EVOLVING TEXT STREAM CLASSIFICATION WITH A NOVEL NEURAL ENSEMBLE ARCHITECTURE

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER ENGINEERING

By

Pouya Ghahramanian

January 2022

Evolving Text Stream Classification with a Novel Neural Ensemble Architecture
By Pouya Ghahramanian
January 2022

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Fazlı Can(Advisor)

Uğur Doğrusöz

İsmail Sengör Altıngövde

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

EVOLVING TEXT STREAM CLASSIFICATION WITH A NOVEL NEURAL ENSEMBLE ARCHITECTURE

Pouya Ghahramanian
M.S. in Computer Engineering
Advisor: Fazlı Can
January 2022

We study on-the-fly classification of evolving text streams in which the relation between the input data 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. We introduce Adaptive Neural Ensemble Network (*AdaNEN*), a novel ensemble-based neural approach, capable of handling concept drift in text streams. With our novel architecture, we address some of the problems neural models face when exploited for online adaptive learning environments. The problem of evolving text stream classification is relatively unexplored and most existing studies address concept drift detection and handling in numerical streams. 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 inducing 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 eight datasets. Our experimental results show that our proposed method, AdaNEN, consistently outperforms the existing approaches in terms of predictive performance with conservative efficiency.

Keywords: Text stream classification, Concept drift, Ensemble learning, Neural networks.

ÖZET

YENİ BİR SİNİR TOPLULUĞU MİMARİSİ İLE GELİŞEN METİN AKIŞI SINIFLANDIRMASI

Pouya Ghahramanian
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Fazlı Can
Ocak 2022

Kavram kayması, girdi verileri ve hedef etiketleri arasındaki ilişkinin zaman içinde değişmesi olarak tanımlanabilir. Bu çalışmada, kavram kayması içeren metin akışlarının anlık sınıflandırılması incelenmektedir. Kavram kayması, zamanla değişen veri varyasyonlarına bağlı olarak modellerin yanlış tahminlerinin artmasına ve performanslarının düşmesine neden olur. Bunu engellemek için kavram kaymasına adapte olabilen modeller geliştirmek gereklidir. Sunduğumuz Uyarlanabilir Sinir Topluluğu Ağı'nı (AdaNEN) modeli, metin akışlarında kavram kaymasına uyum sağlayabilen, topluluk tabanlı yeni bir sinirsel yaklaşım mimarisidir. Özgün AdaNEN mimarisi, çevrimiçi uyarlanabilir öğrenme ortamlarının yarattığı çoğu soruna çözüm üretir. Gelişen metin akışı sınıflandırması problemi, görece az araştırılmıştır. Mevcut çalışmaların çoğu, sayısal akışlarda kavram kayması algılama ve işlemeye yöneliktir. Bunun nedeninin açık ve büyük ölçekli deneysel verilerin eksikliği olduğunu düşünüyoruz. Bu doğrultuda, mevcut literatürdeki bir yaklaşıma dayalı, gerçek dünya metin veri kümelerine çeşitli kavram kaymaları ekleyerek gelişen metin akışları oluşturmaya yönelik, mevcut yeni bir yöntem öneriyoruz. AdaNEN'i teknolojiyi temsil eden 12 referans model ile sekiz veri seti üzerinden karşılaştırılarak yaklaşımımızın kapsamlı bir değerlendirmesini sunuyoruz. Deneysel sonuçlarımız, önerilen yöntemimiz AdaNEN'in, koruyucu verimlilikle tahmine dayalı mevcut yaklaşımlardan tutarlı bir şekilde daha iyi performans gösterdiğini göstermektedir.

Anahtar sözcükler: Metin akışı, Veri akışı, Metin sınıflandırması, Kavram kayması, Topluluk öğrenimi, Sinir ağları.

Acknowledgement

I would like to thank my supervisor Prof. Dr. Fazlı Can for his invaluable support during the course of my Master's degree. I am extremely grateful for our friendly chats in our meetings and his personal support in my academic endeavours. It was really a great chance and pleasure for me to study under his supervision.

I am also thankful to the other jury members for their time and valuable suggestions on my thesis: Prof. Dr. Uğur Doğrusöz and Assoc. Prof. Dr. İsmail Sengör Altıngövde. I would also like to thank faculty and staff of the department for all their kindness and considerate guidance. My gratitude extends to the Computer Engineering Department and TÜBİTAK for the funding opportunity to undertake my studies at Bilkent University.

I am speechless with gratitude to my cousin, Hamed Bonab, for his invaluable guidance and help during the course of this study. I would also like to thank Alper Can for his valuable comments and contributions on this work.

Last but not least, I express my deepest gratitude to my parents and my beloved brother, Mobin, for their tremendous support, understanding and encouragement throughout my life. Without their support, it would be impossible for me to complete my study. Also, thank you to all my friends for making my life more enjoyable with their friendship and support through good times and hard times.

Contents

List of Figures

List of Tables

Chapter 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 [10], lead existing approaches towards more accurate predictions for a variety of text processing tasks [15, 23, 51]. 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 and 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* [31]. One of the biggest challenges is to develop algorithms that adapt themselves with each drift. Existing literature focuses mainly on numerical data such as traffic management, click streams, and the stock market [14, 8, 5, 7]. Gama et al. [12] characterize the adaptive online learning and provide a survey on different types of concept drift: (i) Sudden/Abrupt, (ii) Incremental, (iii) Gradual, and (iv) Reoccuring. Ensembles of classifiers are commonly used for such settings [6, 35, 45, 13] with effective base models such as the Hoeffding Tree (HT) [16].

We refer to text stream with concept drift as *evolving text stream*. We study on-the-fly classification of evolving text streams. For example, detecting spam

emails from incoming emails, in which spam and non-spam labels change over the time, demonstrates certain types of challenges with our problem setting. Due to the volume as well as 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.

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, 29].

In this study, we propose an ensemble-based neural network approach capable of handling concept drift in text streams, named *AdaNEN*–see Chapter 4. The usage of neural models for online environments are 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) [28] and the AG News [52] datasets —see Chapter 5. Our experimental results show that our proposed method outperforms the existing approaches in terms of predictive performance with a conservative efficiency—see Chapter 6. We provide some further analyses on understanding the proposed model behavior as the data flows and present an ablation study showing the importance of the main components of AdaNEN—see Chapter 7. We give a a brief overview of offline and online text classification in Chapter 2.

Our contributions can be summarized as follows: We

- Introduce an ensemble-based neural approach for evolving text stream classification,
- Propose a method to construct large-scale evolving text streams and construct four text streams with concept drift using the proposed method,
- Provide an extensive evaluation of our proposed approach using 12 state-of-the-art baselines and eight datasets.

Chapter 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 category given a document [40]. In the last decade, word embedding methods have experienced a drastic change and novel embedding methods are introduced for representing words as numerical vectors based on their semantics. Different word embedding methods have been proposed in recent years [33].

Various neural and deep learning approaches have been proposed for classifying text documents. Zhang and LeCun [51] 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. [37] investigated performance of LSTM and two of its variants, namely Bidirectional LSTM and GRU in short text classification tasks. More recently, introduction of pre-trained deep neural language models, like BERT [10], shifted almost every downstream task toward such models and huge gains are reported for many NLP and IR tasks. Another recent attempt in text classification is using Capsule Network [15, 41] in text classification. Kim et al. [23] proposed a capsule-based model for text classification. As they indicate in their work, 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, such 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 [12, 22]. General frameworks of learning with drift detection and adaptation capabilities are provided by Lu et al. [31] and Aggarwal et al. [2]. ADWIN [3] 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 [16]. A Hoeffding Tree is capable of handling large data streams.

A more commonly used method for handling concept drift is to use an ensemble of classifiers. Kolter and Maloof [25] introduce one of the most famous ensemble methods designed specially for concept drift handling. Dynamic weighted majority (DWM) adds and drops classifiers to an ensemble based on their performance over time. Bonab and Can [7] introduce a geometrically optimum and online-weighted ensemble method (GOOWE) for evolving data stream classification by assigning optimum weights to the base classifiers. 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 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, event detection and tracking [1, 29, 44]. 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 [46, 34] with exciting improvements especially with the pre-trained language model-based classifiers. However, a few number of these studies have investigated evolving *text* stream environment, where the text data come in forms of a stream [17, 1].

Lindstrom et al. [30] 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. [36] 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. Yoshinaga and Kitsuregawa [50] study the real-time efficient processing of disaster-related Twitter[1] streams using a self-adaptive classifier in which the classifier reuse parts of the past streams which are related to the ongoing stream.

For other learning paradigms on text streams, Katakis et al. [19] introduce Conceptual Clustering and Prediction (CCP), using clustering to identify concepts, and train cluster-specific classifiers in an ensemble. Kumar et al. [27] proposed OSDM, an online model for short text stream clustering by embedding word-occurrence semantic information in the clustering task. Cortes et al. [9] investigated the adaptive learning of neural network structure and its weights. Zhou et al. [53] introduced an online algorithm based on autoencoders for determining the optimal model complexity by adding and merging features.

[1] https://twitter.com/

Chapter 3

Problem Definition

In this chapter, we explore the problem of evolving text stream classification. Firs, we define text stream classification; then, we focus on concept drift adaptation in text streams.

3.1 Text Stream Classification

In a text stream environment, data is generated over time via a source like social media or a news portal. We refer to a text data instance at time t as d_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. As illustrated in Figure 3.1, we define the problem of text stream classification as follows: For an incoming data d_t, the model predicts the class label, i.e. y'_t, by generating probability values of belonging data instance to each P class labels, i.e. $s_t =< S_t^1, S_t^2, \cdots, S_t^P >$. Here, we only deal with the binary scenario[1], i.e. $P = 2$. We assume that, for every instance, the correct

[1]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.

label becomes available at time $t+1$; making the immediate evaluation procedure possible. Finally, using the true class of d_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 [31, 12]. As a result, every data instance is used both for testing and training. A sliding window of recent data instances is stored for any model for updating purposes.

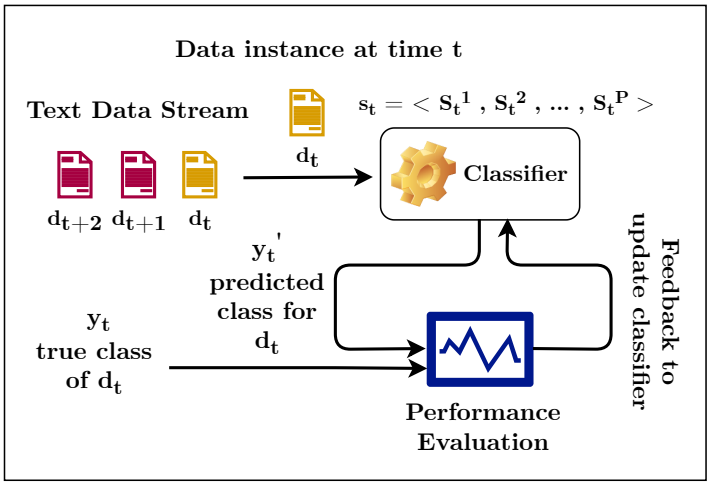


Figure 3.1: Text data stream classification problem.

3.2 Concept Drift in Text Streams

Concept drift in data streams refers to varying relationship between input data and output labels over time. Lu et al. [31] categorize concept drift into four types: (i) Gradual, (ii) Abrupt/Sudden, (iii) Incremental, and (iv) Reoccuring. In gradual concept drift, the input data output labels relationship varies over time, and there is no drift point. There is a gradual drift in the second data chunk in Figure 3.2. Abrupt concept drift occurs instantly in a definite point. In Figure 3.2, an abrupt drift happens in the first data chunk. In incremental concept drift, distribution of the data changes over time. In reoccurring concept drift, previously observed concepts reoccur over time.

In a text stream, concept drift may appear in any of the previously mentioned

forms. As an instance, while classifying a news document as interesting or not interesting for a user, there could be a possible drift where the user finds previously interesting topics not interesting. Such a scenario is our concern in this study.

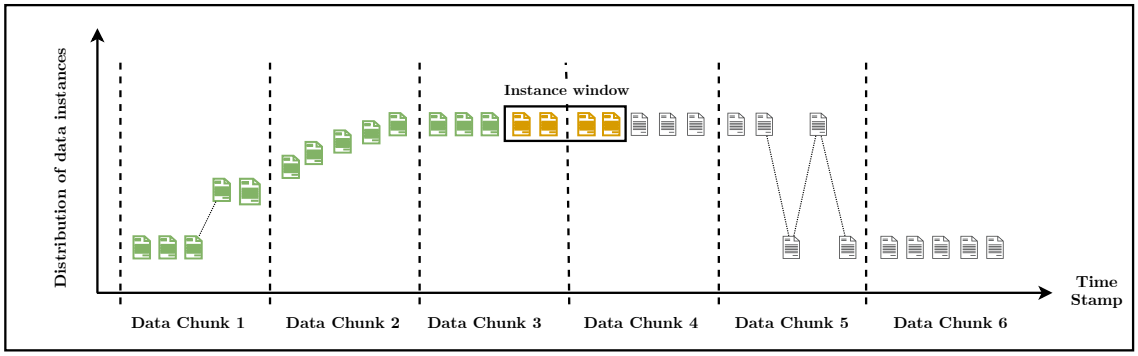


Figure 3.2: An illustration of various concept drift types within a text data stream.

Chapter 4

AdaNEN: Adaptive Neural Ensemble Network

4.1 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 few 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. *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 lower learning rate may cause optimization progress fail. Optimization methods, such as Adam [24] and RMSPbrop [47], decrease the learning rate in the optimization process to find the global minima. However, in evolving data stream classification we may need to increase learning rate to adapt to variation in data faster and handle the concept drift.

4.2 General Principles

Figure 4.1 presents the general schema of AdaNEN. 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 for combining predictions of the output layers and generating the final prediction. By using multiple output layers with different learning rates we let the network to focus on most recent data by increasing the ensemble weight of output layer with higher learning rates. Moreover, using a concatenation layer instead of 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.

4.3 Algorithm

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 j^{th} 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 5-6). Assume that X_k is the current data instance, and h_k^i is the i^{th} hidden layer result for that data instance. We construct the concatenation layer, Z_k, using every hidden layer in the feed-forward network. Let $Y`_k^j$ be the prediction of j^{th} output layer. Final prediction of the model is obtained by weighted majority voting (lines 7-11). The training phase of AdaNEN is described in lines 12-18. First, loss value for each of the output layers is calculated using cross-entropy loss function (lines 13-15). Then, assuming that $loss_j$ is the loss value for j^{th} output layer, total loss is calculated as weighted average of loss values (line 16). Finally, the obtained total loss is backpropagated through the network and the parameters are updated.

After calculating loss values and backpropagating the total loss, ensemble weights and loss values of output layers are updated (line 18). Ensemble weight for the j^{th} output layer is denoted by W_j. Ensemble weights are updated on the basis of loss values of the output layers. Note that β is the penalization parameter ($0 < \beta < 1$) and s is the regularization parameter that prevents assigning near-zero weights for output layers. Also, ensemble weights are divided by the sum of weights to ensure that they sum up to one. This update is similar to the Hedge ensemble method [42].

Algorithm 1 AdaNEN

Input: X : *Input Data Chunk*
Output: Y' : *Class Predictions*

1: **while** data stream has more instance **do**
2: X_k = Data instance at time step k
3: $Z_k = h_k^1 \oplus h_k^2 \oplus ... \oplus h_k^l$
4: Feed the output layers with Z_k
5: **for** j=1 ; j $\leq$ m **do**
6: $Y`_k^j$ = prediction of j^{th} output layer
7: **end for**
8: $Y`_k = \sum_{j=1}^{m} W_j \times Y`_k^j$
9: $Prediction_k$ = Max(Softmax($Y`_k$))
10: Y_k = True label for data instance at time step k
11: **for** j=1 ; j $\leq$ m **do**
12: $Loss_j$ = CrossEntropyLoss($Y`_k^j$, Y_k)
13: **end for**
14: $Loss_{total} = \sum_{j=1}^{m} W_j \times loss_j$
15: Backpropagate the total loss and update model parameters.
16: $W_j = \frac{\max(W_j \times \beta^{loss_j}, \frac{s}{m})}{\sum_{i=1}^{m} W_i}$
17: **end while**

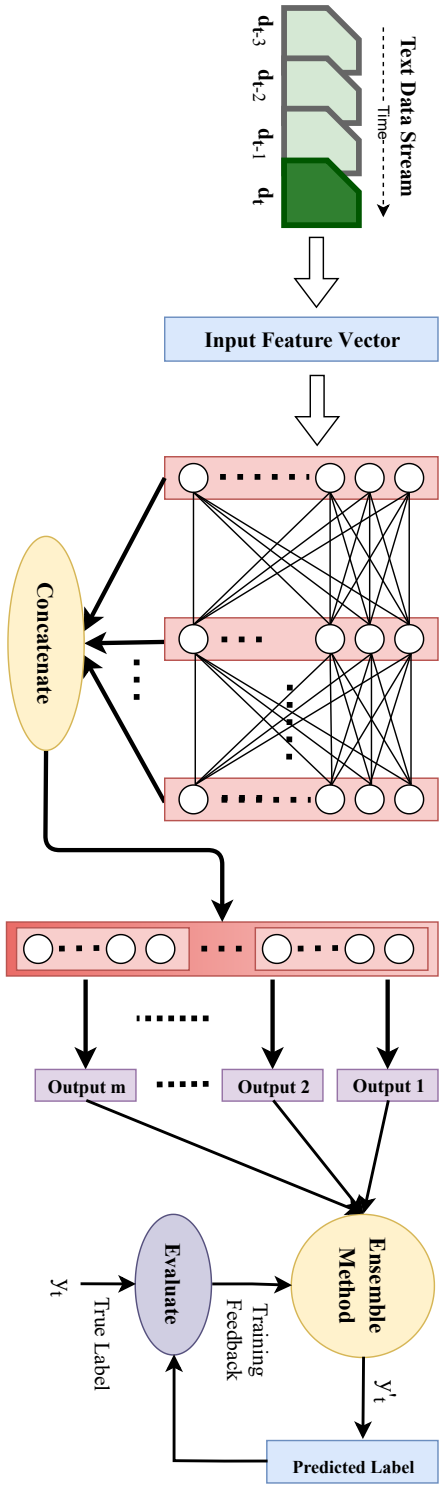


Figure 4.1: Overall schema of our Adaptive Neural Ensemble Network (AdaNEN) for evolving text stream classification.

Chapter 5

Evolving Text Stream Generation

Here, we describe our approach for constructing 20NG-Abrupt, 20NG-Gradual, AGNews-Abrupt and AGNews-Gradual text streams based on 20 News Groups (20NG) [28] and AG News datasets [52]. Our approach is an extended version of the method presented in [20].

20NG is a collection of 20,000 news articles with 20 news categories. We grouped the categories into 4 class labels; documents related to

1. **Cmp**, computers with *comp.** categories,
2. **Rec**, recreation with *rec.** categories,
3. **Sci**, science with *sci.** categories,
4. **Tlk**, talk with *talk.** categories.

AG News topic classification dataset is constructed from the original AG news dataset by choosing 4 largest classes (world, sports, business, sci/tech). The original dataset is a collection of 1 million news articles collected in more than one year from 2,000 news sources. 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. 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. 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. $Y = \{y_1, y_2, \cdots, y_n\}$, 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. [21] 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 time steps. This means $p_{t1}(X, Y) \neq p_{t2}(X, Y)$. Note that X refers to a set of data instances. Using the Bayes' theorem, we can state the joint probability as:

$$p(X, y_i) = p(y_i) \times p(X|y_i) = p(X) \times p(y_i|X) \tag{5.1}$$

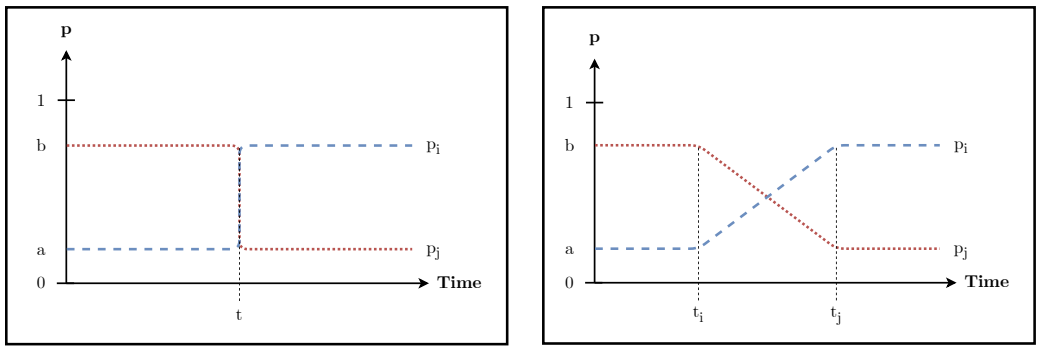
Based on this formulation three possible sources of concept drift can be defined: (i) Change in the prior probability of classes $p(y_i)$, which shows variation in target class distribution, (ii) Change in the posterior probability of target classes $p(y_i|X)$. It is also known as *real* concept drift, meaning that decision boundaries change over time, and (iii) Change in the class conditional probability $p(X|y_i)$, which is known as *virtual* concept drift. In this case, the input data distribution within a class changes. For example, the outbreak of a disease affects the distribution of medical news articles by publishing more medical news, which, in turn, causes a virtual concept drift.

On the basis of 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 5.1 illustrates the process. t is the moment that sudden concept drift happens. Similarly, $t1$, $t2$ 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:

$$\text{DI (Drift Intensity)} = \frac{b-a}{t_2 - t_1} \tag{5.2}$$



(a) Abrupt (b) Gradual

Figure 5.1: Adding abrupt and gradual drift to a stream.

We consider $(b - a)$ to be always 1, and $t_2 - t_1$ as the number of data instances during the drift. Drift intensity for the abrupt drift is ∞ and for a gradual drift is $0 < DI \leq 1$. Therefore, we can rewrite drift intensity as 1/(number of data instances during the drift). We use drifts with different intensity values. Table 5.1 and 5.2 show distribution of concept drift and intensity of drift in the 20NG and AG News datasets, respectively. DI values are randomly generated, and columns are showing the latest data instance—e.g, 3.5 showing the span of stream from the beginning up to 3,500 instances. Moreover, + shows user is interested in the corresponding news group and − means otherwise.

Table 5.1: Concept drift distribution for both 20NG-Abrupt and 20NG-Gradual text streams

	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
Cmp	+	+	-	-	-	-	-	+	+	+	-	-	-	+	+	+	+
Rec	-	-	+	-	-	-	+	+	-	-	+	+	+	-	+	+	-
Sci	+	-	-	+	-	+	+	-	+	-	+	-	+	+	-	+	+
Tlk	-	-	-	-	+	+	-	-	-	+	-	+	+	+	+	-	-
1/DI	-	50	100	25	200	50	25	150	100	75	250	125	75	50	100	25	50

Table 5.2: Concept drift distribution for both AGNews-Abrupt and AGNews-Gradual text streams

	10K	20K	30K	40K	50K	60K	70K	80K	90K	100K	110K	127.6K
World	+	-	+	-	-	+	-	+	-	+	-	+
Sports	+	-	-	+	+	-	+	-	-	+	-	+
Business	-	+	+	+	-	-	+	+	+	-	+	-
Sci/Tech	-	+	-	-	+	+	-	-	+	-	+	-
1/DI	-	5000	1000	500	100	500	2500	250	2000	1000	10000	500

Chapter 6

Experiments

Here, we describe our experimental setup; then, results are given and experimental analyses are provided with some discussion.

6.1 Experimental Setup

6.1.1 Datasets.

In our experiments, we use eight text stream classification datasets to evaluate the effectiveness and efficiency of our model. Four of these datasets are based on 20NG and AG News datasets with 11,500 and 127,600 data instances (see chapter 5), and the remaining four datasets are as follows:

1. **Usenet**[20]: a collection of 1,500 news articles, labeled as interesting or not interesting for a user. In the dataset, concept drift is defined as change in user personal interest, which occurs at each 300 data instances. Each data item has 99 attributes which are the words in the whole 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;

2. **Spam Filtering** [19, 18]: 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;

3. **Email List** [19]: 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 change in the user's interest in the categories. The vocabulary of the dataset consists of 913 words, and the features are the binary values representing the existence of a word in the document;

4. **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[1].

We use word2vec embeddings[2] [33] 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 BERT model on the Spam dataset.

6.1.2 Evaluation Method.

We used prequential test-then-train approach along with an instance window for training and evaluating classifiers, similar to the literature [7, 12, 31].

6.1.3 System Specification.

We perform our experiments with ensemble models on a machine equipped with an Intel Core i7-4702MQ @ 2.2 GHz processor and 8GB DDR3 RAM. For neural

[1]https://www.kaggle.com/benjaminawd/new-york-times-articles-comments-2020
[2]We used spaCy library: https://spacy.io/

models, except for BERT, we use an Nvidia GeForce GTX 960M GPU with 4GB DDR5 memory. As BERT requires significant amount of GPU memory, we used the *Google Colab*[3] platform to run our experiments with the BERT model.

6.1.4 Baselines.

To compare performance of AdaNEN, we use 8 ensemble and 4 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.

Ensemble Models.

1. Additive Expert Ensemble (AddExp) [26]: removes the weakest base classifier to overcome the concept drift in a stream;
2. Accuracy Weighted Ensemble (AWE) [48]: uses test accuracy result to determine the weight of each classifier;
3. Dynamic Weight Majority (DWM) [25]: adds and removes base classifiers and adjusts the weighting based on the performance on the test data;
4. Geometrically Optimum and Online-Weighted Ensemble (GOOWE) [7]: 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;
5. Hoeffding Adaptive Tree classifier (HAT) [4]: is a tree based model with ADWIN detection method for handling concept drift;
6. KNN-ADWIN [3]: KNN classifier and an ADWIN detector for handling the concept drift;
7. Learn++.NSE [11]: a modified version of the famous NSE algorithm, that adds the capability of handling concept drift in a stream environment;

[3] *https://colab.research.google.com*

8. Oza-Adwin [38]: an improved Online Bagging model with ADWIN change detector designed for non-stationary environment.

Neural Models.

1. SGD and Adam: Feedforward Neural Networks with Stochastic Gradient Descent (SGD) and Adam optimizers;

2. Hedge Backpropagation (HBP) [42]: an online, neural network based model for large scale data, capable of handling concept drfit;

3. Bidirectional Transformers for Language Understanding (BERT) [10]: a recent deep neural network model used for text classification. We tune and run the pretrained model on default parameters.

6.2 Experimental Results

Table 6.1 presents our results for prequential accuracy of the models. For example, the total prediction accuracy of AddExp on the entire Spam text stream is 91.30 percent. Table 6.2 shows runtime per data item results of models on data streams. For instance, the AddExp model takes 8.48 centiseconds to test-then-train each data item in the Spam stream. Our results suggest that AdaNEN outperforms the baseline models by achieving highest accuracy values in all datasets with a conservative runtime. Interestingly, simple feed-forward neural networks with Adam and SGD optimizers perform comparable to most of the other ensemble baselines, designed for concept drift handling. For instance, in Email text stream, SGD and Adam achieve the 2nd and 3rd ranks, respectively. We believe that good performance of Adam and SGD on text streams stems from 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 directly applicable for text streams. 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. However, in terms of running time 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 compress the size of the network and lowering the number of parameters [43]. 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. For the case of ensemble-based approaches, it can be due to the complexity of the updating every single model individually—i.e. maintenance cost. In fact with proper distribution implementation over multiple nodes, some of the running time cost might be reduced. For the case of the BERT model, the gap is an expected result because deeper models need considerable amount of time for training and testing even on GPU. Besides, our model is efficient enough for a data stream environment and its results are comparable with our baseline methods. For instance, on Email data stream AdaNEN is around 9 times slower than the fastest approach, while providing 85% increase in the prediction accuracy. We believe running our AdaNEN on a better hardware can compensate the differences with the fastest approach while preserving the high accuracy.

Table 6.1: Prequential Accuracy (%). For each column the best value is marked with the bold text

Model	Spam	Email	Usenet	20NG-Ab	20NG-Gr	AGNews-Ab	AGNews-Gr	NYT	Avg. Rank
AddExp [26]	91.30	55.65	64.76	64.10	64.87	52.36	51.42	64.52	7.25
AWE [48]	76.78	40.48	51.36	67.27	65.48	50.32	49.88	61.80	10.00
DWM [25]	90.61	56.53	69.86	72.10	70.84	69.44	67.49	53.51	5.00
GOOWE [7]	81.65	54.63	60.54	64.03	65.93	71.97	67.85	64.46	6.50
HAT [4]	90.77	55.66	64.49	63.86	64.67	72.07	69.84	62.87	6.00
KNN-Adwin [3]	94.40	57.28	56.80	68.01	67.27	65.74	64.12	61.36	5.38
Learn++.NSE [11]	83.63	46.19	56.33	64.87	64.46	50.03	49.87	60.05	10.50
Oza-Adwin [38]	92.41	54.01	57.76	67.97	68.39	62.58	61.07	61.33	6.63
SGD	83.59	78.91	68.37	62.24	70.04	54.42	53.16	49.97	7.63
HBP [42]	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 [10]	-	53.45	71.27	55.84	54.51	58.18	56.21	52.66	9.57
AdaNEN (Our model)	**97.31**	**85.85**	**79.39**	**80.23**	**72.00**	**74.37**	**72.38**	**64.66**	**1.00**

Table 6.2: Running Time Per Data Item (in centiseconds). For each column the best value is marked with the bold text

Model	Spam	Email	Usenet	20NG-Ab	20NG-Gr	AGNews-Ab	AGNews-Gr	NYT	Avg. Time (Avg. Rank)
AddExp [26]	8.48	10.43	2.08	6.43	7.69	5.88	5.65	2.52	6.15 (8.5)
AWE [48]	10.08	5.37	0.82	6.89	8.32	16.65	15.84	6.19	8.77 (8.88)
DWM [25]	7.70	6.57	1.39	6.37	7.74	6.16	5.93	2.61	5.56 (8.13)
GOOWE [7]	7.95	2.86	0.38	7.64	8.94	17.73	14.89	5.36	8.22 (8.25)
HAT [4]	2.43	6.00	0.61	1.69	2.23	2.19	2.23	1.89	2.41 (6)
KNN-Adwin [3]	16.51	8.13	1.21	11.16	12.87	23.08	22.78	7.82	12.95 (10.63)
Learn++.NSE [11]	**0.21**	**0.08**	**0.08**	0.31	**0.35**	3.71	1.64	2.95	1.17 (3)
Oza-Adwin [38]	18.27	23.24	2.50	19.23	21.58	43.99	40.63	15.73	23.15 (12.5)
SGD	0.23	0.72	0.25	**0.28**	0.37	**0.83**	0.92	**0.59**	**0.52 (1.75)**
HBP [42]	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 [10]	-	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 (3.5)

6.3 Prequential Evaluation

Figures (6.1 - 6.8) show the prequential evaluation of the top five methods for text streams studied with the highest average rank—accuracy values are plotted over time for each dataset. As it can be seen, in concept drifting points, which are indicated by sudden drops in accuracy, our model is able to adapt to the changes quickly, compared to other baseline models. For instance, in the Email data stream (Figure 6.1), our model returns to its previous performance faster than other baselines in concept drift points. In fact, we have similar observations for all datasets. Comparing out 20NG text streams prequntial analyses in Figures 6.5 and 6.6 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. The total accuracy values from Table 6.1 reported this. While none of the other strong baselines are always performing better, we notice that DWM from ensemble baselines and Adam from neural baselines are performing quite well compared to others.

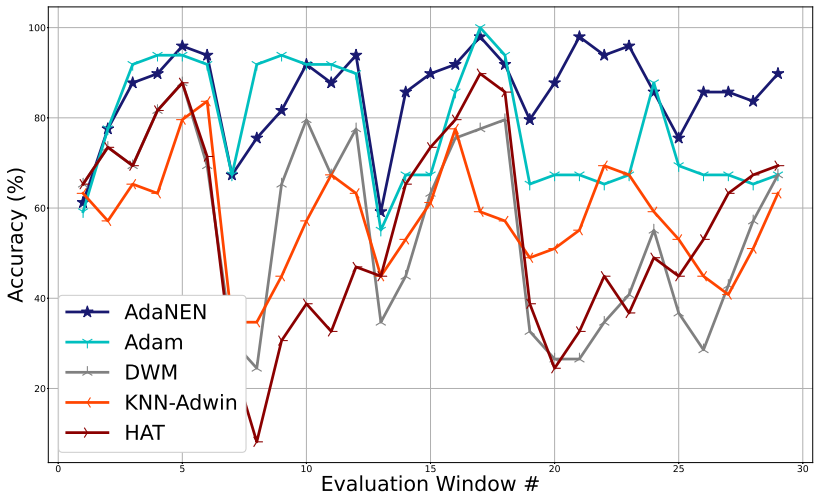


Figure 6.1: Email.

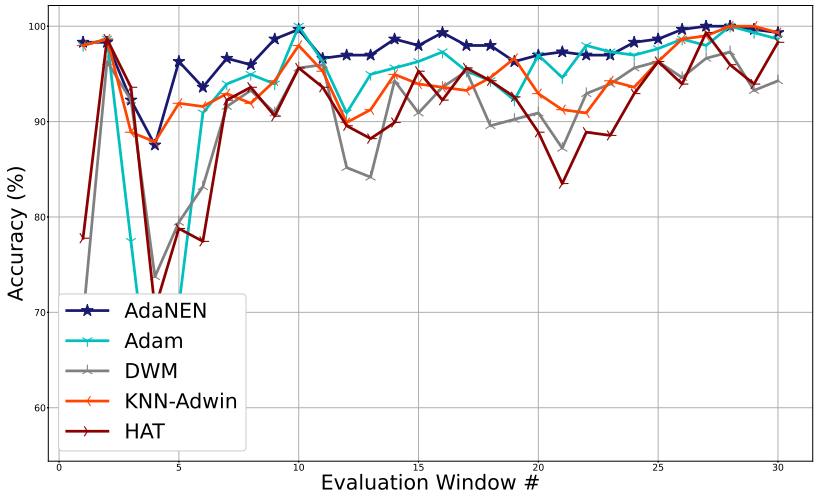


Figure 6.2: Spam.

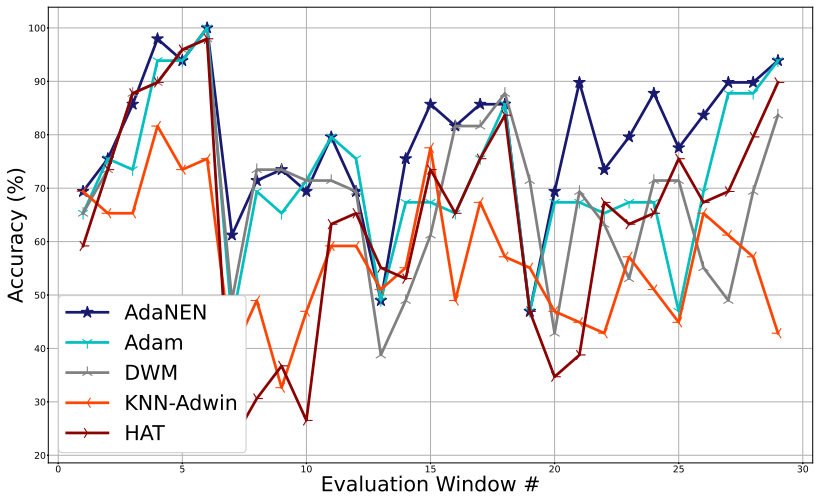


Figure 6.3: Usenet.

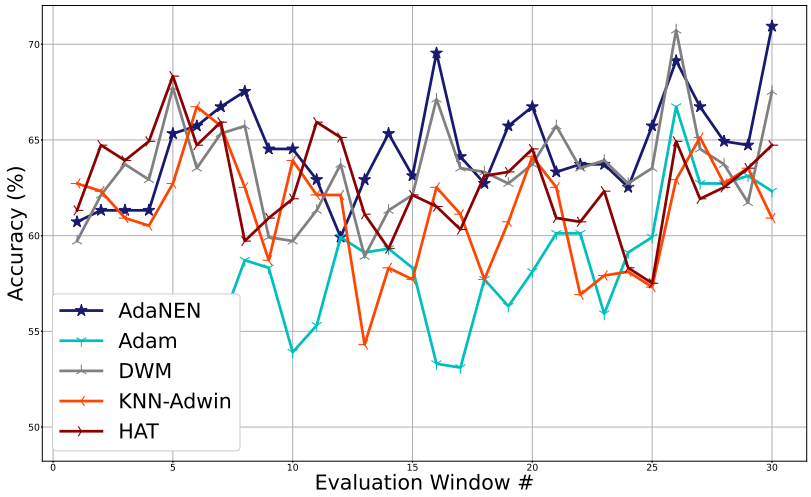


Figure 6.4: NYT.

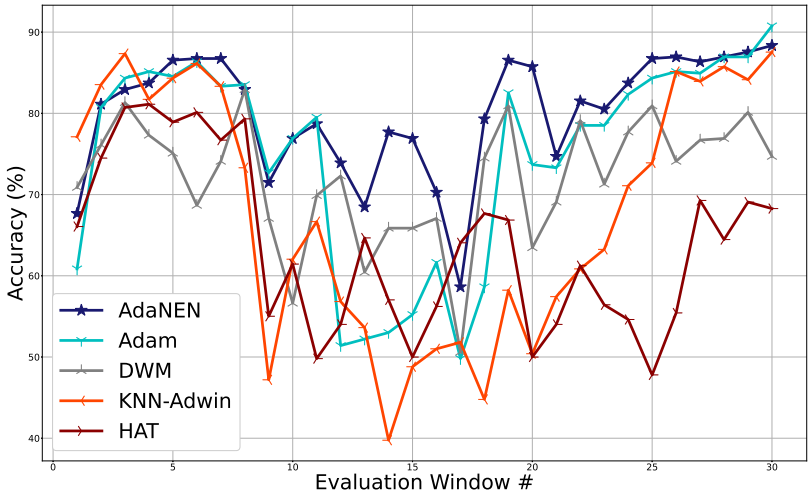


Figure 6.5: 20NG-Abrupt.

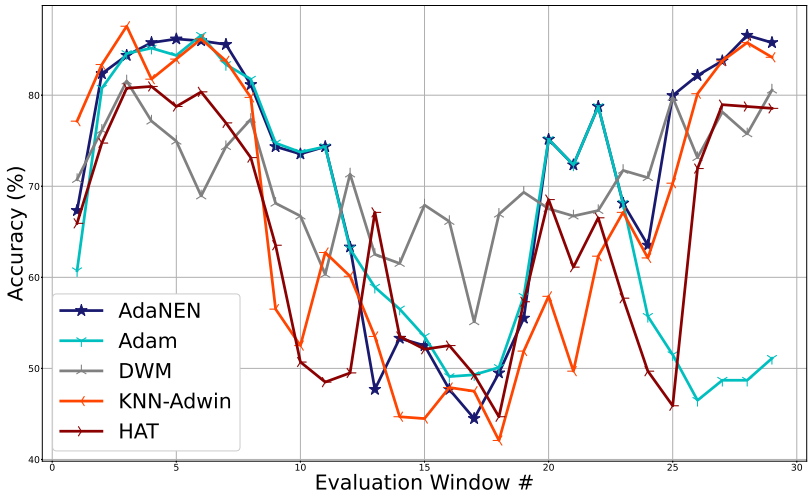


Figure 6.6: 20NG-Gradual.

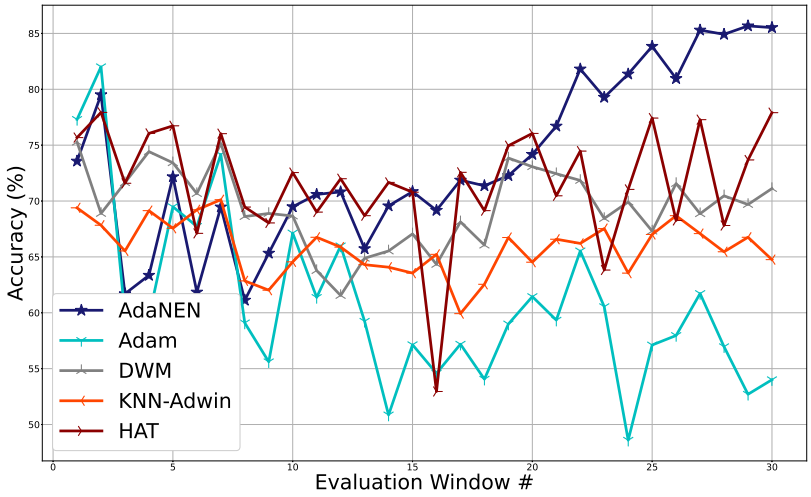


Figure 6.7: AGNews-Abrupt.

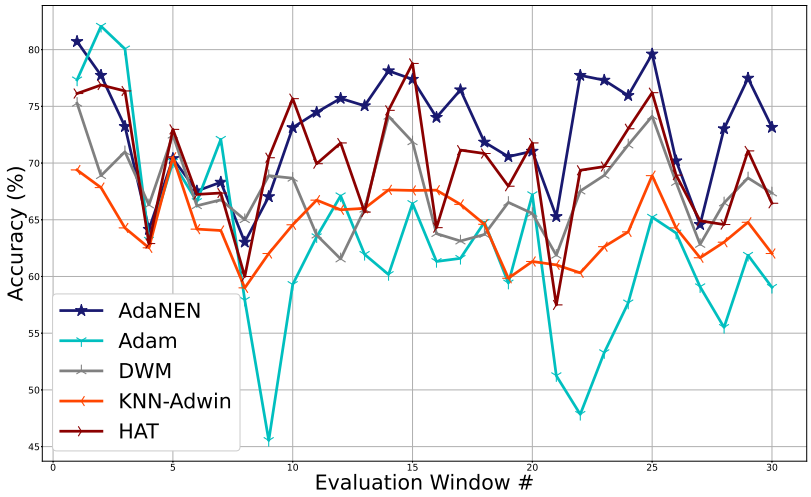


Figure 6.8: AGNews-Gradual.

Chapter 7

Further Analyses and Discussion

In this chapter, we further investigate differences in the performance of our model and baselines in terms of prequential accuracy, concept drift handling and efficiency/effectiveness aspects.

7.1 Ablation Analysis

In order 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.

1. *Complete*: the complete version of AdaNEN;
2. *AdaNEN-lr*: output layers with identical learning rate (1e-2) are used, to assess the effect of using output layers with different learning rates;
3. *AdaNEN-ens*: ensemble method is disabled and majority voting is replaced, to assess the effect of ensemble method.

Figure 7.1 shows our ablation study on the Email dataset. We only report Email dataset 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.

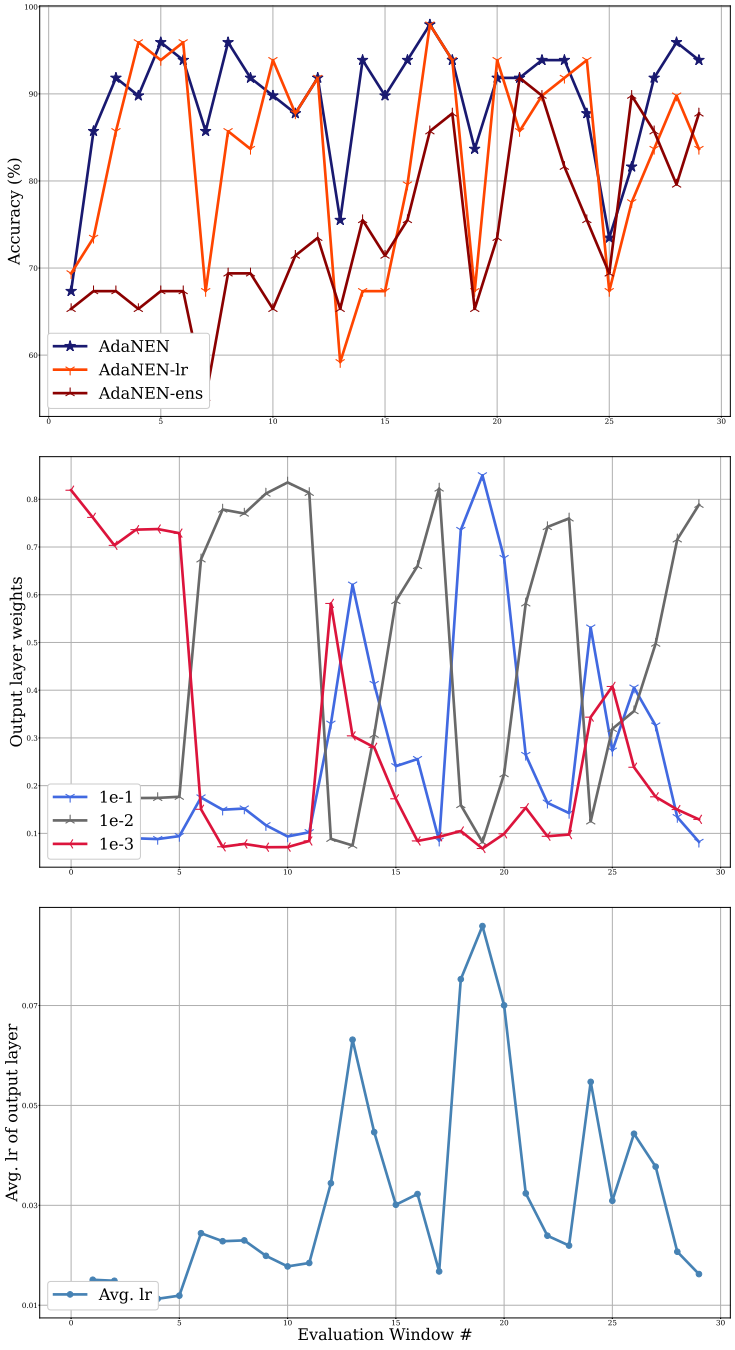


Figure 7.1: Ablation study of AdaNEN's learning rates and ensemble architecture on Email dataset.

7.2 Ensemble Weight Assignment Analysis

In order to show how AdaNEN adapts to the changes in a stream by changing weights for output layers with different learning rates, we track the ensemble weights for output layers. Figure 7.2 shows ensemble weights, average learning rate of output layer and accuracy plots of AdaNEN along with Adam, SGD, and HBP neural models, for the Spam data stream. Average learning rate the of output layer is calculated as:

$$lr_{avg} = \sum_{j=1}^{m} W_j \times lr_j \tag{7.1}$$

We use 3 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 2^{nd} evaluation window of the spam dataset, average learning rate of AdaNEN increases from 0.020 to 0.058 to adapt to the drift. It can be seen from the accuracy plot that AdaNEN maintains its performance at this point, for this, while accuracy of other neural models decreases dramatically. Another example is 11^{th} evaluation window.

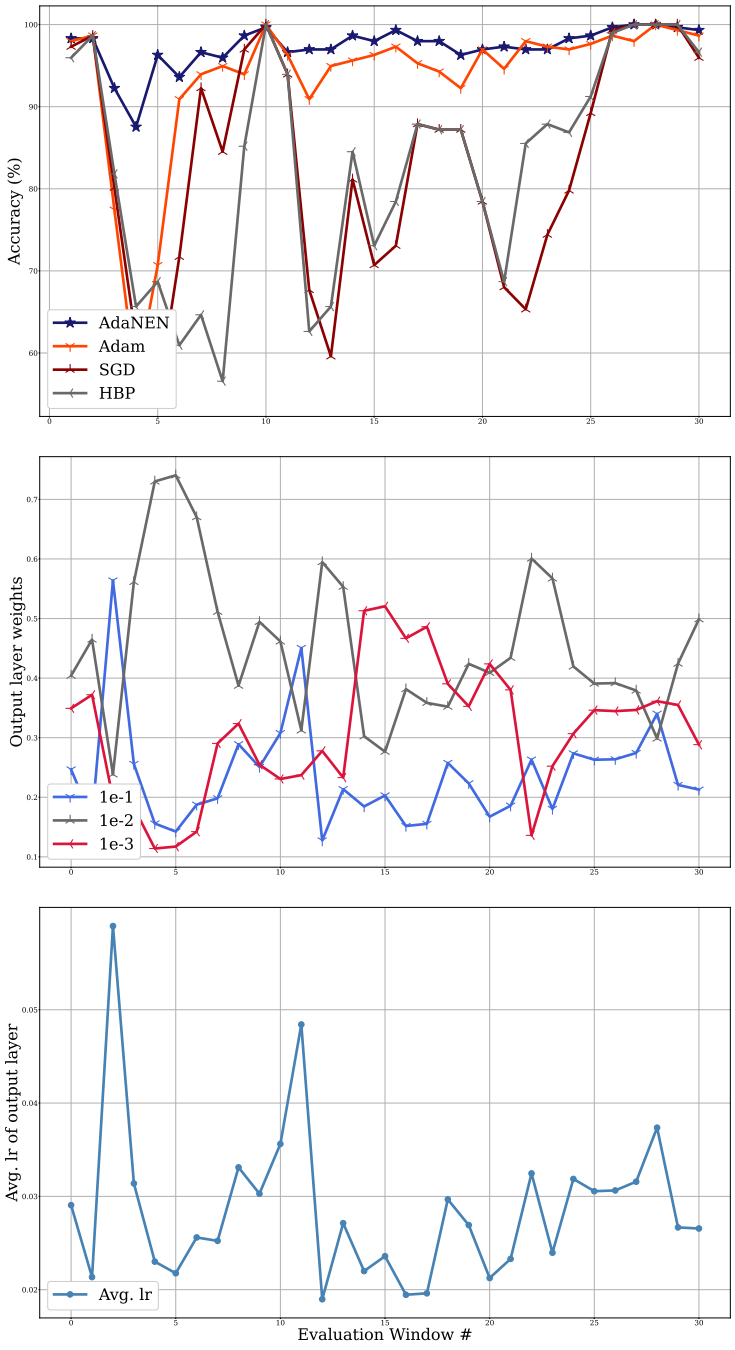


Figure 7.2: Ensemble weight assignment for Spam dataset.

7.3 Hyperparameter Analysis

In this section, we study the effects of hyperparameters in AdaNEN performance. The major hyperparameters of AdaNEN are:

1. l: Number of hidden layers,
2. m: Number of output layers,
3. lr_j: Learning rates of the output layers,
4. β: Ensemble weight update penalization parameter,
5. s: Regularization parameter.

We use grid search for tuning the hyperparameters in AdaNEN. Table 7.1 shows AdaNEN's performance on the 20NG-Ab dataset with different hyperparameter values. 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.

The number of hidden layers is an important factor in the performance of feed-forward neural networks. Using a lower of higher number of hidden layers reduces performance of the neural networks by limiting the model capacity or over-fitting the data. Using a concatenation layer enables AdaNEN to use adjusted number of hidden layers to fit 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

Table 7.1: Hyperparameter Analysis Result

Hyperparameter					Accuracy (%)
l	m	β	s	lr_j	
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**

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 usage of different learning rates in the output layer. We observed similar results for the other streams.

7.4 Effectiveness and Efficiency Analysis

In order to compare accuracy and runtime of AdaNEN and the baseline methods, average rank of the models' accuracy and runtime are plotted in figure 7.3. 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. Due to the properties of the data stream environment, we can claim that a model is a good fit, if it performs well in terms of 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 in comparison to BERT.

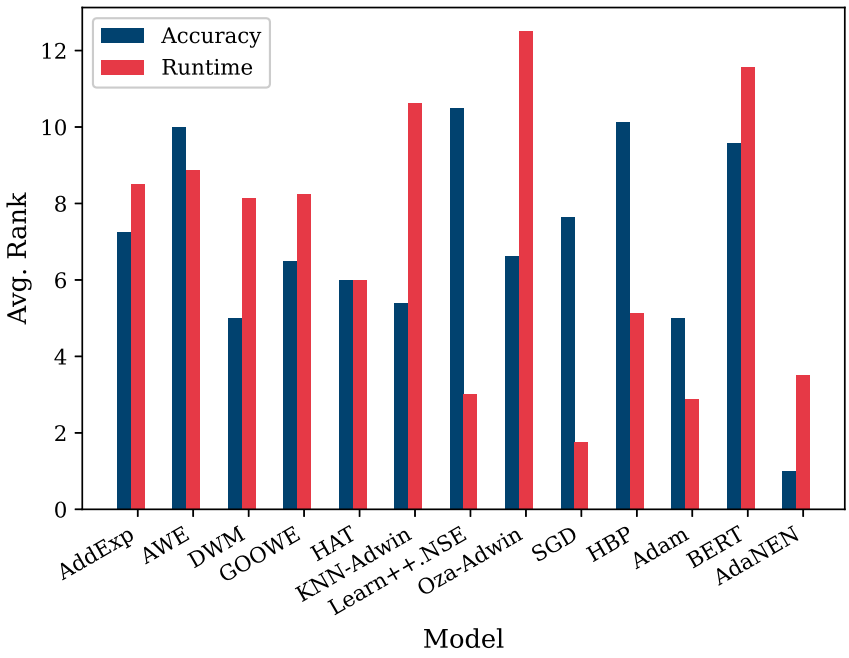


Figure 7.3: Accuracy and runtime avg. ranks.

Chapter 8

Conclusion and Future Work

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 inducing 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 to further extend our model design and data generation. More specifically, we are working on re-purposing the huge text datasets such as Amazon [32] and MIND [49]. Incorporating the adjacent field of sequence-aware recommendation [39] into our problem setup is another opportunity for further research.

Bibliography

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

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

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

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

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

[6] Hamed Bonab and Fazli Can. Less is more: A comprehensive framework for the number of components of ensemble classifiers. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2735–2745, 2019.

[7] Hamed R Bonab and Fazli Can. Goowe: Geometrically optimum and online-weighted ensemble classifier for evolving data streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(2):1–33, 2018.

[8] Dariusz Brzezinski and Jerzy Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94, 2013.

[9] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. In *International conference on machine learning*, pages 874–883. PMLR, 2017.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[11] Ryan Elwell and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011.

[12] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):1–37, 2014.

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

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

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

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

[17] AK Jumani, MH Mahar, FH Khoso, and MA Memon. Online text categorization system using support vector machine. *Sindh University Research Journal-SURJ (Science Series)*, 50(01):85–90, 2018.

[18] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Dynamic feature space and incremental feature selection for the classification of textual data streams. *Knowledge Discovery from Data Streams*, pages 107–116, 2006.

[19] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, 22(3):371–391, 2010.

[20] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis P Vlahavas. An ensemble of classifiers for coping with recurring contexts in data streams. In *ECAI*, pages 763–764, 2008.

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

[22] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada, 2004.

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

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

[25] J Zico Kolter and Marcus A Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8(Dec):2755–2790, 2007.

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

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

[28] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

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

[30] Patrick Lindstrom, Sarah Jane Delany, and Brian Mac Namee. Handling concept drift in a text data stream constrained by high labeling cost. In *FLAIRS Conference*, 2010.

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

[32] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52. ACM, 2015.

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

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

[35] Leandro L Minku and Xin Yao. Ddd: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633, 2011.

[36] Kyosuke Nishida, Takahide Hoshide, and Ko Fujimura. 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*, pages 971–980, 2012.

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

[38] Nikunj C Oza. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345. Ieee, 2005.

[39] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.

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

[41] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866, 2017.

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

[43] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[44] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. Event detection and tracking in social streams. In *Third International AAAI Conference on Weblogs and Social Media*, 2009.

[45] Martin Scholz and Ralf Klinkenberg. Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28, 2007.

[46] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

[47] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

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

[49] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606, 2020.

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

[51] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[52] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657, 2015.

[53] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online incremental feature learning with denoising autoencoders. In *Artificial intelligence and statistics*, pages 1453–1461. PMLR, 2012.