outlined by Fig. 2. (See Sec. 4.1 for details.) First the quantitative measures: as shown in the figure, the MILP finds word sets $\mathcal{V}_k$'s with at least 82.5% up to 98.2% coverage for 11/15 of the subclasses with an overall coverage of 96.7% (rest are the exonerated ones). Moreover, cross-coverage is either zero or very low for all the subclasses. These suggest that accurate SCs can be learned. Importantly, there exist words $\mathcal{V}_0$ that are *distinct* from all $\mathcal{V}_k$'s and yet able to cover 98.6% of the overall risk documents, promising that a broad GC can be learned.

To equip the reader with intuition, we present the selected words for 3 example risk subclasses along with the general risk class words in Fig. 3 (word size is proportional to the within vs. cross-coverage ratio). It is easy to see that very specific words are selected for subclasses; such as `password`, `cyberattack`, `malware` for Cyber attack, and `cosby`, `weinstein`, `fondle` for Sexual assault. On the other hand, in the general risk class, a set of broader corporate risk words appear such as `fraud`, `stock`, `breach` and `sentence`.

These preliminary results show promise for the feasibility of our hypothesized models and demonstrate the rationale behind our proposed RaRecognize, which we formally introduce next.

3 Continual Rare-Class Recognition

In this section, we introduce the individual components of our model, present the underlying reasoning for our formulation, show convexity and present the optimization steps, and conclude with space and time-complexity analysis.

3.1 Model Formulation

As discussed in the previous section, our goal is to learn (1) specialized classifiers f_k's and (2) a general classifier f_0.

The **specialized classifier** f_k, $k = 1 \ldots K$, is to learn a decision boundary that separates the kth rare subclass instances $\mathcal{R}_k$ from the remainder of rare instances $\mathcal{R} \backslash \mathcal{R}_k$. Let us write down the regularized loss function for each f_k as

$$\mathcal{L}(f_k; \mathbf{w}_k, b_k) = \sum_{i=1}^{n_0} \underbrace{\max\left(0, \left[1 - y_i(\mathbf{w}_k^T \mathbf{x}_i + b_k)\right]\right)}_{\ell(\mathbf{x}_i, y_i; \mathbf{w}_k, b_k)} + \frac{\lambda_k}{2} \|\mathbf{w}_k\|^2 \qquad (1)$$

where $y_i = +1$ for $\mathbf{x}_i \in \mathcal{R}_k$ and $y_i = -1$ otherwise. We adopt the hinge loss and the ridge regularization as in Eq. (1), however, one could instead use other loss functions, such as the logistic, exponential or cross-entropy losses, as well as other norms for regularization.

The **general classifier** f_0 is to separate rare class instances $\mathcal{R}$ from the majority instances $\mathcal{N}$, without relying on factors specific to known rare subclasses. One way to achieve this *de-correlation* is to enforce f_0 to learn coefficients $\mathbf{w}_0$ that are different from all $\mathbf{w}_k$'s. The loss function can be written as

$$\mathcal{L}(f_0; \mathbf{w}_0, b_0) = \sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i; \mathbf{w}_0, b_0) + \frac{\lambda_0}{2} \|\mathbf{w}_0\|^2 + \frac{\mu}{2} \sum_{k=1}^{K} \mathbf{w}_0^T \mathbf{w}_k \ , \qquad (2)$$

where $y_i = +1$ for $\mathbf{x}_i \in \mathcal{R} = \{\mathcal{R}_1 \cup \ldots \cup \mathcal{R}_K\}$ and $y_i = -1$ otherwise. As required, the third term in Eq. (2) penalizes $\mathbf{w}_0$ being correlated with any $\mathbf{w}_k$, enforcing it to be as orthogonal to $\mathbf{w}_k$'s as possible. However, it does not prevent $\mathbf{w}_0$ from capturing *different yet correlated features* to those captured by $\mathbf{w}_k$'s. This issue can arise when features exhibit multi-collinearity.

For example, in a document dataset the words `earthquake`, `shockwave`, and `aftershock` could be collinear. In this case it is possible that f_k estimates large coefficients on a strict subset of these words (e.g., `shockwave` and `aftershock`) as they are redundant. This leaves room for f_0 to capitalize on the remaining words (e.g., `earthquake`), which is undesirable since we aim f_0 to learn about the rare class boundaries beyond the specifics of the known subclasses.

Therefore, we reformulate the model correlation penalty as follows.

$$\frac{\mu}{2} \sum_{k=1}^{K} \sum_{p,q} \left(w_{0,p} w_{k,q} \ \mathbf{x}_{[p]}^T \mathbf{x}_{[q]}\right)^2 = \frac{\mu}{2} \left\| (\mathbf{X}^T \mathbf{X}) \odot (\mathbf{w}_0 \mathbf{w}_k^T) \right\|_F^2 \ , \qquad (3)$$

where $w_{0,p}$ and $w_{k,q}$ denote the pth and qth entries of $\mathbf{w}_0$ and $\mathbf{w}_k$ respectively, $\mathbf{X} \in \mathbb{R}^{n \times d}$ denotes the input data matrix, $\mathbf{x}_{[p]}, \mathbf{x}_{[q]}$ respectively denote the pth and qth columns of $\mathbf{X}$, and $\odot$ depicts the element-wise multiplication.

We call Eq. (3) the cross-correlation penalty. Similarly, we introduce self-correlation penalty to each model $k = 0, \ldots, K$ by adding to the respective loss the term $\sum_{p,q} \left(w_{k,p} w_{k,q} \ \mathbf{x}_{[p]}^T \mathbf{x}_{[q]}\right)^2$. Self-correlation prevents each model from estimating large coefficients on higly correlated (near-redundant) features, which improves sparsity and interpretability, and as we show also ensures convexity.

Then, the overall loss function incorporating the cross- and self-correlation penalty terms for all models $f_0, f_1, \ldots, f_K$ is given as follows.

$$\boxed{\begin{aligned}\mathcal{L} = \sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i; \mathbf{w}_0, b_0) + \frac{\lambda_0}{2}\|\mathbf{w}_0\|^2 + \sum_{k=1}^{K}\Bigg[\sum_{i=1}^{n_0}\ell(\mathbf{x}_i, y_i; \mathbf{w}_k, b_k) + \frac{\lambda_k}{2}\|\mathbf{w}_k\|^2\Bigg] \\ + \frac{\mu}{2}\sum_{p,q}\Bigg\{\underbrace{\frac{1}{2}(w_{0,p}^2 w_{0,q}^2) + \frac{1}{2}\sum_{k=1}^{K}(w_{k,p}^2 w_{k,q}^2)}_{\text{self-correlation}} + \underbrace{w_{0,p}^2(\sum_{k=1}^{K} w_{k,q}^2)}_{\text{cross-correlation}}\Bigg\}\big(\mathbf{x}_{[p]}^T\mathbf{x}_{[q]}\big)^2 \quad (4)\end{aligned}}$$

3.2 Convexity and Optimization

We train all the models $f_0, f_1, \ldots, f_K$ *simultaneously* by minimizing the total overall loss $\mathcal{L}$. A conjoint optimization is performed because the cross-correlation penalty terms between $\mathbf{w}_0$ and $\mathbf{w}_k$'s induce dependence between the models.

For optimization we employ the accelerated subgradient descent algorithm which is guaranteed to find the global optimum solution because, as we show next, our loss function $\mathcal{L}$ is convex.

Theorem 1. *The joint loss function $\mathcal{L}$ involving the cross- and self-correlation penalty terms among $\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_K$ remains convex.*

Proof. The non-negative sum of convex functions is also convex. The first line of $\mathcal{L}$ as given above is known to be convex since $\ell(\cdot)$ (hinge loss) and L-p norms for $p \geq 1$ are both convex. The proof is then by showing that the overall correlation penalty term in the second line of $\mathcal{L}$ is also convex by showing that its Hessian matrix is positive semi-definite (PSD). See extended version [10] for details. □

Since our total loss is a convex function, we can use gradient-based optimization to solve it to optimality. To this end, we provide the gradient updates for both $\mathbf{w}_0$ and $\mathbf{w}_k$'s in closed form as follows.

Partial derivative of $\mathcal{L}$ w.r.t. $\mathbf{w}_0$:

$$\frac{\partial \mathcal{L}}{\partial w_{0,p}} = \sum_{i=1}^{n}\frac{\partial\big[1 - y_i(\mathbf{w}_0^T\mathbf{x}_i + b_0)\big]_+}{\partial w_{0,p}} + \lambda_0 w_{0,p} + \mu w_{0,p}\sum_{q=1}^{d}\big(\sum_{k=1}^{K} w_{k,q}^2 + w_{0,q}^2\big)\big(\mathbf{x}_{[p]}^T\mathbf{x}_{[q]}\big)^2$$

where

$$\frac{\partial\big[1 - y_i(\mathbf{w}_0^T\mathbf{x}_i + b_0)\big]_+}{\partial w_{0,p}} = \begin{cases} 0 & \text{if } y_i(\mathbf{w}_0^T\mathbf{x}_i + b_0) \geq 1 \\ -y_i x_{i,p} & \text{otherwise.}\end{cases} \quad (5)$$

The vector-update $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_0}$ can be given in matrix-vector form using the above gradients as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_0} = \big[\mathbf{X}^T(-\mathbf{y}\odot\mathbb{I}(1-\mathbf{y}\odot(\mathbf{X}\mathbf{w}_0+b_0) > 0)) + \mathbf{w}_0\odot\big(\lambda_0\mathbf{1}+\mu(\mathbf{X}^T\mathbf{X})^2(\sum_k \mathbf{w}_k^2+\mathbf{w}_0^2)\big)\big] \quad (6)$$

where $\mathbb{I}(\cdot)$ is the indicator function, $\mathbf{1}$ is a length-n all-ones vector, and $(\mathbf{A})^2 = \mathbf{A} \odot \mathbf{A}$, i.e., element-wise product, for both matrix $\mathbf{A}$ as well as for vector $\mathbf{a}$.

Partial derivative of $\mathcal{L}$ w.r.t. $\mathbf{w}_k$: The steps for each $\mathbf{w}_k$ is similar, we directly provide the vector-update below.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_k} = \left[\mathbf{R}^T(-\mathbf{y}_0 \odot \mathbb{I}(1-\mathbf{y}_0 \odot (\mathbf{R}\mathbf{w}_k + b_k) > 0)) + \mathbf{w}_k \odot \left(\lambda_k \mathbf{1} + \mu (\mathbf{X}^T\mathbf{X})^2 (\mathbf{w}_k^2 + \mathbf{w}_0^2)\right)\right] \tag{7}$$

where $\mathbf{R} \in \mathbb{R}^{n_0 \times d}$ and $\mathbf{y}_0 \in \mathbb{R}^{n_0}$ consist of only the rare-class instances.

3.3 Time and Space-Complexity Analysis

Time Complexity. The (first) gradient term in Eq (6) that is related to the hinge-loss is $\mathcal{O}(nd)$. The (second) term related to the correlation-based regularization requires $\mathbf{X}^T\mathbf{X}$ which can be computed in $\mathcal{O}(nd^2)$ apriori and *reused* over the gradient iterations. The term $(\sum_k \mathbf{w}_k^2 + \mathbf{w}_0^2)$ takes $\mathcal{O}(Kd)$, and its following multiplication with $(\mathbf{X}^T\mathbf{X})^2$ takes an additional $\mathcal{O}(d^2)$. The remaning operations (summation with $\lambda_0 \mathbf{1}$ and element-wise product with $\mathbf{w}_0$) are only $\mathcal{O}(d)$. As such, the overall computational complexity for subgradient descent for $\mathbf{w}_0$ is $\mathcal{O}(nd^2 + t[nd + Kd + d^2])$, where t is the number of gradient itearations.

The time complexity for updating the $\mathbf{w}_k$'s can be derived similarly as Eq. (7) consists of similar terms, which can be written as $\mathcal{O}(t[n_0 d + d^2])$. Note that we omit the $\mathcal{O}(nd^2)$ this time as $\mathbf{X}^T\mathbf{X}$ needs to be computed only once and can be shared across all update rules. Moreover, the number of iterations t is the same as before since the parameter estimation is conjoint.

Space Complexity. We require $\mathcal{O}(d^2)$ storage for keeping $(\mathbf{X}^T\mathbf{X})^2$, $\mathcal{O}(Kd)$ for all the parameter vectors $\mathbf{w}_0, \ldots, \mathbf{w}_K$, and $\mathcal{O}(nd)$ for storing $\mathbf{X}$, for an overall $\mathcal{O}(d^2 + Kd + nd)$ space complexity.

Remarks on massive and/or high-dimensional datasets: Note that both time and space complexity of our RaRecognize is quadratic in d and linear in n. We conclude with parting remarks on cases with large d and huge n.

First, high-dimensional data with large d: In this case, we propose two possible directions to make the problem tractable. Of course, the first one is dimensionality reduction or representation learning. When the data lies on a relatively low-d manifold, one could instead use compound features. We apply our RaRecognize to document datasets, where compound features are not only fewer but also sufficiently expressive of the data. The second direction is to get rid of feature correlations, for instance via factor analysis. This would drop the term $\left(\mathbf{x}_{[p]}^T \mathbf{x}_{[q]}\right)^2$ from $\mathcal{L}$ (See (4)) and lead to updates that are only *linear* in d.

Next, massive data with huge n: We presented our optimization using batch subgradient descent. When n is very, very large then storing the original data in memory may not be feasible. We remark that one could directly employ mini-batch or even stochastic gradient descent in such cases, dropping the space requirement to $O(d^2 + Kd)$.

4 Evaluation

We design experiments to evaluate our proposed method with respect to the following questions:

- **RQ1) Top-level classification (via GC f_0)**: How does RaRecognize perform in differentiating rare-class instances from the majority compared with the state-of-the-art?
- **RQ2) Sub-level classification (via SCs f_k's)**: How does RaRecognize perform in recognizing recurrent and emerging rare subclasses among the compared methods?
- **RQ3) Interpretability**: Can we interpret RaRecognize as a model as to what it has learned and what insights can we draw?
- **RQ4) Efficiency**: What is the scalability of RaRecognize? How does it compare to the baselines w.r.t. the running time-vs-performance trade-off?

4.1 Experiment Setup

Dataset Description. In this study, we use 3 different datasets with characteristics summarized in Table 1. The first two datasets are obtained from our industry collaborator (proprietary) and a third public one which we put together.

Table 1: Summary of datasets.

Name	$\|\mathcal{R}\|$	$\|\mathcal{N}\|$	K	**Avg.** $\|\mathcal{R}_k\|$
Risk-Doc	2948	2777	15	196.5
Risk-Sen	1551	7755	8	193.9
NYT-Dstr	2127	10560	13	163.6

Risk-Doc: This dataset contains online documents, e.g. news articles, social network posts, which are labeled risky or non-risky to the corporate entities mentioned. If a document is risky, it is further assigned to one of the 15 risk subclasses: {Climate change, Cyber attack, Data leak, Drug abuse, Engine failure, Fraud, Gun violation, Low stock, Military attack, Misleading statement, Money laundering, Negative growth, Sexual assault, Spying, Trade war}.

Risk-Sen: This contains labeled sentences attracted from news articles and categorized into 8 different subclasses: {Bankruptcy, Bribery corruption, Counterfeiting, Cyber privacy, Environment, Fraud false claims, Labor, Money laundering}. The majority class consists of non-risky sentences. Note that this dataset comes from a set of articles different from Risk-Doc.

NYT-Dstr: Extracted from the New York Times, this dataset is comprised of articles on the topics of disasters, i.e. both natural and human-instigated. These topics cover 13 disasters from {Drought, Earthquakes, Explosions, Floods, Forest and bush fire, Hazardous and toxic substance, Landslides, Lighting, Snowstorms, Tornado, Tropical storm, Volcanoes, Water pollution}. It also includes a class of random non-disaster news articles from New York Times.

Document representations. In this study we apply our work to document datasets, for which we need to define a feature representation. There are numerous options. We report results with tfidf with top 1K words based on frequency, as well as PCA- and ICA-projected data. Linear embedding techniques reduce dimensionality while preserving interpretability. We omit results using non-linear feature representations (e.g., doc2vec [6]) as they did not provide any significant performance gain despite computational overhead.

Train/Test Splits. For each dataset, we randomly partition 2/3 of rare subclasses as seen and 1/3 of them as unseen. For training, we use 80% of the seen subclass instances at random and the rest 20% forms a seen subclass test set, denoted by $\mathcal{R}_s$. The set of unseen subclass instances, denoted by $\mathcal{R}_u$, goes into the test as well. Thus, the rare subclass test consists of 2 parts, i.e. $\mathcal{R}_{\text{test}} = \mathcal{R}_s \cup \mathcal{R}_u$. In addition, we also reserve a random 80% of the majority class for training and the rest 20% for testing, denoted by $\mathcal{N}_{test}$. Further, to obtain stable results, we repeat our experiment on 5 different random train/test constructions and report averaged outcomes.

Performance Metrics. (1) For measuring top-level classification performance, we use 3 common metrics [16,8]: *Precision*, *Recall*, *F1* formally defined in our context as:

$$\textit{Precision} = \frac{|\mathcal{R}_{\text{test}} \cap \hat{\mathcal{R}}_{\text{test}}|}{|\hat{\mathcal{R}}_{\text{test}}|}, \textit{Recall} = \frac{|\mathcal{R}_{\text{test}} \cap \hat{\mathcal{R}}_{\text{test}}|}{|\mathcal{R}_{\text{test}}|}, \textit{F1} = \frac{2 * \textit{Precision} * \textit{Recall}}{\textit{Precision} + \textit{Recall}},$$

where $\hat{\mathcal{R}}_{\text{test}}$ is the set of examples predicted as rare subclass. To identify which part of the test (seen or unseen subclasses) the model makes mistakes, we also measure *Precision (seen)*, *Recall (seen)* and *Recall (unseen)* defined as follows:

$$\textit{Precision}(\textit{seen}) = \frac{|\mathcal{R}_s \cap \hat{\mathcal{R}}_{\text{test}}|}{|\hat{\mathcal{R}}_{\text{test}}|}, \textit{Recall}(\textit{seen}) = \frac{|\mathcal{R}_s \cap \hat{\mathcal{R}}_{\text{test}}|}{|\mathcal{R}_s|}, \textit{Recall}(\textit{unseen}) = \frac{|\mathcal{R}_u \cap \hat{\mathcal{R}}_{\text{test}}|}{|\mathcal{R}_u|}.$$

(2) For sub-level classification test, to quantify the fraction of seen subclass test instances correctly classified and unseen subclass test instances as emerging, we use the following metric:

$$acc(rare) = \frac{\overbrace{|\mathcal{R}_u \cap \hat{\mathcal{R}}_u|}^{acc(emerging)} + \sum_{k=1}^{K} \overbrace{|\mathcal{R}_{ks} \cap \hat{\mathcal{R}}_{ks}|}^{acc(recurrent)}}{|\mathcal{R}_{\text{test}}|},$$

where $\mathcal{R}_{ks}$ is the set of test examples in subclass k and $\hat{\mathcal{R}}_{ks}$ is the set of examples assigned to that subclass. Here *acc(rare)* = 1 if both seen subclass test instances are perfectly classified to their respective subclasses and unseen subclass instances as emerging. For all of the above metrics, the higher is better.

Compared Methods. We compare RaRecognize with 2 state-of-the-art methods and 2 simple baselines:

- RaRecognize-1K, RaRecognize-PCA, RaRecognize-ICA: 3 versions of RaRecognize when tfidf with 1K word dictionary, PCA and ICA representations are used. In RaRecognize-PCA, we drop the feature correlation terms $\left(\mathbf{x}_{[p]}^T \mathbf{x}_{[q]}\right)^2$ since features are orthogonal. For the sub-level classification, RaRecognize learns a rejection threshold for each specialized classifier based on extreme value theory [15].
- L2AC [16]: the most recent method (2019) in open-world classification setting that is based on deep neural networks. We use the recommended parameters $k = 5, n = 9$ (in their notation) from the paper.
- SENCForest [8]: another state-of-the-art ensemble method (2017) using random decision trees for classification under emerging classes. We run SENCForest with 100 trees and subsample size 100 as suggested in their paper.

– Baseline: a baseline of RaRecognize when both cross- and self-correlation terms in Eq. (4) are removed, via setting $\mu = 0$. Basically, Baseline is independently trained $f_0, \ldots, f_K$.
– Baseline-r: a variant of Baseline when classification threshold (0.5 by default) is chosen so that the *Recall* matches that of RaRecognize.

Note that SENCForest and L2AC aim to detect *any* emerging class without categorizing into rare or majority. For fair comparison, we only inlcude new *rare* subclasses in our test data and consider their rejected instances as belonging to those. In reality, however, emerging classes need to be categorized as rare or not, which these existing methods did not address.

Table 2: Performance of methods on the three datasets.

	Precision			**Recall**			**F1**		
Methods	Risk-Doc	Risk-Sen	NYT-Dstr	Risk-Doc	Risk-Sen	NYT-Dstr	Risk-Doc	Risk-Sen	NYT-Dstr
SENCForest	0.46±0.12	0.14±0.03	0.36±0.09	0.59±0.09	0.41±0.08	0.39±0.10	0.52±0.11	0.21±0.04	0.37±0.06
L2AC	0.79±0.08	0.47±0.06	0.31±0.07	0.57±0.29	0.85±0.05	0.76±0.07	0.63±0.24	0.60±0.04	0.44±0.07
Baseline	0.89±0.04	0.79±0.05	0.86±0.03	0.52±0.06	0.55±0.04	0.63±0.03	0.65±0.05	0.65±0.03	0.73±0.03
Baseline-r	0.76±0.10	0.56±0.07	0.71±0.07	0.58±0.13	0.57±0.14	0.79±0.04	0.65±0.10	0.55±0.06	0.75±0.04
RaRecognize-1K	0.89±0.02	0.83±0.09	0.84±0.03	0.58±0.13	0.57±0.14	0.79±0.04	0.70±0.09	0.66±0.08	**0.81±0.02**
RaRecognize-PCA	0.85±0.06	0.79±0.10	0.84±0.10	0.58±0.17	0.71±0.17	0.80±0.07	0.68±0.14	0.73±0.09	**0.81±0.01**
RaRecognize-ICA	0.74±0.07	0.72±0.11	0.84±0.12	0.73±0.24	0.85±0.18	0.82±0.08	**0.71±0.09**	**0.78±0.07**	**0.81±0.04**

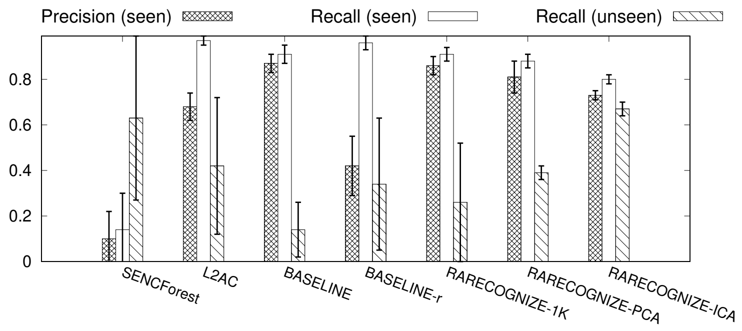


Fig. 4: *Precision (seen)*, *Recall (seen)* and *Recall (unseen)* of methods on Risk-Doc (RaRecognize-ICA achieves the best balance between *Precision* and *Recall* on both seen and unseen test instances).

4.2 Experiment Results

In the following, we sequentially answer the questions by analyzing our experimental results and comparing between methods.

RQ1) Top-level Classification into Rare vs. Majority Class. We report the *Precision*, *Recall* and *F1* of all methods on three datasets in Table 2. For SENCForest, L2AC, Baseline and Baseline-r, we report results for the representation (tfidf top-1K, PCA, ICA) that yielded the highest *F1* value.

From Table 2, we see that RaRecognize-1K, RaRecognize-PCA and RaRecognize-ICA outperform other methods in terms of *F1* score in all cases and *Precision*, *Recall* in most cases. Compared to Baseline and Baseline-r, *F1* score of RaRecognize-ICA is 6-13% higher than the best result among the two. This demonstrates that cross- and self-correlations are crucial in RaRecognize. Surprisingly, the gap to SENCForest and L2AC is even larger in terms of *F1*, between 8-37% higher. This shows that previous methods on detecting any new emerging classes do not work well when we only target rare subclasses.

Among the three versions of RaRecognize, RaRecognize-ICA gives the highest *F1*. RaRecognize-ICA achieves the best balance between precision and recall while RaRecognize-1K and RaRecognize-PCA seem to have very high *Precision* but much lower *Recall*. That means that RaRecognize-1K and RaRecognize-PCA are better than RaRecognize-ICA at discarding majority samples and worse at recognizing rare subclasses.

In Fig. 4, we have the *Precision (seen)*, *Recall (seen)* and *Recall (unseen)* measures of all the methods on Risk-Doc (Figures for other datasets are similar, see [10]). This figure shows that RaRecognize-ICA also achieves a good balance between seen and unseen subclass classification, i.e., it recognizes both these subclasses equally well. On the other hand, most of other methods achieve high *Precision (seen)* and *Recall (seen)* but much lower *Recall (unseen)*, except SENCForest which only has high *Recall (unseen)*. This is because SENCForest rejects most instances as unseen which however hurts *Precision* drastically.

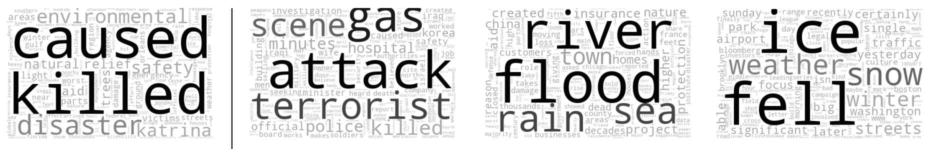
Table 3: *acc(rare)* of methods on the three datasets.

Methods	Risk-Doc	Risk-Sen	NYT-Dstr
SENCForest	0.37±0.09	0.41±0.08	0.34±0.04
L2AC	0.22±0.17	0.20±0.20	0.08±0.12
Baseline	0.41±0.07	0.37±0.04	0.41±0.04
Baseline-r	0.43±0.16	0.38±0.03	0.42±0.04
RaRecognize-1K	0.45±0.08	0.38±0.12	0.46±0.12
RaRecognize-PCA	0.50±0.14	0.59±0.14	0.65±0.15
RaRecognize-ICA	**0.63±0.14**	**0.62±0.09**	**0.64±0.15**

RQ2) Sub-level Classification into Recurrent and Emerging Rare Subclasses. We report the *acc(rare)* of all the methods in Table 3 (Breakdown of errors in confusion tables are given in the extended version of this article [10]).

Tables 2 and 3 reflect that all three versions of RaRecognize are always better or comparable to the others in terms of *acc(rare)* and RaRecognize-ICA achieves the highest value. RaRecognize-ICA achieves significantly higher *acc(rare)* than all the baselines. SENCForest seems to perform the next best due to the fact that it classifies most of the instances as emerging which results in high classification performance on unseen subclasses.

RQ3) Model Interpretation. In Fig. 5, we plot the wordclouds representing the general and 3 specialized classifiers for 3 disaster subclasses (sizes of the words proportional to their weights learned by RaRecognize-1K). Exist-



(a) General disaster (b) Explosions (c) Floods (d) Snowstorms

Fig. 5: Word clouds for the weights of general and 3 specialized classifiers in RaRecognize-1K on NYT-Dstr (See [10] for other subclasses).

ing methods, SENCForest and L2AC, are not interpretable due to respective ensemble and deep neural network-based models they employ.

From Fig. 5, specialized disaster classifiers are clearly characterized by specific words closely related to the respective disasters, whereas the general classifier is heavily weighted by common words to every disaster. Specifically, Explosions classifier picks up `attack`, `gas`, `terrorist`, `scene` as most weighted keywords, and Snowstorms classifier puts heavy weights on words `ice`, `fell`, `snow`, `weather`. The general classifier is highlighted by the words `caused`, `killed`, `disaster` which describe consequences of most disasters. Wordclouds on other disaster subclasses, along with those for other datasets can be found in [10].

Thanks to the interpretability that RaRecognize offers, we can look deeper into the significance of individual words in classifying documents. Besides its promising quantitative performance, these qualitative results confirm that our method has learned what agrees with human intuition.

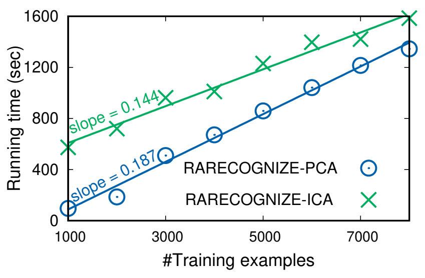


Fig. 6: RaRecognize scales linearly in number of training examples.

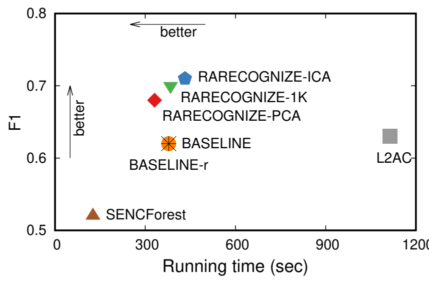


Fig. 7: Time-Performance trade-off of all compared methods.

RQ4) Scalability and Time-Performance Trade-off. Besides our formal complexity analysis, we demonstrate the scalability of RaRecognize empirically. Fig. 6 shows the running time of RaRecognize-PCA and RaRecognize-ICA when varying the amount of training data. The running time increases linearly with the data size. RaRecognize with PCA is faster than that with ICA thanks to no feature correlations (i.e. $\left(\mathbf{x}_{[p]}^T \mathbf{x}_{[q]}\right)^2$ dropped).

In Fig. 7, we show the time-performance trade-off among all compared methods. We conclude that RaRecognize with three representations run relatively fast, only slower than SENCForest, and returns the highest performance in terms of *F1*. L2AC consumes a huge amount of time for training a neural network, with subpar performance.

5 Related Work

Our work is closely related with two fields, namely open-world classification and continual learning. Both belong to the category of lifelong machine learning [1].

Open-world classification. Traditional close-world classification assumes that all test classes are known and seen in training data [4,17]. However, such assumption could be violated in reality. Open-world classification, in contrast, assumes unseen and novel classes could emerge during test time, and addresses the classification problem by recognizing unseen classes. Previous works [13,14,16,9,8] propose different approaches under this setting.

Specifically, DOC [13] leverages convolutional neural nets (CNNs) with multiple sigmoid functions to classify examples as seen or emerging. [14] follows the same DOC module and performs hierarchical clustering to all rejected samples. Later, L2AC [16] proposes to use a meta-classifier and a ranker to add or delete a class without re-training. However, it requires a large amount of computation in both training and testing due to the top-k search over all training data.

SENCForest [8] is a randomized ensemble method. It grows multiple random forests and rejects examples when all random forests yield "new class". Under the same setting, SENC-MaS [9] maintains matrix sketchings to decide whether an example belongs to a seen class or emerging.

In the emerging rare subclass setting, the previous approaches aim at recognizing *any and every* classes and are not able to ignore the *not-of-interest* classes while recognizing emerging ones, thus consume much more memory and time.

Continual learning. There are recent works investigating continual learning or incremental learning [12,11]. They aim at solving the issue of catastrophic forgetting [2] in connectionist networks. In this field, models are proposed to continually learn new classes without losing performance on old seen classes.

Previous works [11,5,7,3] show promising results. However, the number of documents in rare subclasses *of-interest* in our setting is usually not large enough for neural networks to be sufficiently trained. Consequently, the neural network approach does not perform well in rare-class classification and recognition.

6 Conclusion

We proposed RaRecognize for rare-class recognition over a continuous stream, in which new subclasses may emerge. RaRecognize employs a general classifier to filter out not-rare class instances (top-level) and a set of specialized classifiers that recognize known rare subclasses or otherwise reject as emerging (sub-level). Since majority of incoming instances are filtered out and new rare subclasses are a few, RaRecognize processes incoming data fast and grows in size slowly.

Extensive experiments show that it outperforms two most recent state of the art as well as two simple baselines significantly in both top- and sub-level tasks, while achieving the best efficiency-performance balance and offering interpretability. Future work will extend RaRecognize to an end-to-end system that clusters emerging instances and trains all the relevant models incrementally.

Acknowledgments

This research is sponsored by NSF CAREER 1452425 and IIS1408287. In addition, the researchers gratefully acknowledge the support of the Risk and Regulatory Services Innovation Center at Carnegie Mellon University sponsored by PwC. Conclusions in this material are those of the authors and do not necessarily reflect the views, expressed or implied, of the funding parties.

References

1. Z. Chen and B. Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(3):1–145, 2016.
2. R. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 05 1999.
3. R. Kemker and C. Kanan. Fearnet: Brain-inspired model for incremental learning. In *ICLR*, 2018.
4. Y. Kim. Convolutional neural networks for sentence classification. *EMNLP*, 2014.
5. J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017.
6. Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.
7. S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *NeurlPS*, pages 4652–4662, 2017.
8. X. Mu, K. M. Ting, and Z.-H. Zhou. Classification under streaming emerging new classes: A solution using completely-random trees. *IEEE TKDE*, 29(8), 2017.
9. X. Mu, F. Zhu, J. Du, E.-P. Lim, and Z.-H. Zhou. Streaming classification with emerging new class by class matrix sketching. In *AAAI*, 2017.
10. H. Nguyen, X. Wang, and L. Akoglu. Continual rare-class recognition with emerging novel subclasses. *arXiv preprint*, 2019.
11. S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017.
12. H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *NeurlPS*, pages 2990–2999, 2017.
13. L. Shu, H. Xu, and B. Liu. Doc: Deep open classification of text documents. *EMNLP*, 2017.
14. L. Shu, H. Xu, and B. Liu. Unseen class discovery in open-world classification. *arXiv preprint arXiv:1801.05609*, 2018.
15. A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet. Anomaly detection in streams with extreme value theory. In *KDD*, pages 1067–1075. ACM, 2017.
16. H. Xu, B. Liu, L. Shu, and P. Yu. Open-world learning and application to product classification. In *WWW*, 2019.
17. X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *NeurlPS*, pages 649–657, 2015.