A Novel Neural Ensemble Architecture for On-the-fly Classification of Evolving Text Streams

POUYA GHAHRAMANIAN and SEPEHR BAKHSHI, Bilkent University, Turkey
HAMED BONAB, Amazon Inc., USA
FAZLI CAN, Bilkent University, Turkey

We study on-the-fly classification of evolving text streams in which the relation between the input data and target labels changes over time—i.e., "concept drift." These variations decrease the model's performance, as predictions become less accurate over time and they necessitate a more adaptable system. While most studies focus on concept drift detection and handling with ensemble approaches, the application of neural models in this area is relatively less studied. We introduce Adaptive Neural Ensemble Network (*AdaNEN*), a novel ensemble-based neural approach, capable of handling concept drift in data streams. With our novel architecture, we address some of the problems neural models face when exploited for online adaptive learning environments. Most current studies address concept drift detection and handling in numerical streams, and the evolving text stream classification remains relatively unexplored. We hypothesize that the lack of public and large-scale experimental data could be one reason. To this end, we propose a method based on an existing approach for generating evolving text streams by introducing various types of concept drifts to real-world text datasets. We provide an extensive evaluation of our proposed approach using 12 state-of-the-art baselines and 13 datasets. We first evaluate concept drift handling capability of AdaNEN and the baseline models on evolving numerical streams; this aims to demonstrate the concept drift handling capabilities of our method on a general spectrum and motivate its use in evolving text streams. The models are then evaluated in evolving text stream classification. Our experimental results show that AdaNEN consistently outperforms the existing approaches in terms of predictive performance with conservative efficiency.

CCS Concepts: • **Information systems** → **Data stream mining**; • **Computing methodologies** → **Ensemble methods**; **Neural networks;**

Additional Key Words and Phrases: Data stream mining, concept drift, text stream classification, ensemble methods, neural networks

ACM Reference Format:
Pouya Ghahramanian, Sepehr Bakhshi, Hamed Bonab, and Fazli Can. 2024. A Novel Neural Ensemble Architecture for On-the-fly Classification of Evolving Text Streams. *ACM Trans. Knowl. Discov. Data.* 18, 4, Article 101 (February 2024), 24 pages.
https://doi.org/10.1145/3639054

H. Bonab Work done while at the University of Massachusetts Amherst.

This study is partially supported by TüBİTAK grant no. 120E103.

Authors' addresses: P. Ghahramanian, S. Bakhshi, and F. Can (Corresponding author), Bilkent Information Retrieval Group, Computer Engineering Department, Bilkent University, Ankara, Turkey, 06800; e-mails: {ghahramanian, sepehr.bakhshi}@bilkent.edu.tr, canf@cs.bilkent.edu.tr; H. Bonab, Amazon.com, Inc., Seattle, WA 98109, USA; e-mail: hamedrab@amazon.com.

ACM 1556-4681/2024/02-ART101
https://doi.org/10.1145/3639054

1 INTRODUCTION

Text classification is one of the well-known tasks in the literature with many real-world applications. Recent progress with pretrained neural networks, like BERT [15], led existing approaches towards more accurate predictions for a variety of text-processing tasks [26, 34, 38, 68]. However, such approaches are ill-suited for online learning settings where data flows over time, with possible changes of the relationship between the input data and its target variable.

In dynamically changing non-stationary environments, such as text streams, the conditional distribution of the target label given the input features can change over time. This phenomenon is called *concept drift* [45]. One of the biggest challenges is to develop algorithms that adapt themselves with each drift. Existing works focus mainly on numerical data such as traffic management, click streams, and the stock market [6, 9, 10, 23]. Gama et al. [19] characterize the adaptive online learning and provide a survey on different types of concept drift: —Sudden/Abrupt, —Incremental, —Gradual, and —Reoccuring. Ensembles of classifiers are commonly used for such settings [7, 20, 49, 61] with effective base models such as the **Hoeffding Tree (HT)** [27].

We refer to text stream with concept drift as *evolving text stream*. We study the on-the-fly classification of evolving text streams. For example, detecting spam emails from incoming emails, in which definitions of spam evolve over time, demonstrates certain challenges with our problem setting [30]. Due to the volume and velocity of incoming data, it is impractical to train the model with multiple passes over the data—as is the practice with offline settings. In addition, label prediction becomes time-sensitive, requiring quick inferences [19].

For studying evolving text streams, we consider a user's (or a community of users') evolving interests over a stream of news documents. Concept drift can be defined as the change of the user's interest in news topics. For example, consider a news portal publishing on three categories: *politics*, *sports*, and *health*. In the beginning, a user might be interested in *politics*. After a while, in the case of a viral pandemic, the user's interest might change toward health-related news. This change can be in any of the four drift types aforementioned. Only a limited number of studies address this problem in the literature [1, 41].

In this article, we propose an ensemble-based neural network approach capable of handling concept drift in text streams, named *AdaNEN* (Section 3). The usage of neural models for online environments is overlooked, despite their impressive performance in many fields. With AdaNEN, we address some of the problems neural models face when exploited for online adaptive learning environments. Due to the lack of public and large-scale experimental data, we extend an existing approach to introduce different types of concept drifts to real-world text datasets and construct four evolving text streams based on the **20 News Groups (20NG)** [40] and the AG News [69] datasets (Section 4). We evaluate AdaNEN and the baseline models' concept drift handling strategies on evolving numerical streams. We then extend our experiments to evolving text streams. Our experimental results show that our proposed method outperforms the existing approaches in terms of predictive performance with a conservative efficiency (Section 5). We provide further analyses on understanding the behavior of the proposed model as data flows and present an ablation study showing the importance of the main components of AdaNEN (Section 6). In the same section, we also present an analysis of hyperparameters, evaluate the effectiveness and efficiency of the models, and conduct a statistical study. We conclude our article in Section 7 with a discussion of some future pointers. We give a brief overview of text, evolving stream, and evolving text stream classification in Section 2. Our contributions are summarized as follows: We

— Introduce an ensemble-based neural approach for evolving text stream classification,
— Offer a new method to create large-scale evolving text streams and use it to construct four drifting text streams,

— Present an extensive evaluation of our proposed approach using 12 state-of-the-art baselines and 13 datasets.
— Provide public access to our implementation of AdaNEN and newly constructed datasets, enabling reproducible experiments and facilitating their application in future studies.[1]

2 RELATED WORK

Here, we first briefly describe general text classification approaches. Then, we give a summary of evolving data stream classification methods and survey the existing text stream classifiers.

2.1 Text Classification

Text classification is defined as automatically assigning predefined labels to text segments based on a similarity measure obtained by a machine learning algorithm. Traditionally, the focus has always been on using statistical and probabilistic approaches for text classification. As an example of probabilistic classifiers, Naive Bayes uses joint probabilities of words and categories for estimating the probability of a category given a document [56]. Clustering methods are also used for supervised text categorization. Aggarwal et al. [2] proposed a method for improving text classification performance by using clustering techniques especially when target categories are closely related.

In the past decade, word embedding methods have experienced a drastic change, and novel embedding methods have been introduced for representing words as numerical vectors. Two prominent instances of the recent word embedding methods are Word2Vec [47] and GloVe [54], which map words into numerical vectors that capture their semantic meanings. These advancements in word embedding paved the way for leveraging neural models in text classification, further revolutionizing the field.

Various neural and deep learning approaches have been proposed for classifying text documents. Zhang and LeCun [68] exploit a convolutional neural architecture for text classification by using a character-level quantization of text documents as input data. The authors reported significant gains for the examined text classification tasks. Recurrent neural networks are the other type of deep neural networks used in text classification. Nowak et al. [52] investigated the performance of LSTM and two of its variants, namely, Bidirectional LSTM and GRU in short text classification tasks. More recently, the introduction of pre-trained deep neural language models, like BERT [15], shifted almost every downstream task toward such models, and huge gains have been reported for many NLP and IR tasks. Another recent attempt in text classification is using Capsule Network [26, 57]. Kim et al. [34] proposed a capsule-based model for text classification. As they indicate in their work, the capsule network has enough potential to be used as a powerful model for text classification, and it has some advantages over convolutional neural networks. However, as we show in our experiments, neural and deep learning models are not yet deployable for evolving text stream environments.

2.2 Evolving Stream Classification

Concept drift occurs in a data stream due to changes in the underlying data distribution or target concept [8, 19, 33]. General frameworks of learning with drift detection and adaptation capabilities are provided by Lu et al. [45] and Aggarwal et al. [3]. ADWIN [4] is a well-known concept drift detection method that uses error rate statistics to adjust instance window size. Another revolutionary model used in data stream classification is the Hoeffding Tree [27]. A Hoeffding Tree is capable of handling large data streams.

[1]The source code of AdaNEN and newly constructed datasets are available at https://github.com/pouyaghahramanian/AdaNEN

A more commonly used method for handling concept drift is an ensemble of classifiers. Kolter and Maloof [37] introduce one of the most famous ensemble methods designed especially for concept drift handling. **Dynamic weighted majority (DWM)** adds and drops classifiers to an ensemble based on their performance over time. Hidalgo et al. [25] propose PEP, an approach for identifying optimal data stream-related parameters of the ensemble methods. Bonab and Can [9] introduce a **geometrically optimum and online-weighted ensemble method (GOOWE)** for evolving data stream classification by assigning optimum weights to the base classifiers. Unsupervised methods for active concept drift detection were recently proposed by Gözüaçık et al. [20, 21]. Gulcan and Can [22] introduced an unsupervised concept drift detector for multi-label settings. However, such methods are not thoroughly studied for the text stream classification problem.

2.3 Evolving Text Stream Classification

Due to ubiquitous and popular social media and weblog platforms as well as many other data sources, a huge amount of text data is generated on the Web. Online text classification can be applied to a number of different downstream tasks, such as news filtering, and event detection and tracking [1, 11, 41, 60]. Aggarwal [1] provides a comprehensive survey of the existing approaches for text streams covering different learning paradigms such as classification and clustering. As discussed, a considerable amount of literature has been published on text classification [48, 62] with exciting improvements, especially with the pre-trained language model-based classifiers. However, a few of these studies have investigated evolving text stream environments, where the text data come in the form of a stream [1, 28].

Lindstrom et al. [43] propose an approach for classifying documents' stream in which the labeling cost is high by training the model on the most informative data instances. Nishida et al. [51] study the problem of evolving short text stream classification, where the underlying data distribution changes over time, and introduce P-Switch, a model suitable for this setting. Li et al. [42] study the application of deep learning models in short text stream classification and introduce a distributed LSTM ensemble model for this problem. Yoshinaga and Kitsuregawa [67] study the real-time efficient processing of disaster-related Twitter[2] streams using a self-adaptive classifier in which the classifier reuses parts of the past streams that are related to the ongoing stream. Feature selection methods are used to reduce the dimensionality of text data to enable classification in real-time. de Moraes and Gradvohl [13] compare six feature selection methods for binary text stream classification under concept drift and introduce the OFSER algorithm to limit the effect of feature drift in evolving text stream classification. Nanculef et al. [50] study the problem of multi-label text stream classification and introduce a model for online text representation and classification.

For other learning paradigms on text streams, Katakis et al. [30] introduce **Conceptual Clustering and Prediction (CCP)**, using clustering to identify concepts and train cluster-specific classifiers in an ensemble. Kumar et al. [39] proposed OSDM, an online model for short text stream clustering by embedding word-occurrence semantic information in the clustering task. Cortes et al. [12] investigated the adaptive learning of neural network structure and its weights. Zhou et al. [70] introduced an online algorithm based on autoencoders for determining the optimal model complexity by adding and merging features.

3 AdaNEN: ADAPTIVE NEURAL ENSEMBLE NETWORK

Problem Setup. Figure 1 presents the general schema, and Table 1 defines the notation of symbols. The text data stream is generated over time via a source such as social media or a news portal. We

[2] https://twitter.com/

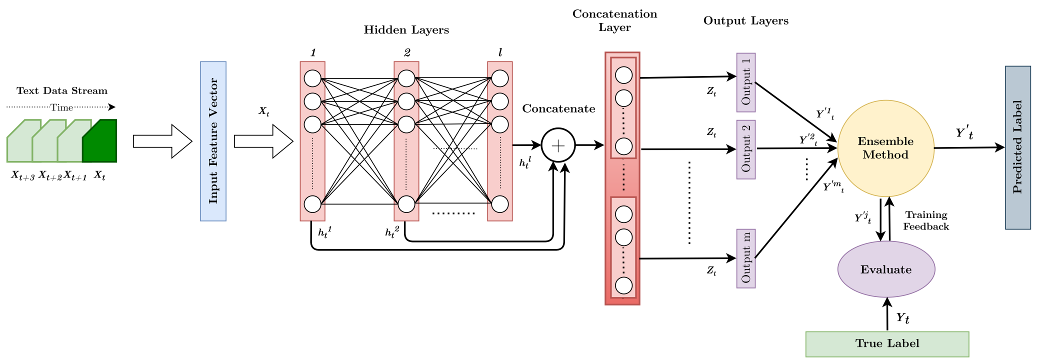


Fig. 1. Overall schema of our Adaptive Neural Ensemble Network (AdaNEN) for evolving text stream classification. AdaNEN uses an ensemble of output layers with different learning rates and a concatenation layer.

Table 1. Notation and Definition of the Symbols in AdaNEN's Algorithm

Notation	Definition
X_t	Input data instance at time t
Y_t	True class label for X_t
Y_t^k	True class indicator for X_t and target class k: 1 if k is the true class of X_t; 0 otherwise
$Y_t'^j$	Output of the jth output layer for X_t
Y_t'	Result of output layers' weighted average ensemble for X_t: $Y_t' = \sum_{j=1}^{m} W_j \times Y_t'^j$
P_t^{jk}	Probability prediction of jth output layer for X_t belonging to class k
h_t^i	Output of ith hidden layer for X_t
Z_t	Concatenation of the hidden layers as: $Z_t = h_t^1 \oplus h_t^2 \oplus \cdots \oplus h_t^l$
A^i	Weights matrix between ith and $(i+1)^{th}$ hidden layers
B_j^i	Weights matrix between ith hidden layer and jth output layer
$loss_j$	Cross entropy loss of the jth output layer
W_j	Ensemble weight of the jth output layer
C	Number of target classes
lr_{avg}	Weighted average learning rate of the network used for optimizing hidden layers
β	Penalization parameter used to update ensemble weights
l	Number of hidden layers
s	Regularization parameter for the ensemble method that prevents assigning near zero weights
m	Number of output layers

refer to a text data instance at time t as X_t—assuming a single instance arrives at a time. To feed the incoming text data instance to our model, a numerical dense feature representation is constructed for each instance. Similar to offline text classification, various word embedding methods can be used for this purpose. Any given classifier model in our problem setting works as follows: For an incoming data X_t, the model predicts the class label, i.e., Y_t', by generating probability values of belonging data instance to each C class labels, i.e., $P_t =< P_t^1, P_t^2, \ldots, P_t^C >$. Here, we only deal with the binary scenario,[3] i.e., $C = 2$. We assume that, for every instance, the correct label becomes available at time $t+1$, making the immediate evaluation procedure possible. Finally, using the true class of X_t, i.e., Y_t, and the obtained features, the model is updated incrementally. This is known as *prequential learning* or test-then-train learning paradigm in the literature [19, 45]. As a result, every data instance is used both for testing and training.

[3]Note that one might think of this problem as a multi-class problem with different interest levels. Here, for simplicity, we only consider the binary scenario and leave the other variations as the future work.

Motivation. Deep neural networks have shown impressive performance in text classification. However, there are a couple of reasons that make them ineffective for online learning. *First*, neural networks use a preset number of hidden layers, depending on the complexity of data. Using a small number of hidden layers limits the capacity of the network for learning, while excessive layers might cause over-fitting. Hence, choosing the proper number of hidden layers before training is not possible in data stream environments, as the complexity and characteristics of the data may change dynamically over time, making it challenging to determine an optimal structure in advance. *Second*, the learning rate of a neural network is pre-defined before training. Using a higher learning rate prevents the network from finding the global minima, while using a lower learning rate may cause optimization progress to fail. Optimization methods, such as Adam [35] and RMSPbrop [64], decrease the learning rate in the optimization process to find the global minima. However, in evolving data stream classification, we may need to increase the learning rate to adapt to variations in data faster and handle the concept drift.

Algorithm. We leverage the success of ensemble methods and address the issues with neural models for the design of our architecture. AdaNEN is a modification of a feed-forward neural network that uses multiple output layers connected to a concatenation layer. AdaNEN uses more than one output layer updating with different learning rates. Then, an ensemble method is used to combine predictions of the output layers and generate the final prediction. By using multiple output layers with different learning rates, we let the network focus on the most recent data by increasing the ensemble weight of the output layers with higher learning rates. Moreover, using a concatenation layer (instead of the last hidden layer) as input to the output layers enables the network to use an adjusted number of hidden layers to fit the complexity of current data.

AdaNEN's procedure is described in Algorithm 1. Let l be the number of hidden layers, m be the number of output layers, and lr_j be the learning rate of jth output layer. First, the network is fed with the current data instance. Then, hidden layers are concatenated ($\oplus$) and used as input to the output layers (lines 4–7). Assume that X_t is the current data instance and h_t^i is the ith hidden layer result for that data instance. We construct the concatenation layer, Z_t, using every hidden layer in the feed-forward network as shown in Equation (1).

$$Z_t = h_t^1 \oplus h_t^2 \oplus \cdots \oplus h_t^l \tag{1}$$

Let $Y_t^{\prime j}$ be the output of jth output layer. Class probability predictions for each output layer (P_t^{jk}) are obtained by using a softmax function, and the final output of the model ($Y_t^{\prime}$) is calculated by using a weighted ensemble (lines 8–14). Then, the softmax function is applied to $Y_t^{\prime}$ to obtain the model's prediction for X_t (line 15). Equation (2) shows how $Y_t^{\prime}$ is obtained.

$$Y_t^{\prime} = \sum_{j=1}^{m} W_j \times Y_t^{\prime j} \tag{2}$$

Figure 2 presents an example for the testing phase of AdaNEN with three hidden and three output layers for an incoming text data instance. The dimensionality of each layer is given above that layer in the parentheses. First, embedding of the text document (X_t) is obtained and fed to the hidden layers. Then, output layers are fed with the concatenation of hidden layers' results. The final output is calculated from Equation (2) by using the results of output layers and the current ensemble weights. Finally, the softmax function is applied to obtain the probability values of classes. As class 1 gets a higher probability value (0: 0.4978, 1: 0.5022), the model assigns this label to the incoming data instance.

The training phase of AdaNEN is described in lines 16–35 and presented in Figure 3. First, the loss value for each output layer is calculated using the cross-entropy loss function (lines 17–19).

ALGORITHM 1: AdaNEN: Adaptive Neural Ensemble Network

```
Input: $X$: Input Data Chunk
Output: $Y'$: Class Predictions
while data stream has more instance do
    $X_t$ = Data instance at time step $t$
    Feed the network with $X_t$ and get hidden layers' output ($h_t^i$)
    $Z_t = h_t^1 \oplus h_t^2 \oplus ... \oplus h_t^l$
    Feed the output layers with $Z_t$
    for $j$=1 ; $j \leq m$ do
        $Y_t'^j$ = output of $j^{th}$ output layer
        for $k$=1 ; $k \leq C$ do
            $P_t^{jk}$ = Softmax($Y_t'^j$)[$k$]
        end
    end
    $Y_t' = \sum_{j=1}^m W_j \times Y_t'^j$
    $Prediction_t$ = Max(Softmax($Y_t'$))
    $Y_t$ = True label for data instance at time step $t$
    for $j$=1 ; $j \leq m$ do
        $loss_j$ = CrossEntropyLoss($Y_t'^j$, $Y_t$) = -$\sum_{k=1}^C Y_t^k \times log(P_t^{jk})$
    end
    $loss_{total} = \sum_{j=1}^m W_j \times loss_j$
    for $j$=1 ; $j \leq m$ do
        for $i$=1 ; $i \leq l$ do
            $B_j^i = B_j^i - lr_j \times \frac{\partial loss_j}{\partial B_j^i}$
        end
    end
    $lr_{avg} = \sum_{j=1}^m W_j \times lr_j$
    for $i$=1 ; $i \leq l$ do
        for $j$=1 ; $j \leq m$ do
            for $k$=$i$ ; $i \leq l$ do
                $A^i = A^i - lr_{avg}\left[\sum_{j=1}^m W_j \times \left[\sum_{k=i}^l \frac{\partial loss_j}{\partial B_j^k} \times \frac{\partial B_j^k}{\partial A^k} \times \frac{\partial A^k}{\partial A^{k-1}} \cdots \frac{\partial A^{i+1}}{\partial A^i}\right]\right]$
            end
        end
    end
    $W_j = \frac{\max(W_j \times \beta^{loss_j}, \frac{s}{m})}{\sum_{i=1}^m W_i}$
end
```

Let P_t^{jk} be the probability prediction of jth output layer for X_t belonging to kth target class. Y_t^k is the correct class indicator for X_t data instance, which is 1 if k is the correct class of X_t, 0 otherwise. Equation (3) shows how the loss value of each output layer for the incoming data instance X_t is obtained.

$$loss_j = -\sum_{k=1}^{C} Y_t^k \times log(P_t^{jk}) \tag{3}$$

Total loss of the network for the input data instance X_t is calculated as the weighted average of loss values as shown in Equation (4) (line 20).

$$loss_{total} = \sum_{j=1}^{m} W_j \times loss_j \tag{4}$$

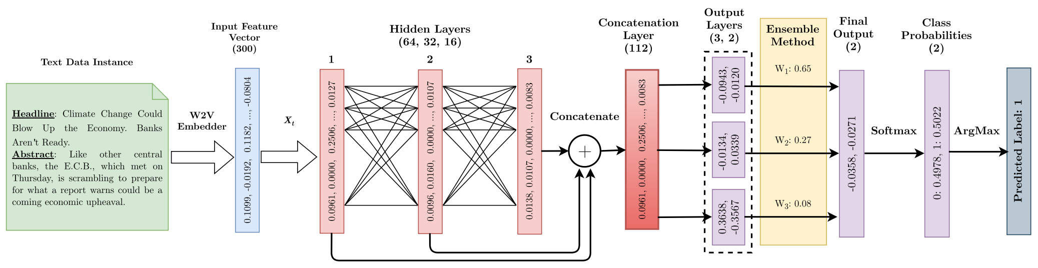


Fig. 2. Testing phase of AdaNEN with three hidden and three output layers for a text data instance. Layer dimensions are specified in parentheses above each layer. The text document is embedded using Word2Vec and processed through hidden layers. The output layers, using a concatenation of hidden layers and the ensemble weights, yield a final output (Equation (2)). The softmax function determines class probabilities, assigning the highest (in this case 1) as the data instance's label.

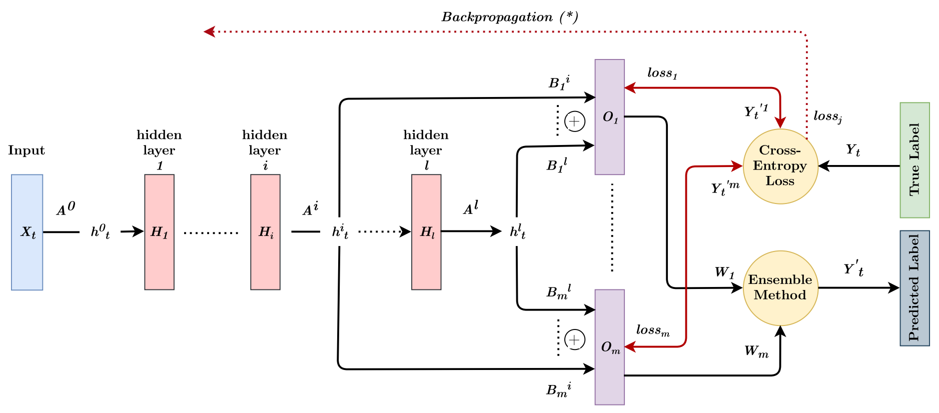


Fig. 3. Training phase of AdaNEN for updating network weights. For a training data instance, the cross-entropy loss for each output layer ($loss_j$) is calculated and backpropagated (*) to update output layers and hidden layers' weights (A^i, B^i_j).

There are three sets of parameters in AdaNEN's architecture: (i) Connection weights between two consecutive hidden layers H_{i-1} and H_i (A^i); (ii) Connection weights between each hidden layer and output layer pair H_i, O_j (B^i_j); (iii) Ensemble weights of the output layers (W_j). Correspondingly, AdaNEN's parameters are updated in three steps. First, for each pair of hidden layer and output layer (H_i, O_j) $loss_j$ is backpropagated to update B^i_j using the associated learning rate (lr_j) as shown in Equation (5) (lines 21–25).

$$B^i_j = B^i_j - lr_j \times \frac{\partial loss_j}{\partial B^i_j} \tag{5}$$

Then, the output layers' weighted average learning rate and loss values are used to optimize the weights of hidden layers through backpropagation (lines 26–33). The weighted average learning rate of the output layers is calculated as:

$$lr_{avg} = \sum_{j=1}^{m} W_j \times lr_j. \tag{6}$$

Loss values of the output layers contribute to backpropagation in proportion to their ensemble weights (W_j). Equation (7) shows how A^i weights are updated.

$$A^i = A^i - lr_{avg}\left[\sum_{j=1}^{m} W_j \times \left[\sum_{k=i}^{l} \frac{\partial loss_j}{\partial B_j^k} \times \frac{\partial B_j^k}{\partial A^k} \times \frac{\partial A^k}{\partial A^{k-1}} \cdots \frac{\partial A^{i+1}}{\partial A^i}\right]\right] \tag{7}$$

After optimizing hidden layers and output layers' weights, ensemble weights are updated using the hedge algorithm [18] based on the loss values as shown in Equation (8) (line 34). Ensemble weight for the jth output layer is denoted by W_j.

$$W_j = \frac{\max(W_j \times \beta^{loss_j}, \frac{s}{m})}{\sum_{i=1}^{m} W_i} \tag{8}$$

Note that β is the penalization parameter ($0 < \beta < 1$) and s is the regularization parameter that prevents assigning near-zero weights for the output layers. Also, ensemble weights are divided by the sum of weights to ensure that they sum up to one.

4 EVOLVING TEXT STREAM GENERATION

Evolving text stream classification domain suffers from the lack of large-scale benchmark datasets. This hinders the possibility of developing and evaluating classification models. In this section, we describe our approach for generating evolving text streams using available text classification datasets. We construct 20NG-Abrupt, 20NG-Gradual, AGNews-Abrupt, and AGNews-Gradual text streams based on **20 News Groups (20NG)** [40] and AG News datasets [69]. Our approach is an extended version of the method presented by Katakis et al. [31]. We introduce abrupt and gradual drifts to text datasets by using two transformation functions that map data instances into a new conceptual space (user interest) over time. Although we focus on news streams, our method may be applied to other domains as well. Classifying interesting movies from a streaming website or items from an online shop based on reviews are a few of other possibilities.

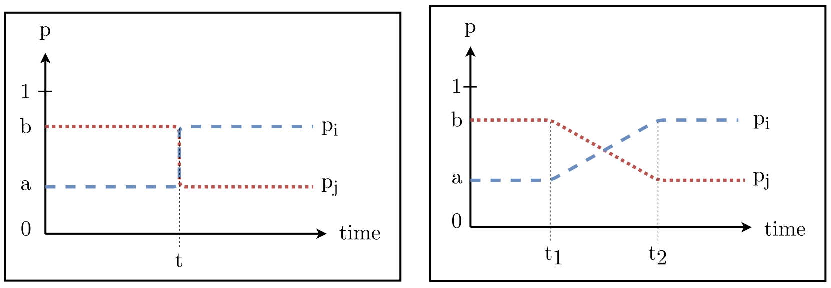
20NG is a collection of news articles with 20 news categories. We grouped the categories into four class labels: documents related to (1) **Cmp**, computers with *comp.** categories; (2) **Rec**, recreation with *rec.** categories; (3) **Sci**, science with *sci.** categories; and (4) **Tlk**, talk with *talk.** categories. The total number of data instances with the selected categories is 15,102. AG News topic classification dataset is constructed from the original AG news dataset by choosing four largest classes (world, sports, business, sci/tech). The total number of news articles in the dataset is 127,600.

We aim to classify news articles as interesting (positive) or not-interesting (negative) for a user.[4] At any given time, t, only a subset of these news groups are the interest of a user and the rest are not interesting. With n pre-defined class labels, i.e., $Y = \{y_1, y_2, \ldots, y_n\}$, the aim of a stream classifier is to assign labels to the incoming data instance—x. This prediction is done based on a set of joint probabilities $p(x, y_i)$. Khamassi et al. [32] define the set of joint probabilities at time t as a concept. Accordingly, concept drift is defined as a change in the joint probabilities between two timesteps. This means $p_{t_1}(X, Y) \neq p_{t_2}(X, Y)$. Note that X refers to a set of data instances. Using Bayes' theorem, we can state the joint probability as:

$$p(X, y_i) = p(y_i) \times p(X|y_i) = p(X) \times p(y_i|X). \tag{9}$$

Based on this formulation, three possible sources of concept drift can be defined: (i) Change in the prior probability of classes $p(y_i)$, which shows variation in target class distribution; (ii) Change in the posterior probability of target classes $p(y_i|X)$, which is known as *real* concept drift, meaning that decision boundaries change over time; and (iii) Change in the class conditional probability

[4]Note that different levels of interest can be defined for this problem. We only focus on binary scenario ($n = 2$) and leave the other possibilities as the future work.



(a) Abrupt (b) Gradual

Fig. 4. Adding abrupt and gradual drift to a stream. Abrupt drift occurs at time t, while gradual drift spans from t_1 to t_2. a and b are the probability values at the start and end of the drift.

Table 2. Concept Drift Distribution and Drift Intensity for Both 20NG-Abrupt and 20NG-Gradual Text Streams. Columns Represent the Most Recent Data Instance within that Span

	3.5k	4k	4.5k	5k	5.5k	6k	6.5k	7k	7.5k	8k	8.5k	9k	9.5k	10k	10.5k	11k	11.5k	15.1k
Cmp	+	+	-	-	-	-	-	+	+	+	-	-	-	+	+	+	+	+
Rec	-	-	+	-	-	-	+	+	-	-	+	+	+	-	+	+	-	-
Sci	+	-	-	+	-	+	+	-	+	-	+	-	+	+	-	+	+	+
Tlk	-	-	-	-	+	+	-	-	-	+	-	+	+	+	+	-	-	-
1/DI	-	50	100	25	200	50	25	150	100	75	250	125	75	50	100	25	50	-

$p(X|y_i)$, which is known as *virtual* concept drift. In the case of a virtual drift, the input data distribution within a class changes over time. For instance, an outbreak of a disease may change the distribution of medical news articles within the interesting category for users, resulting in a virtual concept drift.

On the basis of the type of distribution change over time, abrupt and gradual drifts are two major drift types. In abrupt drift, the distribution of data changes suddenly, and a concept is replaced by another at time t, while in gradual drift this change occurs smoothly over a time interval. Figure 4 illustrates the process. t is the moment that sudden concept drift happens. Similarly, t_1, t_2 are the start and end times of a gradual drift—with a and b probability values. Depending on the time period of drift, gradual drift can happen fast or slow. To include different variety of gradual drifts, we define *intensity* of drift as:

$$\text{DI (Drift Intensity)} = \frac{b-a}{t_2-t_1}. \tag{10}$$

We consider $(b-a)$ to be always 1, and t_2-t_1 as the number of data instances during the drift.[5] Therefore, we can rewrite drift intensity as 1/(number of data instances during the drift).

We use drifts with different intensity values. Tables 2 and 3 show the distribution of concept drift and intensity of drift in the 20NG and AG News datasets, respectively. DI values are randomly chosen, and columns show the latest data instance in the corresponding span— e.g., 3.5k showing the span of the stream from the beginning up to 3,500 instances. Moreover, + shows user is interested in the corresponding newsgroup, and – means otherwise. In the last row (1/DI), – indicates the absence of drift in the respective span.

[5]Drift intensity for the abrupt drift is ∞ and for a gradual drift is $0 < DI \leq 1$.

Table 3. Concept Drift Distribution and Drift Intensity for Both AGNews-Abrupt and AGNews-Gradual Text Streams. Columns Represent the Most Recent Data Instance within that Span

	10k	20k	30k	40k	50k	60k	70k	80k	90k	100k	110k	127.6k
World	+	-	+	-	-	+	-	+	-	+	-	+
Sports	+	-	-	+	+	-	+	-	-	+	-	+
Business	-	+	+	+	-	-	+	+	+	-	+	-
Sci/Tech	-	+	-	-	+	+	-	-	+	-	+	-
1/DI	-	5,000	1,000	500	100	500	2,500	250	2,000	1,000	10,000	500

Table 4. Summary of Datasets: Includes Eight Text and Five Numerical Streams; Three Synthetic and 10 Real Streams

Dataset	Size	# Features	# Classes	Type	Re/Syn	Drift Type
Email [30]	1,500	913	2	Text	Re	A&R
Spam [29, 30]	6,213	499	2	Text	Re	U
Usenet [31]	1,500	99	2	Text	Re	A&R
NewYorkTimes	16,000	300	2	Text	Re	U
AGNews-Ab	127,600	300	2	Text	Re	A
AGNews-Gr	127,600	300	2	Text	Re	G
20NG-Ab	15,102	300	2	Text	Re	A
20NG-Gr	15,102	300	2	Text	Re	G
Electricity [24]	45,312	8	2	Num	Re	U
Phishing [63]	11,055	46	2	Num	Re	U
MG2C2D [16]	200,000	2	2	Num	Syn	I&G
Moving Squares [44]	200,000	2	4	Num	Syn	I
Rotating Hyperplane [44]	200,000	10	2	Num	Syn	I

5 EXPERIMENTS

In this section, we describe our experimental setup; then, results are given and experimental analyses are provided with some discussion. We present our results on two sets of data streams: (i) Evolving numerical streams and (ii) Evolving text streams. We use numerical streams to evaluate the concept drift handling ability of the models in evolving environments. The performance of the models is then assessed in text streams, where the incoming data is more complex compared to numerical data.

5.1 Experimental Setup

Datasets. In our experiments, we use eight text and five numerical stream classification datasets. To compare the performance of AdaNEN and the baseline models in concept drift handling, we use numerical streams with different types of drift. Then, we evaluate AdaNEN and the baseline models in the evolving text stream classification task.

Table 4 summarizes statistics of the datasets. There are 10 real (Re) and three Synthetic (Syn) datasets with four different drift types: Abrupt (A), Gradual (G), Incremental (I), and Recurring (R). Unknown drift type is denoted by (U) in the same table. The data sizes of the available evolving text classification datasets are relatively small. To address this issue, we construct and use four datasets based on 20NG and AG News datasets with 15,102 and 127,600 data instances (see Section 4). The constructed datasets are AGNews-Ab, AGNews-Gr, 20NG-Ab, and 20NG-Gr. The remaining text datasets are as follows:

- **Usenet** [31]: A collection of 1,500 news articles, labeled as interesting or not interesting for a user. In the dataset, concept drift is defined as the change in user personal interest, which occurs at every 300 data instances. Each data item has 99 attributes, which are the words in the dataset. Each instance is a binary vector of length 99, where the feature value is one if a word exists in that instance and zero otherwise;
- **Spam Filtering** [29, 30]: Contains 9,324 data instances and 499 attributes (words). This dataset contains gradual concept drift, and the characteristics of spam messages change over time. Each instance is a binary vector of length 499, where the feature is one if a word exists in that instance and zero otherwise;
- **Email List** [30]: Contains 1,500 articles from medicine, space, and baseball categories, classified as interesting or junk to the end-user. Concept drift in the dataset is defined as the change in users' interest in the categories. The vocabulary of the dataset consists of 913 words, and the features are binary values, representing the existence of the words in the documents;
- **New York Times (NYT)**: Contains around 16,000 articles from January 1, 2020–December 31, 2020. Our aim in this dataset is to predict popular articles, which is defined by is_popular feature in the dataset. We obtained the dataset from Kaggle.[6]

We use word2vec embeddings [47] to extract features by averaging the constituting word vectors. For the Spam dataset, the word list is not available and data instances are provided as binary features. For the same reason, we are not able to run the BERT model on the Spam dataset.

Evaluation Method. We used the prequential test-then-train approach, similar to the literature [9, 19, 45], for training and evaluating classifiers. In the test-then-train approach, each incoming data instance is initially used for testing and subsequently for training the models. The concept of training epochs is not applied, as each data instance is processed only once in an incremental manner.

Methods are compared using their prequential accuracy, average runtime, and average rank. The prequential accuracy of the models is calculated by sequentially testing each instance in the stream, updating the model with the true class value of the instance right after testing, and keeping a running average of the accuracy over evaluation windows. The average runtime of a model on a data stream is obtained by dividing its overall runtime by the total number of instances in that stream. We use the Friedman's test rank [14] to calculate the average accuracy and runtime ranks of the models. For a given model j, its average rank R_j is calculated as $R_j = \frac{1}{N}\sum_{i=1}^{N} r_i^j$, where r_i^j is the rank of the jth model on the ith data stream and N is the total number of data streams.

System Specification. We perform our experiments with ensemble models on a machine equipped with an Intel Core i7-4702MQ @ 2.2 GHz processor and 8 GB DDR3 RAM. For neural models, except for BERT, we use an Nvidia GeForce GTX 960M GPU with 4 GB DDR5 memory. As BERT requires a significant amount of GPU memory, we used the *Google Colab* platform to run our experiments with the BERT model.

Baselines. To compare the performance of AdaNEN, we use eight ensemble and four neural models. We use the first 200 data instances for a given stream to determine and tune the hyper-parameters for each of our datasets. By leveraging the first 200 data instances from each data stream, we determine the optimal hyperparameters for each method across all data streams. The hyperparameters remain consistent across all experiments, ensuring the uniformity of our experimental conditions and results.

Ensemble Models. (i) **Additive Expert Ensemble (AddExp)** [36]: removes the weakest base classifier to overcome the concept drift in a stream; (ii) **Accuracy Weighted Ensemble (AWE)**

[6]https://www.kaggle.com/benjaminawd/new-york-times-articles-comments-2020

Table 5. Major Hyperparameter Values of AdaNEN and the Baseline Methods (Consistent across All Data Streams)

	Model	Hyperparameters
Ensemble	AddExp [36]	Max estimators count: 5, base estimator: NaiveBayes, beta: 0.83, gamma: 0.13, pruning: weakest
	AWE [65]	Max estimators count: 10, base estimator: NaiveBayes, window size: 100, folds count: 10
	DWM [37]	Max estimators count: 10, base estimator: NaiveBayes, period: 100, beta: 0.6, theta: 0.01
	GOOWE [9]	Max estimators count: 10, window size: 100
	HAT [5]	Grace period: 200, split confidence: 0.0000001, tie threshold: 0.05, leaf prediction strategy: perceptron
	KNN-Adwin [4]	Nearest neighbors count: 5, max window size: 2000, leaf size: 30
	Learn++.NSE [17]	Base estimators count: 15, base estimator: decision tree, crossing point: 0.5, slope: 0.5, pruning: age-based
	Oza-Adwin [53]	Estimators count: 10, base estimator: KNN (neighbors:8, max window size:2000, leaf size:30)
Neural	SGD	Learning rate: 5e-3, hidden layers architecture: [128, 64, 32, 16]
	HBP [58]	Hidden layers count: 4, hidden layer size: 16, learning rate: 1e-2, β: 0.99, regularization parameter (s): 0.2
	Adam	Initial learning rate: 1e-2, α: $1e-5$, hidden layers architecture: [128, 64, 32, 16]
	BERT [15]	Max seq. len: 512, optimizer (Adam): 5e-5, dropout: 0.1, attention head count: 12, transformer block count: 12
	AdaNEN (Our model)	β: 0.8, regularization parameter (s): 0.2, architecture: [128, 64, 32, 16], output layers: 3 (lrs: [1e-1, 1e-2, 1e-3])

[65]: uses test accuracy result to determine the weight of each classifier; (iii) **Dynamic Weight Majority (DWM)** [37]: adds and removes base classifiers and adjusts the weighting based on its performance on the test data; (iv) **Geometrically Optimum and Online-Weighted Ensemble (GOOWE)** [9]: weights the classifiers based on the distance of their vote vectors and ideal points and a least square problem is solved to find the final weights; (v) **Hoeffding Adaptive Tree classifier (HAT)** [5]: is a tree-based model with ADWIN detection method for handling concept drift; (vi) **KNN-ADWIN** [4]: KNN classifier and an ADWIN detector for handling the concept drift; (vii) **Learn++.NSE** [17]: a modified version of the famous NSE algorithm, which adds the capability of handling concept drift in a stream environment; (viii) **Oza-Adwin** [53]: an improved Online Bagging model with ADWIN change detector designed for non-stationary environments.

Neural Models. (i) **SGD** and **Adam**: Feedforward Neural Networks with **Stochastic Gradient Descent (SGD)** and Adam optimizers; (ii) **Hedge Backpropagation (HBP)** [58]: an online, neural model for large-scale data; (iii) **Bidirectional Transformers for Language Understanding (BERT)** [15]: a recent deep neural network model used for text classification. We tune and run the pre-trained model on default parameters.

Table 5 lists the major hyperparameter values of the compared methods. A comparative analysis investigating the impact of hyperparameters on AdaNEN's performance is discussed in Section 6.3.

5.2 Motivation for Effective Concept Drift Handling

Evolving numerical stream classification is less challenging due to the absence of additional complexity associated with text data. Due to this, the concept drift handling strategy of the classification methods contributes more to their success. This enables a better comparison of

Table 6. Prequential Accuracy (%) Results for Numerical Streams: Overall Model Performance across All Data Instances within the Corresponding Stream with Best Values Marked with Bold Text

	Model	Electricity	Phishing	R-Hyperplanes	Moving Squares	MG2C2D	Avg. Rank
Ensemble	AddExp [36]	73.63	91.38	81.90	32.73	57.06	9.00
	AWE [65]	75.63	90.53	89.37	48.93	92.19	5.40
	DWM [37]	80.10	91.82	89.62	70.63	91.92	3.80
	GOOWE [9]	76.39	90.16	87.72	70.76	91.02	5.20
	HAT [5]	81.88	89.33	85.56	75.43	92.56	4.20
	KNN-Adwin [4]	77.17	93.64	82.66	37.91	92.51	5.00
	Learn++.NSE [17]	74.99	89.90	71.54	41.38	90.03	8.40
	Oza-Adwin [53]	76.32	92.81	82.27	43.62	92.58	5.00
Neural	SGD	56.55	76.31	57.17	69.41	61.59	10.00
	HBP [58]	73.77	70.55	70.64	29.04	87.86	10.40
	Adam	59.73	74.55	78.57	28.71	85.73	10.60
	AdaNEN (Our model)	**89.64**	**94.38**	**90.65**	**99.06**	**92.63**	**1.00**

Table 7. Average Running Time per Numerical Data Item (in Centiseconds): Calculated by Total Runtime/Instance Count with Best Values Marked with Bold Text

	Model	Electricity	Phishing	R-Hyperplanes	Moving Squares	MG2C2D	Avg. Time (Avg. Rank)
Ensemble	AddExp [36]	0.11	0.73	0.14	0.10	0.08	0.23 (3.60)
	AWE [65]	0.23	1.07	0.34	0.20	0.13	0.39 (6.00)
	DWM [37]	0.11	0.68	0.15	0.08	0.07	0.22 (2.80)
	GOOWE [9]	0.28	1.11	0.40	0.21	0.15	0.43 (7.60)
	HAT [5]	**0.07**	0.22	**0.10**	**0.06**	**0.07**	**0.10 (1.40)**
	KNN-Adwin [4]	0.32	1.77	0.42	0.24	0.19	0.59 (8.80)
	Learn++.NSE [17]	0.27	0.24	0.34	0.29	0.36	0.30 (7.20)
	Oza-Adwin [53]	0.53	2.64	0.85	0.77	0.35	1.03 (11.00)
Neural	SGD	0.18	**0.16**	0.17	0.18	0.16	0.17 (3.80)
	HBP [58]	0.83	0.71	0.76	0.83	0.74	0.77 (10.80)
	Adam	0.19	**0.16**	0.19	0.21	0.18	0.19 (4.80)
	AdaNEN (Our model)	0.52	0.45	0.47	0.52	0.46	0.48 (9.20)

AdaNEN and the baseline models' concept drift handling method. Tables 6 and 7 present our accuracy and runtime results on numerical streams. The average ranks of the models are given in the last column of the tables.

We observe that except for AdaNEN, other neural models achieved the lowest ranks. This is expected, given that, excluding HBP, other neural baselines do not employ a concept drift handling mechanism. It appears that the drift adaptation technique of HBP is also ineffective. This clarifies the need for an effective concept drift handling strategy in evolving stream classification methods.

In terms of runtime efficiency, AdaNEN achieves average runtime of 0.48 centiseconds and an average rank of 9.20 on numerical data streams. This is 53% faster than the least efficient model, Oza-Adwin (AdaNEN: 0.48 vs. Oza-Adwin: 1.03) and 54% slower than the second average ranked model, DWM (AdaNEN: 0.48 vs. DWM: 0.22).

5.3 Evolving Text Stream Classification

5.3.1 Overall Accuracy and Runtime Performance Evaluation. Tables 8 and 9 present our results for prequential accuracy and running time of the models on text streams. For example, the total prediction accuracy of AddExp on the entire Spam text stream is 91.30%, and the model takes 8.48

Table 8. Prequential Accuracy (%) Results for Text Streams: Overall Model Performance across All Data Instances within the Corresponding Stream with Best Values Marked with Bold Text

	Model	Spam	Email	Usenet	20NG-Ab	20NG-Gr	AGNews-Ab	AGNews-Gr	NYT	Avg. Rank
Ensemble	AddExp [36]	91.30	55.65	64.76	64.10	64.87	52.36	51.42	64.52	7.25
	AWE [65]	76.78	40.48	51.36	67.27	65.48	50.32	49.88	61.80	10.00
	DWM [37]	90.61	56.53	69.86	72.10	70.84	69.44	67.49	53.51	5.00
	GOOWE [9]	81.65	54.63	60.54	64.03	65.93	71.97	67.85	64.46	6.50
	HAT [5]	90.77	55.66	64.49	63.86	64.67	72.07	69.84	62.87	6.00
	KNN-Adwin [4]	94.40	57.28	56.80	68.01	67.27	65.74	64.12	61.36	5.38
	Learn++.NSE [17]	83.63	46.19	56.33	64.87	64.46	50.03	49.87	60.05	10.50
	Oza-Adwin [53]	92.41	54.01	57.76	67.97	68.39	62.58	61.07	61.33	6.63
Neural	SGD	83.59	78.91	68.37	62.24	70.04	54.42	53.16	49.97	7.63
	HBP [58]	83.47	58.16	59.73	56.03	55.77	50.98	50.54	52.47	10.13
	Adam	93.29	78.23	72.65	74.88	65.23	61.17	61.74	57.82	5.00
	BERT [15]	*	53.45	71.27	55.84	54.51	58.18	56.21	52.66	9.57
	AdaNEN (Our model)	**97.31**	**85.85**	**76.35**	**80.23**	**72.00**	**74.37**	**72.38**	**64.66**	**1.00**

*We are unable to run BERT on the Spam dataset, because the word list is not provided.

Table 9. Average Running Time per Text Data Item (in Centiseconds): Calculated by Total Runtime/Instance Count with Best Values Marked with Bold Text

	Model	Spam	Email	Usenet	20NG-Ab	20NG-Gr	AGNews-Ab	AGNews-Gr	NYT	Avg. Time (Avg. Rank)
Ensemble	AddExp [36]	8.48	10.43	2.08	6.43	7.69	5.88	5.65	2.52	6.15 (8.50)
	AWE [65]	10.08	5.37	0.82	6.89	8.32	16.65	15.84	6.19	8.77 (8.88)
	DWM [37]	7.70	6.57	1.39	6.37	7.74	6.16	5.93	2.61	5.56 (8.13)
	GOOWE [9]	7.95	2.86	0.38	7.64	8.94	17.73	14.89	5.36	8.22 (8.25)
	HAT [5]	2.43	6.00	0.61	1.69	2.23	2.19	2.23	1.89	2.41 (6.00)
	KNN-Adwin [4]	16.51	8.13	1.21	11.16	12.87	23.08	22.78	7.82	12.95 (10.63)
	Learn++.NSE [17]	**0.21**	**0.08**	**0.08**	0.31	**0.35**	3.71	1.64	2.95	1.17 (3.00)
	Oza-Adwin [53]	18.27	23.24	2.50	19.23	21.58	43.99	40.63	15.73	23.15 (12.50)
Neural	SGD	0.23	0.72	0.25	**0.28**	0.37	**0.83**	0.92	**0.59**	**0.52 (1.75)**
	HBP [58]	0.97	1.91	1.21	1.12	1.33	1.56	1.65	1.63	1.42 (5.13)
	Adam	0.26	0.90	0.27	0.34	0.36	1.36	1.37	0.65	0.69 (2.88)
	BERT [15]	*	3.56	1.63	17.26	17.48	47.86	46.34	23.51	19.70 (11.57)
	AdaNEN (Our model)	0.68	0.78	0.84	1.09	0.93	0.84	**0.67**	0.70	0.82 (3.50)

*We are unable to run BERT on the Spam dataset, because the word list is not provided.

centiseconds, on average, to test-then-train each data instance. Our results indicate that AdaNEN outperforms the baseline models by achieving the highest accuracy values in text streams with a conservative runtime.

Interestingly, simple feed-forward neural networks with Adam and SGD optimizers perform comparable to most ensemble baselines, designed for concept drift handling. This is contrary to our observation in evolving numerical stream classification. For instance, in the Email text stream, SGD and Adam achieve the 2nd and 3rd ranks, respectively. We believe that the good performance of Adam and SGD on text streams stems from the strength of neural networks in text classification. This implies that despite the success of existing ensemble methods in concept drift handling and numerical data stream classification, they might not be directly applicable to evolving text stream classification. This is probably due to the extra complexity of text compared to numerical data.

Among the non-neural ensemble approaches, we see a general consistency for prediction accuracy, despite their big difference in the running time of the methods. In terms of accuracy, there is no single winner among the datasets for ensemble approaches; KNN-Adwin for Spam and Email, DWM for Usenet, 20NG-Abrupt, and 20NG-Gradual, HAT for AGNews-Ab and AGNews-Gr, and AddExp for NYT. However, in terms of running time, the Learn++.NSE method is quite fast.

One surprising observation is the low predictive performance of BERT compared to other neural and non-neural approaches. This can be explained by the deep architecture of BERT with many stacked layers and millions of parameters that takes a long time to adapt itself with every change in the data distribution, which is not suitable for evolving environments. It would be interesting to study the network distillation techniques in the literature for compressing the network size and lowering the number of parameters [59]. As it is out of the scope of our study, we leave it for future work.

In terms of running time, AdaNEN is much more efficient than the majority of ensemble-based approaches and BERT (Table 9, last column). In the case of ensemble-based approaches, their lower efficiency can be due to the complexity of updating every single model individually—i.e., maintenance cost. In fact, with proper distributed implementation over multiple nodes, some of the running time costs might be reduced. For the case of the BERT model, the gap is an expected result, because deeper models need a considerable amount of time for training and testing even on GPU. Besides, our model is favorably efficient for a text stream environment, as it achieves an average efficiency rank of 3.5 on text streams. For instance, on Spam text stream (Tables 8 and 9) AdaNEN is around 24 times faster than the second-best approach (AdaNEN: 0.68 vs. KNN-Adwin: 16.51), while providing 3% increase in the prediction accuracy (AdaNEN: 97.31% vs. KNN-Adwin: 94.40%).

By comparing the accuracy results of AdaNEN and the baseline models on numerical and text streams from Tables 6 and 8, we observe that, in general, all models perform better on numerical streams. This bolsters the idea that text stream classification is more challenging. Tables 7 and 9 further support this, as models take more test-then-train time per text data instance compared to numerical data. Neural baselines are ineffective in evolving numerical stream classification while performing better on evolving text streams. Adam performs quite well on text streams by achieving the second-best average rank (Table 8, last column). The effectiveness of neural networks in text classification may be a reason for this. Comparing the accuracy results of the models on numerical and text streams confirms this.

5.3.2 Prequential Evaluation. Figure 5 shows the prequential evaluation of the top five methods for evolving text streams studied with the highest average rank—accuracy values over time. The concept drift locations in the 20NG and AGNews text streams are marked by vertical dashed lines. For the remaining streams, the drift locations are unknown, indirectly suggested by sudden drops in prequential accuracy. We observe that in concept drifting points, our model is able to adapt to the changes quickly compared to other baseline models. For example, we observe that in the 20NG-Abrupt text stream (Figure 5(d)), methods have comparable performance until the first concept drift point, which occurs around the 7th evaluation window. AdaNEN demonstrates the smallest decline in prequential accuracy and emerges as the top-performing method following the initiation of the concept drift. In the Email data stream (Figure 5(b)), our model returns to its previous performance faster than other baselines in concept drift points. In fact, we have similar observations for all data streams.

The 20NG and AGNews text streams prequential analyses in Figure 5(d)–(g) show that AdaNEN is better with Abrupt drifts compared to gradual drifts. However, as data flows with the gradual drift, AdaNEN successfully corrects itself and outperforms other strong baselines. For example, in Figure 5(e) AdaNEN shows the lowest performance at the 17th evaluation window. Despite this, AdaNEN recovers promptly, demonstrating its superior adaptability to drift in comparison to the baseline methods. The overall accuracy values from Tables 6 and 8 reported this. While none of the

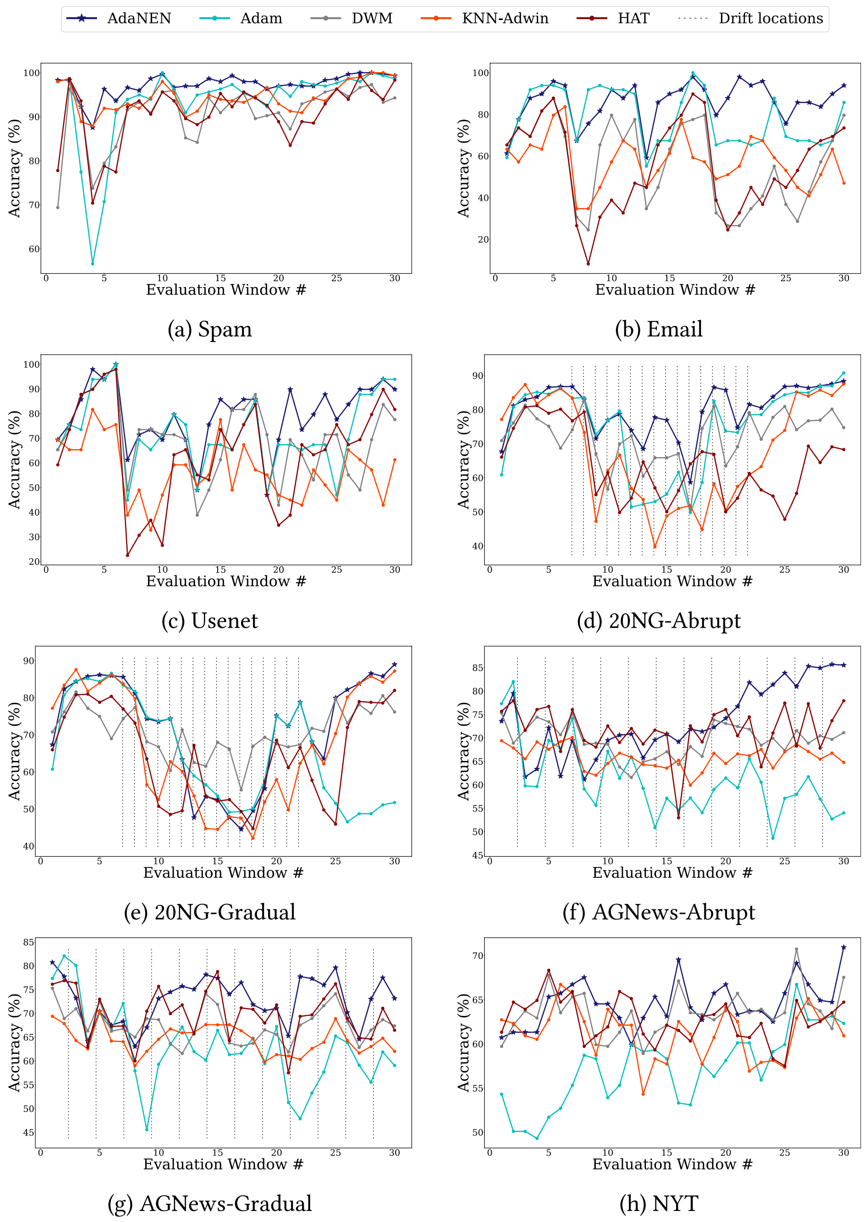


Fig. 5. Prequential accuracy results for evolving text streams. The top-5 models with the best average rank are plotted for each text stream. AdaNEN ranks first on all data streams. DWM, KNN-Adwin, and HAT are the top-3 best-performing ensemble models, and Adam is the best-performing neural model. The drift locations in the 20NG and AGNews streams are marked by vertical dashed lines; for the remaining streams, the drift locations are unknown.

other strong baselines are always performing better, we notice that DWM, HAT, and KNN-Adwin from ensemble baselines are performing quite well compared to others. Neural baselines are ineffective in evolving numerical stream classification while performing better on evolving text streams. Adam performs quite well on text streams by achieving second-best average rank.

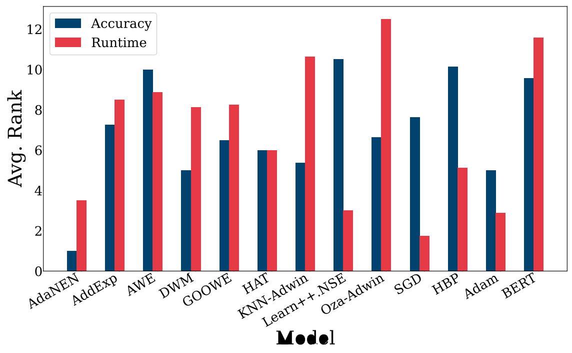


Fig. 6. Average accuracy (Table 8) and runtime (Table 9) ranks for evolving text streams (lower is better). AdaNEN ranks first in terms of accuracy and achieves an average runtime ranking of 3.5 across all text streams.

5.3.3 Effectiveness and Efficiency Analysis. To compare the accuracy and runtime of AdaNEN and the baseline methods, the average accuracy and runtime ranks of the models on evolving text streams are plotted in Figure 6. Note that all models are run on systems that match their nature; for example, BERT on Google Colab.

We evidence that in terms of accuracy, AdaNEN outperforms the baseline models with a conservative efficiency. Considering the characteristics of data stream environments, a model is considered a good fit if it exhibits strong performance in both accuracy and runtime. For instance, we can clearly say that compared to AWE, AddExp is a better choice in our experiments, since it reaches better average accuracy and runtime rank. Similarly, GOOWE is a better option compared to BERT.

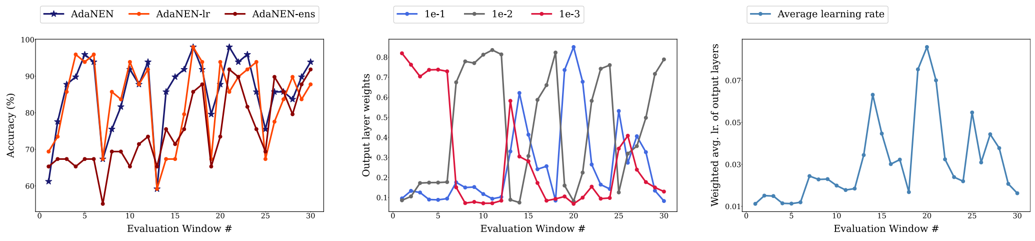
6 FURTHER ANALYSES AND DISCUSSION

In this section, we further investigate differences in the performance of our model and baselines through ablation analysis, ensemble weight assignment analysis, hyperparameter analysis, and statistical analysis.

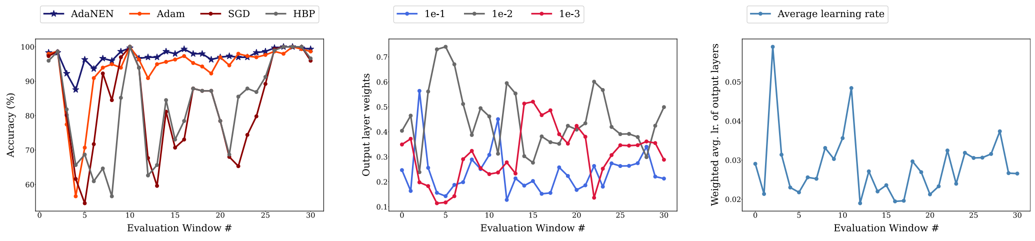
6.1 Ablation Analysis

To further study the AdaNEN's neural ensemble architecture, here, we conduct an ablation study and show that the ensemble approach and different learning rates are highly important factors for the success of our method. For our study, we consider the following variants of the AdaNEN model by disabling the ensemble module or using the same learning rate in the output layers: (i) ***Complete***: The complete version of AdaNEN; (ii) ***AdaNEN-lr***: Output layers with identical learning rate (1e-2) are used to assess the effect of using output layers with different learning rates; (iii) ***AdaNEN-ens***: Ensemble method is disabled and majority voting is replaced to assess the effect of ensemble method.

Figure 7(a) shows our ablation study results for the Email dataset. We only report the Email data stream here and confirm that similar results are observed for other streams. We observe that the complete AdaNEN model outperforms the other variants by improving the accuracy by 6.78% (AdaNEN: 85.85% vs. AdaNEN-lr: 80.40%) and 18.90% (AdaNEN: 85.85% vs. AdaNEN-ens: 72.20%). Between AdaNEN-lr and AdaNEN-ens, we observe that the latter has more performance deterioration, indicating that the ensemble module plays a more important role than using different learning rates in the AdaNEN model.



(a) Email



(b) Spam

Fig. 7. (a) Ablation analysis and (b) Ensemble weight assignment analysis of AdaNEN on the Email and Spam data streams. The left-to-right sequence of plots represents the prequential accuracy, ensemble weights of the output layers, and their weighted average learning rate. Increases in the weighted average learning rate correspond to the local dips in the accuracy, signaling AdaNEN's adaptive response to concept drift-induced accuracy declines.

6.2 Ensemble Weight Assignment Analysis

To show how AdaNEN adapts to the changes in a stream by changing the ensemble weights of output layers with different learning rates, we track the ensemble weights for output layers. Figure 7(b) shows ensemble weights, weighted average learning rate of output layers, and accuracy plots of AdaNEN along with Adam, SGD, and HBP neural models, for the Spam data stream. We use three output layers in our experiments with $1e-1$, $1e-2$, and $1e-3$ learning rates. Our observations show that at concept drift points AdaNEN assigns more weights to the output layers with higher learning rates to learn the new trend in data faster. Similarly, at the end points of concept drift, the learning rate is returned to the previous level. For instance, in the 3rd evaluation window of the Spam dataset, the average learning rate of AdaNEN increases from 0.021 to 0.059 to adapt to the drift. It can be seen from the accuracy plot that AdaNEN maintains its performance at this point, while the accuracy of other neural models decreases dramatically. Another example is the 11th evaluation window.

6.3 Hyperparameter Analysis

In this section, we examine the impact of hyperparameters on the performance of AdaNEN. The major hyperparameters of AdaNEN are: (1) $\mathbf{l}$: Number of hidden layers, (2) $\mathbf{m}$: Number of output layers, (3) $\boldsymbol{\beta}$: Ensemble weight update penalization parameter, (4) $\mathbf{s}$: Regularization parameter, (5) $\boldsymbol{lr_j}$: Learning rates of the output layers (in e-notation). Table 10 shows AdaNEN's performance on the 20NG-Ab dataset with different hyperparameter values. The optimal value of the hyperparameters obtained by grid search is given in the last row of the table. To assess the effect of each hyperparameter, in each row of the table value of one hyperparameter is changed and the other hyperparameters are set to their optimal values.

Table 10. Hyperparameter Analysis Results of AdaNEN

Hyperparameter					
l	m	β	s	lr_j	Accuracy (%)
2	3	0.8	0.2	[1e-1, 1e-2, 1e-3]	55.86
6	3	0.8	0.2	[1e-1, 1e-2, 1e-3]	77.86
8	3	0.8	0.2	[1e-1, 1e-2, 1e-3]	74.07
4	**2**	0.8	0.2	[1e-1, 1e-3]	68.18
4	**4**	0.8	0.2	[1e-1, 1e-2, 1e-3, 1e-4]	70.15
4	**6**	0.8	0.2	[5e-1, 1e-1, 5e-2, 1e-2, 5e-3, 1e-3]	56.43
4	3	**0.4**	0.2	[1e-1, 1e-2, 1e-3]	74.83
4	3	**0.6**	0.2	[1e-1, 1e-2, 1e-3]	73.93
4	3	**0.9**	0.2	[1e-1, 1e-2, 1e-3]	78.09
4	3	0.8	**0.1**	[1e-1, 1e-2, 1e-3]	77.16
4	3	0.8	**0.4**	[1e-1, 1e-2, 1e-3]	74.30
4	3	0.8	**0.8**	[1e-1, 1e-2, 1e-3]	74.38
4	3	0.8	0.2	**[5e-1, 5e-2, 5e-3]**	69.84
4	3	0.8	0.2	**[5e-2, 5e-3, 5e-4]**	76.74
4	3	0.8	0.2	**[1e-2, 1e-3, 1e-4]**	77.72
4	**3**	**0.8**	**0.2**	**[1e-1, 1e-2, 1e-3]**	**80.23**

Columns represent five major hyperparameters. Each row alters one hyperparameter while holding others at their optimal values to evaluate the impact of each hyperparameter. The last row provides the optimal hyperparameter values.

The number of hidden layers (l) is an important factor in the performance of feed-forward neural networks. Using a lower or higher number of hidden layers reduces the performance of neural networks by limiting the model capacity or overfitting the data. Using a concatenation layer enables AdaNEN to use an adjusted number of hidden layers to fit the complexity of the current data. We observe that, by using a lower number of hidden layers, the performance of AdaNEN deteriorates, while increasing the number of hidden layers has less performance deterioration. This observation validates our idea that AdaNEN is able to use the required number of hidden layers to adjust to the complexity of current data.

Among other hyperparameters, we observe that the number of output layers (m) and the regularization parameter (s) are, respectively, the most- and the least-important factors in the performance of the model. This was expected, since the number of output layers is effective in both adjusting the learning rate by the ensemble module and the use of different learning rates in the output layer. We observed similar results for the other streams.

6.4 Statistical Analysis

To statistically analyze the performance of the compared methods, we applied the Friedman test with Bonferroni-Dunn post hoc analysis [14] to the average accuracy ranks of the models across all numerical and text streams. The BERT model was excluded from this statistical analysis due to its incompatibility with numerical and spam text streams. We employ the Friedman's test rank to obtain the average accuracy ranks of the models, where each model's rank is calculated as its mean rank across all data streams (see Section 5). Among t models compared on a data stream, the best-performing model gets the rank of 1, and the worst model gets the rank of t. Table 11 presents the average accuracy ranks of the models (excluding BERT) across 13 numerical and text streams.

Table 11. Average Accuracy Ranks of the Models (Excluding BERT) across 13 Numerical and Text Streams (Lower Is Better)

Model	AdaNEN	Adam	AddExp	AWE	DWM	GOOWE	HAT	HBP	KNN-Adwin	Learn++.NSE	Oza-Adwin	SGD
Avg. rank	1.000	7.154	7.692	7.923	4.461	5.923	5.231	9.923	5.154	9.385	5.923	8.231

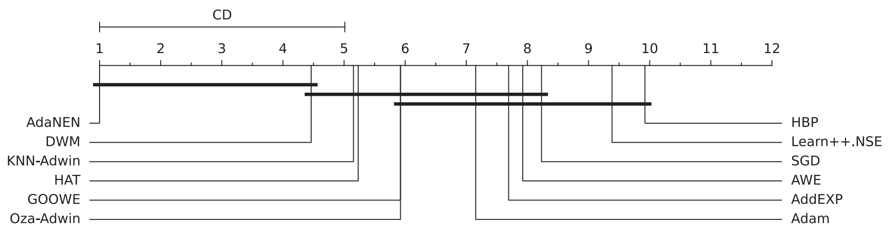


Fig. 8. The Bonferroni-Dunn statistical analysis of the average accuracy rank results (Table 11) for AdaNEN and the baseline models (excluding BERT) across 13 numerical and text streams. AdaNEN statistically significantly performs better than 11 baselines (Table 4), excluding DWM (CD = 4.013).

We obtained the Friedman test statistic as 65.071 and the p-value as 1.045e-9. The obtained p-value rejects the null hypothesis at a significance level of 0.05 (p-value < 0.05), suggesting that at least one of the methods performs significantly differently.

Following the Friedman test, we apply the Bonferroni-Dunn post hoc analysis to identify which methods differ significantly in their performance. Two methods are considered to have statistically significantly different performances if the difference between their average ranks is greater than the **critical distance (CD)** value. The CD value is obtained as $CD = q_{\alpha,k}\sqrt{\frac{k(k+1)}{6n}}$, where n is the number of data streams ($n = 13$), k is the number of models ($k = 12$), and $q_{\alpha,k}$ is a critical value from the studentized range statistic (q) distribution table. For $\alpha = 0.05$ and $k = 12$, $q_{\alpha,k} \approx 2.838$.

We obtained the CD for the Bonferroni-Dunn test at a significance (α) level of 0.05 as CD = 4.013. Figure 8 shows the results of our statistical test on the average accuracy rank of the models across 13 numerical and text streams. Our results indicate that AdaNEN statistically significantly outperforms 11 neural and ensemble models, except for the DWM model.

7 CONCLUSION

We introduce AdaNEN, an online neural classifier, suitable for evolving text stream environments. We highlight the real-world and specific obstacles that text data streams pose, with a particular focus on concept drift adaptation. The research community suffers from the lack of large-scale evolving text streams. Our concept drift introducing approach offers a solution to this problem by using real-world datasets. Our experiments demonstrate that with comparable efficiency, AdaNEN outperforms strong approaches in the literature, including ensemble and neural models. We demonstrate that AdaNEN is a suitable choice for evolving text stream environments, and mega-models with millions of parameters might not be a practical solution.

We are interested in further extending our model design and data generation method. More specifically, we are working on re-purposing huge text datasets such as Amazon [46] and MIND [66]. Our future works will also focus on extending our evolving text stream generation method and AdaNEN to multi-label settings. Incorporating the adjacent field of sequence-aware recommendation [55] into our problem setup is another opportunity for further research.

REFERENCES

[1] Charu C. Aggarwal. 2014. Mining text and social streams: A review. *ACM SIGKDD Explor. Newslett.* 15, 2 (2014), 9–19.

[2] Charu C. Aggarwal, Stephen C. Gates, and Philip S. Yu. 2004. On using partial supervision for text categorization. *IEEE Trans. Knowl. Data Eng.* 16, 2 (2004), 245–255.

[3] Charu C. Aggarwal, Philip S. Yu, Jiawei Han, and Jianyong Wang. 2003. A framework for clustering evolving data streams. In *Proceedings of the VLDB Conference*. Elsevier, 81–92.

[4] Albert Bifet and Ricard Gavalda. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 443–448.

[5] Albert Bifet and Ricard Gavaldà. 2009. Adaptive learning from evolving data streams. In *Proceedings of the International Symposium on Intelligent Data Analysis*. Springer, 249–260.

[6] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. 2010. Leveraging bagging for evolving data streams. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 135–150.

[7] Hamed Bonab and Fazli Can. 2019. Less is more: A comprehensive framework for the number of components of ensemble classifiers. *IEEE Trans. Neural Netw. Learn. Syst.* 30, 9 (2019), 2735–2745.

[8] Hamed R. Bonab and Fazli Can. 2016. A theoretical framework on the ideal number of classifiers for online ensembles in data streams. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*. 2053–2056.

[9] Hamed R. Bonab and Fazli Can. 2018. GOOWE: Geometrically optimum and online-weighted ensemble classifier for evolving data streams. *ACM Trans. Knowl. Discov. Data* 12, 2 (2018), 1–33.

[10] Dariusz Brzezinski and Jerzy Stefanowski. 2013. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Trans. Neural Netw. Learn. Syst.* 25, 1 (2013), 81–94.

[11] Fazli Can, Seyit Kocberber, Ozgur Baglioglu, Suleyman Kardas, H. Cagdas Ocalan, and Erkan Uyar. 2010. New event detection and topic tracking in Turkish. *J. Am. Societ. Inf. Sci. Technol.* 61, 4 (2010), 802–819.

[12] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. 2017. AdaNet: Adaptive structural learning of artificial neural networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 874–883.

[13] Matheus Bernardelli De Moraes and Andre Leon Sampaio Gradvohl. 2021. A comparative study of feature selection methods for binary text streams classification. *Evolv. Syst.* 12, 4 (2021), 997–1013.

[14] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* 7, Jan. (2006), 1–30.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.

[16] Karl B. Dyer, Robert Capo, and Robi Polikar. 2013. Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. *IEEE Trans. Neural Netw. Learn. Syst.* 25, 1 (2013), 12–26.

[17] Ryan Elwell and Robi Polikar. 2011. Incremental learning of concept drift in nonstationary environments. *IEEE Trans. Neural Netw.* 22, 10 (2011), 1517–1531.

[18] Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55, 1 (1997), 119–139.

[19] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Comput. Surv.* 46, 4 (2014), 1–37.

[20] Ömer Gözüaçık, Alican Büyükçakır, Hamed Bonab, and Fazli Can. 2019. Unsupervised concept drift detection with a discriminative classifier. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2365–2368.

[21] Ömer Gözüaçık and Fazli Can. 2021. Concept learning using one-class classifiers for implicit drift detection in evolving data streams. *Artif. Intell. Rev.* 54, 5 (2021), 3725–3747.

[22] Ege Berkay Gulcan and Fazli Can. 2023. Unsupervised concept drift detection for multi-label data streams. *Artificial Intelligence Review* 56, 3 (2023), 2401–2434.

[23] Donghong Han, Christophe Giraud-Carrier, and Shuoru Li. 2015. Efficient mining of high-speed uncertain data streams. *Appl. Intell.* 43, 4 (2015), 773–785.

[24] M. Harries and University of New South Wales. 1999. School of computer science, and engineering. 1999. Splice-2 Comparative evaluation: electricity pricing. (1999). https://books.google.com.tr/books?id=1Zr1vQAACAAJ

[25] Juan I. G. Hidalgo, Silas G. T. C. Santos, and Roberto S. M. Barros. 2021. Dynamically adjusting diversity in ensembles for the classification of data streams with concept drift. *ACM Trans. Knowl. Discov. Data* 16, 2, Article 31 (July 2021), 20 pages. DOI: https://doi.org/10.1145/3466616

[26] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming auto-encoders. In *Proceedings of the International Conference on Artificial Neural Networks*. Springer, 44–51.

[27] Geoff Hulten, Laurie Spencer, and Pedro Domingos. 2001. Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 97–106.

[28] A. K. Jumani, M. H. Mahar, F. H. Khoso, and M. A. Memon. 2018. Online text categorization system using support vector machine. *Sindh Univ. Res. J. (Sci. Series)* 50, 01 (2018), 85–90.

[29] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2006. Dynamic feature space and incremental feature selection for the classification of textual data streams. In *Proceedings of ECML/PKDD-2006 International Workshop on Knowledge Discovery from Data Streams* (2006), 102–116.

[30] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2010. Tracking recurring contexts using ensemble classifiers: An application to email filtering. *Knowl. Inf. Syst.* 22, 3 (2010), 371–391.

[31] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis P. Vlahavas. 2008. An ensemble of classifiers for coping with recurring contexts in data streams. In *Proceedings of the European Conference on Artificial Intelligence (ECAI'08)*. 763–764.

[32] Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, and Khaled Ghédira. 2018. Discussion and review on evolving data streams and concept drift adapting. *Evolv. Syst.* 9, 1 (2018), 1–23.

[33] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *Proceedings of the VLDB Conference*. 180–191.

[34] Jaeyoung Kim, Sion Jang, Eunjeong Park, and Sungchul Choi. 2020. Text classification using capsules. *Neurocomputing* 376 (2020), 214–221.

[35] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[36] Jeremy Z. Kolter and Marcus A. Maloof. 2005. Using additive expert ensembles to cope with concept drift. In *Proceedings of the 22nd International Conference on Machine Learning*. 449–456.

[37] J. Zico Kolter and Marcus A. Maloof. 2007. Dynamic weighted majority: An ensemble method for drifting concepts. *J. Mach. Learn. Res.* 8, Dec. (2007), 2755–2790.

[38] Dilek Küçük and Fazli Can. 2020. Stance detection: A survey. *ACM Comput. Surv.* 53, 1 (2020), 1–37.

[39] Jay Kumar, Junming Shao, Salah Uddin, and Wazir Ali. 2020. An online semantic-enhanced Dirichlet model for short text stream clustering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 766–776.

[40] Ken Lang. 1995. NewsWeeder: Learning to filter netnews. In *Proceedings of the Conference on Machine Learning*. Elsevier, 331–339.

[41] Peipei Li, Lu He, Xuegang Hu, Yuhong Zhang, Lei Li, and Xindong Wu. 2016. Concept based short text stream classification with topic drifting detection. In *Proceedings of the IEEE 16th International Conference on Data Mining (ICDM'16)*. IEEE, 1009–1014.

[42] Peipei Li, Yingying Liu, Yang Hu, Yuhong Zhang, Xuegang Hu, and Kui Yu. 2022. A drift-sensitive distributed LSTM method for short text stream classification. *IEEE Trans. Big Data* 9, 1 (2022), 341–357.

[43] Patrick Lindstrom, Sarah Jane Delany, and Brian Mac Namee. 2010. Handling concept drift in a text data stream constrained by high labeling cost. In *Proceedings of the Florida Artificial Intelligence Research Society Conference (FLAIRS'10)*.

[44] Viktor Losing, Barbara Hammer, and Heiko Wersing. 2016. KNN classifier with self adjusting memory for heterogeneous concept drift. In *Proceedings of the IEEE 16th International Conference on Data Mining (ICDM'16)*. IEEE, 291–300.

[45] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE Trans. Knowl. Data Eng.* 31, 12 (2018), 2346–2363.

[46] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[48] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: A comprehensive review. *ACM Comput. Surv.* 54, 3 (2021), 1–40.

[49] Leandro L. Minku and Xin Yao. 2011. DDD: A new ensemble approach for dealing with concept drift. *IEEE Trans. Knowl. Data Eng.* 24, 4 (2011), 619–633.

[50] Ricardo Nanculef, Ilias Flaounas, and Nello Cristianini. 2014. Efficient classification of multi-labeled text streams by clashing. *Expert Syst. Applic.* 41, 11 (2014), 5431–5450.

[51] Kyosuke Nishida, Takahide Hoshide, and Ko Fujimura. 2012. Improving tweet stream classification by detecting changes in word probability. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 971–980.

[52] Jakub Nowak, Ahmet Taspinar, and Rafał Scherer. 2017. LSTM recurrent neural networks for short text and sentiment classification. In *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*. Springer, 553–562.

[53] Nikunj C. Oza. 2005. Online bagging and boosting. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2340–2345.

[54] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1532–1543.

[55] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM Comput. Surv.* 51, 4 (2018), 1–36.

[56] Jason D. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. 2003. Tackling the poor assumptions of naive Bayes text classifiers. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*. 616–623.

[57] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 3856–3866.

[58] Doyen Sahoo, Quang Pham, Jing Lu, and Steven C. H. Hoi. 2018. Online deep learning: Learning deep neural networks on the fly. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2660–2666.

[59] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[60] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. 2009. Event detection and tracking in social streams. In *Proceedings of the 3rd International AAAI Conference on Weblogs and Social Media.*

[61] Martin Scholz and Ralf Klinkenberg. 2007. Boosting classifiers for drifting concepts. *Intell. Data Anal.* 11, 1 (2007), 3–28.

[62] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Comput. Surv.* 34, 1 (2002), 1–47.

[63] Tegjyot Singh Sethi and Mehmed Kantardzic. 2017. On the reliable detection of concept drift from streaming unlabeled data. *Expert Syst. Applic.* 82 (2017), 77–99.

[64] Tijmen Tieleman, Geoffrey Hinton, and others. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning* 4, 2 (2012), 26–31.

[65] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 226–235.

[66] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A large-scale dataset for news recommendation. In *(ACL'20)*. Retrieved from https://www.microsoft.com/en-us/research/publication/mind-a-large-scale-dataset-for-news-recommendation/

[67] Naoki Yoshinaga and Masaru Kitsuregawa. 2014. A self-adaptive classifier for efficient text-stream processing. In *Proceedings of the 25th International Conference on Computational Linguistics*. 1091–1102.

[68] Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710* (2015).

[69] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 649–657.

[70] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. 2012. Online incremental feature learning with denoising autoencoders. In *Artificial Intelligence and Statistics*. PMLR, 1453–1461.

Received 24 September 2022; revised 24 July 2023; accepted 15 December 2023