Fashion News Popularity Prediction

Zhe Zheng

Supervisor: Prof Philip Treleaven

Supervisor: Julija Bainiaksina

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Science

of

University College London.

Department of Computer Science

University College London

August 21, 2018

I, Zhe Zheng, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

This dissertation investigates the use of concept drift framework and deep learning techniques in fashion news popularity prediction. A new design of a deep neural work in text classification is also presented. The dissertation consists of two key experiments:

Experiment 1: Concept Drift Detection

The first experiment applies the concept drift detection algorithm to predict the popularity of the fashion news. More specifically, it investigates the performance between the concept drift detection framework and traditional incremental learning framework. Methodologies, such as drift detection method(DDM), Multinomial Bayes classifier, are proposed to classify the fashion news. The dataset for this experiment, collected by the web crawler, is the four-year fashion news from www.hypebeast.com. It contains 87,517 fashion news labeled by categories, keywords which including the related celebrities and brands, and the index of popularity.

Experiment 2: Multi-Input Deep Neural Network in Fashion News Popularity Prediction

The second experiment proposes a new structure of deep neural network in text classification. The new model combines the advantages of fastText model and bidirectional long-term short memory model and resorts the techniques commonly used on images, such as focal loss, and spatial dropout. The effectiveness of the proposed model in improving the prediction performance is evaluated by comparing with four state-of-the-art deep learning models, including fastText, Character-level CNN, bidirectional LSTM, and attention based bidirectional LSTM. The proposed model is trained on the fashion news from 1/1/2014 to 31/3/2018 and evaluated on the news from 1/4/2018 to 13/6/2018.

Contributions to Science

The major contribution of this dissertation is to offer a text-based deep learning model to predict the popularity of the fashion news. The prediction results can be used to 1) solve the cold start problem of the news providers' recommendation system; 2) Track the fashion trend quantitatively; 3) indicate the market attention to the fashion retail companies.

This thesis contributes to the existing literature in a number of ways. First, this research investigates the application of the concept drift detection algorithm in real world data, especially the high-dimensional and noisy text data. Most published concept drift detection algorithms are evaluated in the low-dimensional and noise-free dataset and few of them were applied to text data. Second, the research proposed a new structure of the deep learning model in text prediction problem which could achieve a competitive results. Third, the experiment proves that the techniques commonly used on image data also have a wide application on text data.

Acknowledgements

Acknowledge all the things!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents an overview of the dissertation. First, the motivation for this research is introduced along with the research topic. Second, the objectives of this research are discussed. Third, this chapter described the research methodology. This chapter ends by describing the structure of the whole dissertation.

1.1 Motivation

Predicting fashion news popularity from on-line fashion media is critically important from many aspects. First, there are many different news agencies on-line and they posted dozens of news every day. It is impossible to read all of them. Many website already have the recommendation system of popular news based on the page-views. As they use the page-views which have already occurred , it is not timely and has a cold start problem. The prediction of the popularity can highly avoid the cold start problem and provide the important features to the recommendation system. Second, it benefits the advertisement. A popular news means it will have more page-views in the future and namely has a higher value for advertisement. In this case, an effective fashion news popularity prediction enables the fashion medias to maximize revenue through differential pricing for access to contents with different popularity. Third, fashion trend is a very subjective concept and is hard to be described quantitatively. By predicting the popularity, it is possible to observe the change of fashion objectively and obtain a quantitative indicator which reflect the fashion hot-spot. Finally, some recent research also suggests that the very early indicators can be extracted from news to predict changes in various economic and commercial indicators. The popularity of the fashion news is, to a certain extent, an indicator of market attention, which can be helpful feature in stock price movement prediction.

Many research tried to track the popularity of the news and tweets based on the information propagation model, social dynamic model and the virus spreading model rather than news themselves. Wu and Shen [5] developed a model which can predict a news tweets popularity based on the number of its retweets soon after being published. Lerman and Hogg [6] used early user reactions to new content to create the stochastic models of user behavior on the website to predict the popularity.Rizos and el.[7] predicted the news story popularity indicators based on the user graphs. Unlike these work that paid little attention on the news itself, we particularly study the popularity of news from the text content.

Popularity prediction is a very timely task as we can obtain the ground truth of the popularity after a while and the prediction result is not needed anymore. It requires the model to not only accurate but timely. Some of the models aforementioned have the cold start problem. Because there is no information flow when the news is just released. However, there is not such problem for a context-based model. In addition, dozen of news are posted everyday, the model must be updated everyday. Most of the published model use SVM, Random forest. It is not easy to do the incremental learning with these algorithms. It means the model can only updated by training with th whole data set which is very inefficient. However, deep learning model can easily support the incremental training and update the model quickly with the new instance.

The first challenge is that fashion hot-spot is not stable. From a machine learning perspective, fashion trend may depend on some hidden context, not given explicitly in the form of predictive features. In other words, tracking fashion trend from fashion news is a concept drift problem. Thus, the conventional prediction framework may not work in such problem. It is necessary to create a system to detect the concept drift phenomenon and exam the performances between the concept-drift model and conventional machine learning model.

The second challenge is that fashion trend can change both gradually and suddenly [8]. Namely, the underlying data distribution may change gradually or suddenly with time. Unfortunately, none of published algorithm has a good performance in handling both change. Some algorithms may overreact to the noise, erroneously interpreting it as concept drift, while others may be highly robust to noise, adjusting to the changes too slowly [9]. In addition, most of the published algorithms were only tested with the low-dimensional concept drift problem, while in the most cases, text classification problem is high-dimensional.

In addition to the first two challenges aforementioned, the third challenge is that the data set is imbalanced. Most machine learning algorithms works best when the number of instances of each class are roughly equal. In this research, popular news only accounts for less than 10% of all the news. Imbalance data set may leads to accuracy paradox. In this case, where the accuracy measures tell the story that the model has excellent accuracy, but the accuracy is only reflecting the underlying class distribution. Thus, it is challenging to select a suitable model and metrics. It is worth trying a variety of techniques to balance the dataset.

Finally, there are many published text classification model based on deep learning right now. Each of them has very different structure and mechanism, thus has its own pros and cons. It is necessary to do multiple experiments to select the effective structure and layers. The new model can be developed based on the results of these experiments

The main motivation of this research is to improve the accuracy of the fashion news popularity prediction model to help fashion media improve their recommendation system, advertising business and provide a quantitative indicator for fashion trend.

1.2 Objective

Our research objective is to improve the accuracy of predicting the popularity of the fashion news by testing the published concept drift algorithm and propose a new deep learning model. The main hypothesis of this research states that:

> The fashion trend can be tracked by analyzing the popularity of the fashion news. Current published concept drift algorithm do not have obvious advantage compared with the conventional machine learning framework. By exam the state-of-the-art deep classification model, the new model can significantly increase the accuracy of the fashion news popularity prediction.

To validate this hypothesis, there are two main tasks for this research:

1. Build a concept drift detection framework.

The first task is to detect the concept drifts and compare the performances between the framework which can handling the concept drift problem and the classical machine learning framework. It can be done easily by comparing the metrics of prediction result of the test data. This research, therefore, gives the conclusion if the concept drift increase the performance of the prediction model.

Table 1.1: Popularity Classes

Popularity	HYPES
Hot	>20000
Medium	5000-20000
Cold	0-5000

2. Propose a system design for predicting the fashion news popularity

Based on the first experiment's result, the second task is to construct a model to predict the fashion new popularity. First, four state-of-the-art text classification models are implemented and their performances are compared. Then, based on the comparison, the new model structure is proposed. This research presents a multiple inputs and outputs model which can deal with the cold start problem and easily be updated.

1.3 Methodology

1.3.1 Data used in this research

The dataset for this dissertation, crawled from the famous fashion websites http://www.hypebeast.com and http://www.hypebae.com, contains 12-years fashion news labeled with keywords, categories, comments, and index of HYPE. The official definition of HYPE is

> HYPES is a real time popularity metric informing reader of what is trending

It contains 182,294 fashion news from 2005-4-20 to 2018-6-14. According to the experience, the website also divides the index of HYPES into three classes shown in Table 1.1.

Hypebeast.com is the most famous fashion and street culture website around world, attracting 9.4 million unique visitor per month and 44 million page views per month. Its social media account has 9.3 million followers. It can be said that HYPEBEAST is the most influential fashion and street culture media.

1.3.2 Model design

In this research, there are two models which need to be designed. The first one is the model that uses the concept drift framework. The second model is the deep learning model combines a variety of the state-of-the-art neural network layers.

In the part of the first model, the Drift Detection Method (DDM) is applied to detect the concept drift. In this method we assume that examples arrive one at a time. The framework could be easy extended to situations where data comes on batches of examples. We consider

the online learning framework. In this framework when an example becomes available, the decision model must take a decision (e.g. an action). Only after the decision has been taken the environment react providing feedback to the decision model (e.g. the class label of the example) [10].

The second model is the deep learning model with multiple inputs and outputs. There are three inputs layers. The first one receive the tokenized titles and embed them. The embedded titles are feed into the average pooling layer to extract the features. The tokenized articles are sent to the second input layer to be embedded. The two layer bidirectional GRU are used to deal with the sequence input and the extracted features are concatenated with the feature of title. The concatenated feature vector is connect to a dense layer and output the prediction result of the cold start fashion news. For a non cold start problem, there is another input to receive the comments from the readers. After feature engineering, the feature vector of the comments are then concatenated with the other two vectors and connected to the dense layers. The prediction result for the non cold start news is produced by these dense layers.

1.3.3 Model implementation

All the models are implemented in Python. The deep learning model is implemented by Tensorflow and Keras. The raw data is collected by the website crawler implemented by Scrapy library. In addition, MongoDB is used as the database to store the raw data.

1.4 Structure of the dissertation

Structure of this dissertation is as follows,

- Chapter 2 reviews background information on a number of key concepts in the area of Machine Learning, concept drift problem, text classification problem. It also presents the basic theory of deep learning and the state-of-art neural network structure in text classification. The literature related to the research is also briefly described in this chapter.

- Chapter 3 describes the state-of-the-art framework for concept drift detection and compare the performance of the concept drift framework and the conventional machine learning framework.

- Chapter 4 exams the performance of four published deep classification models. Based

on the experiment results, a new model is proposed. The model test also shows in this chapter. The investigation of the model is presented as well.

- Chapter 6 summarises this research and outlines possibilities for future work.

Chapter 2

Background

This chapter introduces the general context of text classification problem. I present the motivation for the fashion news popularity prediction. In addition, a brief introduction of the concept drift is given. Also, a detailed review of some latest concept drift detectors is presented. In addition, basic theory of deep learning model and its application in natural language is discussed. An introduction of some latest deep learning models in text classification is presented at last.

2.1 Text Classification and News Popularity Prediction

Predicting news popularity is not a new topic in the field of machine learning. However, most of the previous prediction method is based on the information propagation on online social network. Sakaki et al. considered each social network user as a sensor and used machine learning to do the prediction [11]. Wu and Shen use the topology reconstruction and inferences of network diffusion to predict the popularity of the twitter news [5]. Few of the them paid attention to the news themselves. Most of the classification model focusing on the text itself is classifying the news into its category. For example, Yahoo tried use machine learning to classify the sports news into different classes of sports [12].

Applying text classification model to news content does not make sense because even a professional editor can not know exactly if the news would be popular when he sees the news at the first time. However, fashion news is different to the conventional news. Marketing and advertising play an important role in fashion industry. The close connection between marketing and fashion determines brand, celebrities, and products contributes to the heat of the fashion news [13]. Simply speaking, predicting fashion news popularity is a task that extracting which brands, products, and celebrities attract most attention. Based

on this hypothesis, it works that predicting the fashion news popularity from the news itself rather than tracking the information flow of the news.

Currently, here are two main types of text classification methods. One is the conventional machine learning approaches which require detailed feature engineering to achieve good prediction results. With this type of method, the quality of the hard-crafted feature determines the final performance of the classifier. Bag-of-words or bag-of-ngrams and its TFIDF are normally used feature extractor. For the bag-of-words(ngrams), the counts of each word(ngram) is used as the feature. For the TFIDF, the counts is used as the term-frequency [14]. IDF is the inverse document frequency which is a measure of how much information the word provides. The IDF can be computed by the following equation

$$\text{IDF} = \log\frac{N}{n_t} \tag{2.1}$$

where N is the total number of documents in the corpus and n_t is the number of documents where the word(ngram) appears. Thus, the term frequency-inverse document frequency (TFIDF) is defined as

$$\text{TFIDF} = \text{TF} \times \text{IDF} \tag{2.2}$$

Another type of method uses deep learning technique, including deep learning neural network (DNN), convolutional neural network (CNN), and recurrent neural network (RNN). Different architecture of the models determines the pros and cons of the model. We will discuss deep learning model in detail in Section 2.3.

2.2 Concept Drift

2.2.1 Definition and Framework

Learning from sequence data is a topic that addresses to many machine learning filed, such as data mining, natural language process, and pattern recognition. The main difficulty of learning from time series data is that the target concept may change over time which is known as the concept drift problem [15]. For example, in fashion news popularity problem, the news about Under Amour might be popular in 2015 while it is unwelcome in 2018. In this case, a conventional framework of machine learning may not about to classify the instance from different year correctly.

According to the reason of the change, there are two type of change. The first one is the

real concept drift which occurs when a set of example has class labels at one time and has different labels at another time [16]. Another type of drift is the virtual concept drift which occurs when the target concepts remain the same but the data distribution changes [17]. However, in the real world problem, these two concept drifts often occur together.

As of today, many concept drift have been proposed. Most of the concept drift detectors are composed of two parts. One is a base learner, such as decision tree [18], support vector machine [19], which is used to classify the incoming data. Then, The prediction result is compared with the true label and compute the metrics. Another part is the drift detection method which indicate whether the concept drift occurs or not based on the classification result. If the concept drift occurs, according to different detector framework, the base learner will re-trained in a new subset of the instances. For each incoming instance, the framework repeat the process.

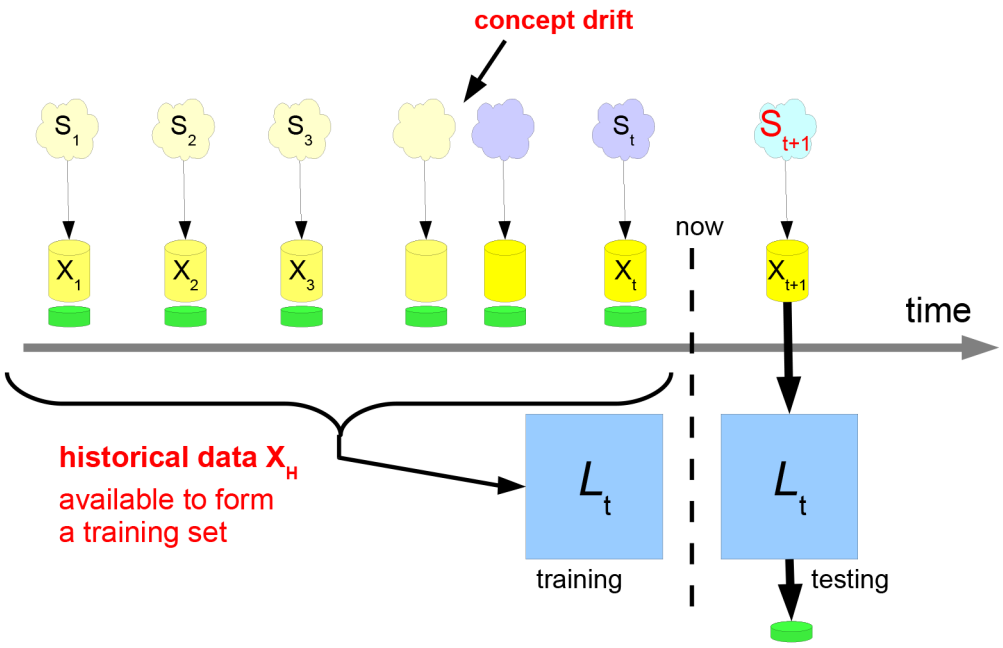
To decrease the training time, most of the concept drift detectors uses incremental learning framework, which show in Figure 2.1. Let $\mathbf{x}_t \in \mathbb{R}^p$ is the feature vector observed at time t and $y_t \in \mathbb{R}$ us the corresponding label. We assume the stream $\mathbf{X}^H = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t)$ as the historical data and the vector $\mathbf{x}_{t+1}$ as the test data. In incremental learning framework, we use all or a subset of the historical data $\mathbf{X}^H$ to train the learner $\mathscr{L}_t$ to predict the next instance $\mathbf{x}_{t+1}$. At next step, the label of $\mathbf{x}_{t+1}$ is available. Thus, we are able to repeat this process. The key function of the concept drift detector is to select a suitable subset of the historical data $\mathbf{X}^H$ which can represent the distribution of the data.

2.2.2 Concept Drift Detectors

According to the number of citations, there are four popular concept drift detectors, DDM [10], EDDM [**?**], PHT [20], ADWIN [21], Paired Learners [22], ECDD [23], DoF [24], and STEPD [25]. The most popular and effective method is the Drift Detection Method (DDM). Its mechanism is simple and easy to understand. DDM uses a base learning (decision tree) to classify the incoming instance and the the online error-rate is computed with the prediction result. In the probably approximately correct learning model [26], it is assumed that the error rate of the base learner will decrease when the number of training instance increases if the distribution of the incoming instances is stationary. Thus, if the model detects a significant increase of the online error-rate, it suggests the concept drift occurs.

Suppose every instance in the form of $(\mathbf{x}_i, y_i)$. The base learner gives the prediction $\hat{y}_i$

Figure 2.1: The incremental learning framework at time t [1]



that can be used to compute the error-rate p_i and the standard deviation s_i given by

$$s_i = \sqrt{\frac{p_i(1-p_i)}{i}} \tag{2.3}$$

If it is a binary classification problem, for the set of incoming instance, the error is a random variable from Bernoulli trails and the binomial distribution gives the probability for the random variable representing the number of errors in a sample of n instances. If the number of instance is large, the Binomial distribution is closely approximated by a Normal distribution with same mean and variance. We assume the confidence level is $alpha$. Then, the confidence interval for error p is about $p_i \pm \alpha * s_i$. The framework records the minimum of p_i and s_i, obtaining p_min and s_min, during the training process. The warning level is defined as $p_{min} + 2 * s_{min}$ by setting α to 95%. The drift level is $p_{min} + 3 * s_{min}$ with 99% confidence level. Suppose a data stream where the error reaching the warning level at example w and the drift level at example d. The base learner is re-trained using only the instances from w till d. Based on previous experiments, DDM is proved to be the best method in datasets suffering from gradual concept drifts [15].

The Early Drift Detection Method (EDDM) [**?**] is similar to the Drift Detection Method (DDM). It uses the distance-error-rate rather than the online error-rate to identify the occur-

rence of the concept drift. While there is no concept drift, the predictions and the distance between two errors will be increased during the training process. For every incoming instance, p'_i, the avergae distance between two errors, and its standard deviation s'_i are computed. The maximums of p'_i and s'_i are recorded, obtaing p'_{max} and s'_{max}. Similar to DDM, there are also two thresholds. The threshold is defined as

$$\frac{p'_i + 2 * s'_i}{p'_{max} + 2 * s'_{max}} < \alpha \tag{2.4}$$

where α is the confidence level. Beyond the confidence level of the warning level, the instances are stored in advance for the re-training process. Beyond the confidence level of the drift level, the base learning is re-trained using the stored examples. The EDDM is designed for dealing with the gradual concept drift. It has better performance than DDM with noise free data but worse with real world data [15].

The Adaptive Windowing Method (ADWIN) is another approach for dealing with concept drift. It uses a sliding window whose size changes during the training process. The window size is recomputed online according to the rate of change observed from the data stream. The window size grows when there is no apparent concept change and decreases when the change is detected.

We assume ADWIN keeps a sliding window W with size of n. $\hat{\mu}_W$ is the average of the instance in W while μ_W is the average of μ_t for $t \in W$. When two large enough sub-windows W_0 and W_1 exhibit distinct enough averages, the old portion window is drop. The value of ε_{cut} for a partition $W_0 \cdot W_1$ of W is computed as follows

$$\varepsilon_{cut} = \sqrt{\frac{1}{2m} \cdot \ln \frac{4}{\varepsilon'}} \tag{2.5}$$

where ε is the confidence level, $m = \frac{1}{1/n_0 + 1/n_1}$, n_0 and n_1 are the length of the corresponding sub-windows. For each time-step, the new incoming instance is added to the head of W and the elements from the tail is dropped one by one until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \varepsilon_c ut$ holds. ADWIN is memory-efficient and it is the fastest one of these four popular algorithm. In some real world problem, it is even faster than the base learner [15].

The Paired Learner is a slightly different framework which uses two learners, a stable learning predicting based on all instances and a reactive learned predicts based on some recent instances with a fixed window size. The PL method has a circular list of bits with

the same length of the window which stores 1 if the example is misclassified by the stable learner but classified by the reactive learner correctly, and 0 otherwise. While the number of 1 is great than the threshold θ, it indicates the concept drift occurs. Then, the stable learner is substituted by the reactive one and the circular list is set to 0.

Though the paired learner is a very creative method compared with other approaches. It is the worst methods dealing with the abrupt concept drift. As it uses two learner, PL is the slowest among these methods.

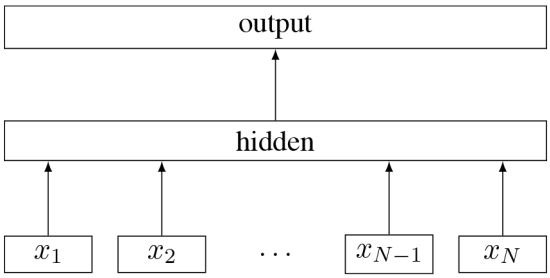
2.3 Deep Learning Models in Text Classification

Deep learning refers to the machine learning techniques for learning and utilizing neural networks with multiple layers, such as deep neural network (DNN), convolutional neural network (CNN), recurrent neural network (RNN). Deep learning has proven effective in reshaping the processing of speech recognition, and machine vision in a revolutionary way. In speech, image, deep learning effectively addresses the semantic gap problem by learning high-level concepts from raw data in a direct manner. Compared with these tasks, natural language processing is less straightforward as it is doubtful whether the neural representations learned from textual data can provide equally direct insight onto natural language.

However, recently, deep learning has been successfully applied to the NLP problem and some significant achievements have been made. There are five main tasks in NLP, including translation, matching, classification, structured prediction and the sequential decision process [27]. The deep learning technique has proven to outperformed in the first four tasks. Because in traditional machine learning, features are designed by humans and feature engineering is bottleneck, require much human experience. Deep learning break away these difficulties by using deep model structure. End-to-end training and representation learning really e it powerful machinery for NLP.

Text classification is an import task in NLP. It has a variety of applications, including web search, ranking, recommendation system, document classification. Recently, many deep learning models in text classification problem has been proposed [28][2]. each of them achieves good performance and has its own pros and cons. In this section, four latest text classification models using deep learning techniques are introduced. Meanwhile, the fundamental deep learning technologies are discussed.

Figure 2.2: Model structure of fastText for a sentence with N n-gram features.



2.3.1 fastText

fastText, as its name would suggest, is a fast text classification model developed by Facebook AI Research. While other deep learning models tend to be slow to train and test which significantly limits their usage on large scale datasets. fastText can be trained on more than one billion words in less than ten minutes using a standard multicore CPU, and classify half a million sentences among 312k classes in less than a minutes.

The structure of fastText is shown in Figure 2.2. Different with other deep learning models which just use all words as the input features, fastText uses a bag of n-grams as additional features to capture the partial information about the word order. For example, for a sentence like "I like eating apples", the bag of 2-grams includes "I like", "like eating", "eating apples". Therefore, the input features are ["I", "like", "eating", "apples", "I like", "like eating", "eating apples"]. With this trick, the classifier can easily distinguish the difference between "Jerry loves Tom" and "Tom loves Jerry".

Then every word and n-gram is transformed into a corresponding number and feed into the embedding layer. Simply speaking, embedding is just sparse matrix multiplication. For each word or n-gram, it corresponds to an unique number. The number can be represented as a one-hot coding vector. We assume the word of "apples" corresponds to number of 2 and there are totally 6 words and n-grams in our corpus. The one-hot vector $\mathbf{v}$ for "apples" is $[0,1,0,0,0,0]$. There is only one element can be 1 and others are 0. Each position is a word and the length of v is the corpus size. As the number of "apples" is 2, the second element of the vector is 1. To embed a word or n-gram, we simply do

$$\mathbf{f} = \mathbf{W}\mathbf{v} \tag{2.6}$$

where $\mathbf{f}$ is the embedded word vector, $\mathbf{W}$ is a $D \times$ corpus size matrix and D is the dimension

of the embedded vector for each word. After the embedding layer, each word obtains a unique embedded word vector and fastText averages these vectors. The averaged vector is called text representation which stores the features of the whole sentences. Finally, the text representation is in turn fed to a linear classifier. The softmax function is used to compute the probabilities of each class. For text representation in form of $\mathbf{e}$, the probability of class j is shows as follows

$$p(y = j|\mathbf{f}) = \frac{e^{\mathbf{f}^\intercal \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{f}^\intercal \mathbf{w}_k}} \tag{2.7}$$

where $\mathbf{w}$ is a weighting vector, K is the number of classes.

fastText uses the negative log-likelihood as the loss function. For a set of N instances, the loss function is defined as

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(p_{y_n}) \tag{2.8}$$

where y_n is the ground truth of the example n. The model is trained by using stochastic gradient descent. To increasing the convergence speed, the linear learning rate decay is applied.

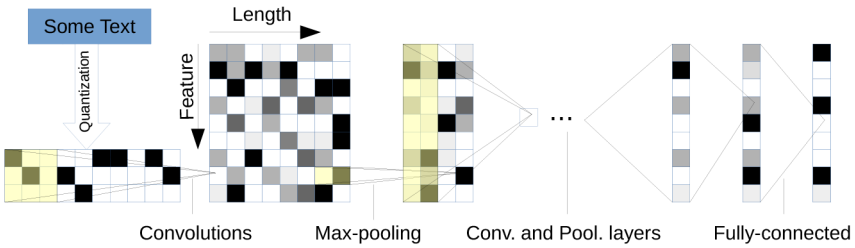
For a large number of classes, fastText uses hierarchical softmax to improve the efficiency in computing the linear classifier. As, in our experiment, the number of classes is so small that hierarchical softmax has no obvious advantages, we will not discuss it in this chapter.

2.3.2 Character-level Convolutional Network

Different with most of the deep learning models which deal with the inputs on word level, Character-level Convolutional Network (Char-CNN) treats text as a kind of raw signal at character level and applies deep convolutional networks (ConvNets) on them. Compared with conventional machine learning methods and other deep learning models, Char-CNN has many advantages. Firstly, as the inputs are treated as signal on character-level, Char-CNN does not require the knowledges of words. Secondly, previous research proves that the knowledge about syntactic or semantic structure of a language are not required in ConvNets. Thirdly, as the model accepts a sequence of characters which are encoded by prescribing an alphabet for the text, Char-CNN can deal with different language. Fourthly, the model only works on character level, thus, it can easily learn the abnormal character combinations such as emoticons and misspellings.

The architecture of the Char-CNN is shown in Figure 2.3. The model receives the text

Figure 2.3: The model architecture of Character-level Convolutional Network [2]



and quantize each character. This pre-process is called character quantization which is done by prescribing an alphabet of size m for the raw text, and each character is quantized into one-hot encoding. For English text, we normally use the alphabet consisting 70 characters, which is presented as follows

```
abcdefghijklmnopqrstuvwxyz0123456789
-,;.!?:"/\|_@#$&*~`'+-=<>()[]{}'^%
```

There are 26 English letters, 10 digits, 33 other characters and the new line character, which is not shown above. For example, the character “b” is quantized into [0,1,0,0,0,.....,0]. Only the second element has value of 1 while other elements are all zero.

The encoded text is fed to the convolutional and the maxpooling layers after the quantization. These two layers are the key components of the ConvNets. In Char-CNN, the temporal, as known as one-dimensional, convolutional modules are used. The temporal convolution is similar to the convolutions commonly used in computer vision, which just simply computes a 1-D convolution. For a input function and a kernel function in form of $g(x) \in [1,l] \rightarrow \mathbb{R}$ and $f(x) \in [1,k] \rightarrow \mathbb{R}$, where l is the input feature size and k is the kernel size. The convolution function $h(y) \in [1, \frac{l-k}{d}+1] \rightarrow \mathbb{R}$ is defined as

$$h(y) = \sum_{x=1}^{k} f(x) \cdot g(y \cdot d - x + c) \tag{2.9}$$

where d is the stride, c is an offset constant which is defined as

$$c = k - d + 1 \tag{2.10}$$

For a ConvNets with input feature size of m and output feature size of n, each inputs and outputs element are $g_i(x)$ and $h_j(y)$ correspondingly. And the weight of the kernel is $f_{i,j}(x)$ where $i = 1,2,...,m$ and $j = 1,2,...,n$. The output $h_j(y)$ is the sum over i of the convolutions

Layer	Feature	Kernel Size	Pooling Size
1	1024	7	3
2	1024	7	3
3	1024	3	N/A
4	1024	3	N/A
5	1024	3	N/A
6	1024	3	3

Table 2.1: The configurations of the convolutional layers in Character-level Convolutional Network

between $g_i(x)$ and $f_{i,j}(x)$. As objects tend to have a local spatial support, for example, the connect between each characters in a word, such computation can extract the local features which makes ConvNets so powerful.

After the convolutional layer, it is the temporal max-pooling layer which helps us to train deeper models. Pooling is a effective method used to combine several nearby features into a local global 'bag of features' to preserve task-related information while removing irrelevant information. For example, the vector $\mathbf{v}$ with P components is reduced by the pooling layer h to a single scalar $h(v)$. There are two main pooling way: average pooling $h(\mathbf{v}) = \frac{1}{P}\sum_{i=1}^{P} v_i$ and max pooling $f(\mathbf{v}) = \max_i v_i$. The temporal max-pooling is the same as spatial max-pooling except it is in 1-D [29]. For a input function $g(x) \in [1,l] \to \mathbb{R}$, the max-pooling function $h(y) \in [1, \frac{l-k}{d}+1] \to \mathbb{R}$ is defined as

$$h(y) = \max_{x=1}^{k} g(y \cdot d - x + c) \tag{2.11}$$

In Char-CNN, there are 6 convoltional layers and the first two convolutional layers and the last convolutional layer followed by a max-pooling layer. The configurations of the convolutional layers are presented in Table 2.1

Following the 6 convolutional layers, there are 3 fully connected layers. The non-linearity used in Char-CNN is the ReLUs which defined as

$$\text{ReLU}(x) = \max(0, x) \tag{2.12}$$

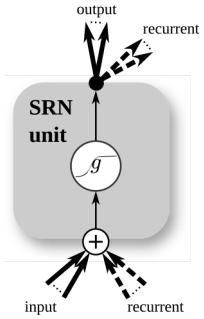
The configurations of the fully connected layers is shown in Table 2.2

All the weights in the Char-CNN is initialed by a Gaussian distribution with mean of 0 and standard deviation of 0.05. Same as fastText, the Char-CNN is trained by stochastic gradient descent with batch size of 128. It also uses momentum to accelerate the descent.

Layer	Output units
7	1024
8	1024
9	Number of classes

Table 2.2: The configurations of the fully-connected layers in Character-level Convolutional Network

Figure 2.4: The mechanism of the recurrent neural network [3]



The 0.9 momentum is initialed with the step size of 0.01, which is halved 3 epochs for 10 times.

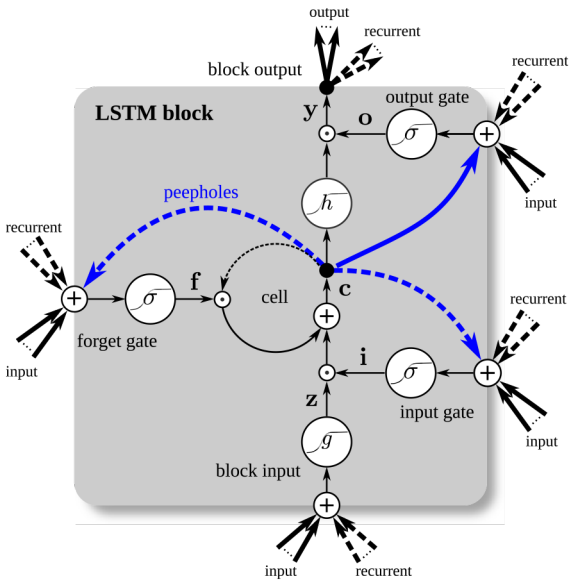
Compared with traditional machine learning method in text classification, such as bag-of-words [14], bag-of-ngrams, bag-of-means on word embedding [30], and some deep learning methods, such as word-level ConvNets, Long-short term memory, Character-level Convolutional Network is an effective method and it performs better than most of the traditional machine learning methods in large datasets [2].

2.3.3 Bidirectional Long-short Term Memory Network

Recurrent Neural Network (RNN) is treated as a good type of neural network to deal with sequence-like data, because it can use its internal state to process sequences of inputs and present the dynamic temporal behavior for the sequence. The basic mechanism of RNN is shown in Figure 2.4. The simplest form of RNN is an multiple layer perceptron with the previous set of the hidden unit activations feeding back into the network along with the inputs [31].

However, RNN has the gradient vanishing problem which makes it hard to train. To overcome this problem, many new types of recurrent neural networks were introduced. The Long-Short Term (LSTM) unit is one of the most effective units which was firstly proposed by Hochreiter and Schmidhuber [32]. The basic idea behind LSTM units is that it introduces

Figure 2.5: The schematic of the LSTM unit [3]



some gates to control the degree which the units keep the previous states and memorize the current inputs. The detailed schematic of LSTM unit is shown in Figure 2.5.

We suppose the input vector at time t is $\mathbf{x}^t$, the state at time $t-1$ is $\mathbf{h}^{t-1}$ N is the number of LSTM units, and M is the number of instances. There are following weights for the LSTM unit:

- Input weights: $\mathbf{W}_z, \mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o \in \mathbb{R}^{N \times M}$
- Recurrent weights: $\mathbf{R}_z, \mathbf{R}_i, \mathbf{R}_f, \mathbf{R}_o \in \mathbb{R}^{N \times N}$
- Peephole weights: $\mathbf{p}_i, \mathbf{p}_f, \mathbf{p}_o \in \mathbb{R}^N$
- Bias weights: $\mathbf{b}_z, \mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o \in \mathbf{R}^N$

Then the computation of LSTM layer forward propagation can be presented as

$$\mathbf{z}^t = g(\mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{h}^{t-1} + \mathbf{b}_z) \tag{2.13}$$

$$\mathbf{i}^t = \sigma(\mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{h}^{t-1} + \mathbf{p_i} \odot \mathbf{c}^{t-1} + \mathbf{b}_i) \tag{2.14}$$

$$\mathbf{f}^t = \sigma(\mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{h}^{t-1} + \mathbf{p_f} \odot \mathbf{c}^{t-1} + \mathbf{b}_f) \tag{2.15}$$

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t \tag{2.16}$$

$$\mathbf{o}^t = \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{h}^{t-1} + \mathbf{p_o} \odot \mathbf{c}^t + \mathbf{b}_o) \tag{2.17}$$

$$\mathbf{h}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t \tag{2.18}$$

where σ, g, and h are activation functions

$$\text{Logistic sigmoid: } \sigma(x) = \frac{1}{1+e^{-x}} \tag{2.19}$$

$$\text{Hyperbolic tangent: } g(x) = h(x) = \tanh(x) \tag{2.20}$$

Based on this mechanism, LSTMs do not suffer from the optimization problem and can capture long-term temporal dependences. LSTMs have been applied to many difficult problem, such as handwriting generation [33], translation [34], speech synthesis [35].

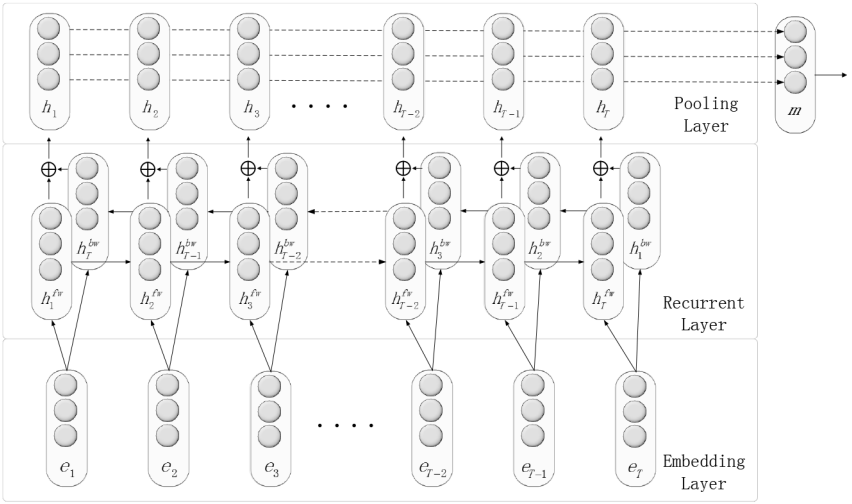
However, the disadvantage of LSTMs is that they ignore the future context as it only read the data in temporal order. To overcome this limit, Bidirectional LSTMs (BLSTM) introduces a second layer where the LSTM units process the sequence in opposite order. Based on this design, the model has the ability to extract the features both from past and future. There are two main method to combine the outputs from forward and backward propagation. One is concatenating two output into a new vector at each time-step. Another way is using element-wise sum operator. In this case, the output at time i is defined as

$$\mathbf{h}^t = \mathbf{h}^{t,\leftarrow} \oplus \mathbf{h}^{t,\rightarrow} \tag{2.21}$$

Zhang et al. firstly proposed a text classification model based on BLSTM [36]. The architecture of the model is shown in Figure 2.6. Similar to fastText model, the first layer is the embedding layer on word level. To use the knowledge of general domains, the weights of the embedding layer are initialized by the pre-trained word vector. The pre-trained word vector is obtained by the word2vector tool [37]. In the BLSTM layer, the output is obtained by sum the backward and forward outputs as introduced in Equation 2.21. Based on the hypothesis that only several key words and the associated features contribute to the classification, the model uses the max-pooling rather than average-pooling after the BLSTM layer to extract the semantic meaning of a sentence.

The prediction result is given by the output layer with the activation function of softmax. Similar to aforementioned models, the loss function is the negative log-likelihood and the optimization is done by the stochastic gradient descent algorithm. In order to help prop-

Figure 2.6: Model structure of bidirectional LSTM



agating the gradient back to early steps easier, the fan-in technique is used to initialized the weights [38].

Compared with ConvNets-based approach which is the lack of capability to learning temporal features, the BLSTM shows a significant advantage in dealing with long-distance patterns. In addition, the semantic distribution of the BLSTM based model tends to be smoother than the one produced by the ConvNet-based model.

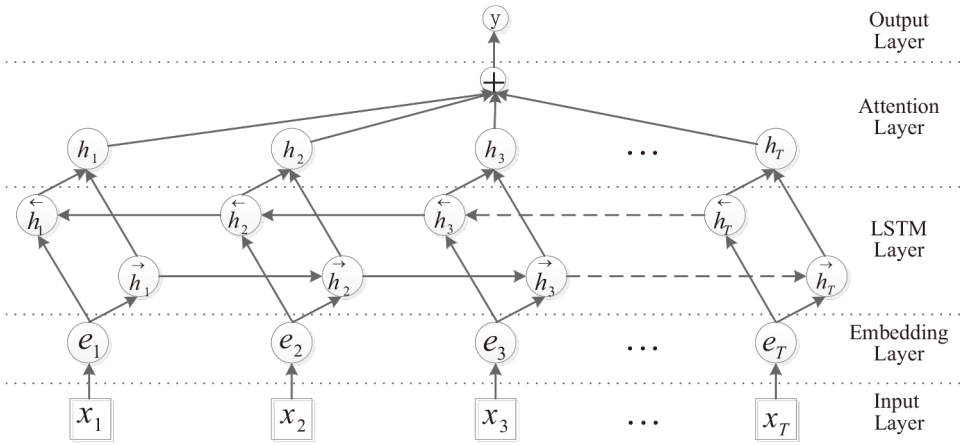
2.3.4 Attention Based Bidirectional Long Short-Term Memory Network

However, BLSTMs are effective models in dealing with sequence, it still has some disadvantages. As not every word in the sequence has the same importance, BLSTMs do not have such ability to capture the most important semantic information in a sentence. The ideal classification model is expected to be able to automatically focus on the words that have decisive effective on the prediction results. Attention mechanism was proposed to tackle this problem. Combining BLSTM and attention mechanism, a attention-based BILSTM (Att-BLSTM) for classification was proposed and outperforms most of the existing approaches [39]. The structure of A-BLSTM is shown in Figure 2.7. In this model, the forward and backward is combined by the element-wise sum.

Firstly, we suppose $\mathbf{H}$ is the matrix which consists the output vectors $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T]$from BLSTM, where T is the length of the sentence. The representation $\mathbf{r}$ of the sentence is defined as

$$\mathbf{M} = \tanh(\mathbf{H}) \tag{2.22}$$

Figure 2.7: Model structure of attention-based bidirectional LSTM



$$\alpha = \text{softmax}(\mathbf{w}^{\intercal}\mathbf{M}) \tag{2.23}$$

$$\mathbf{r} = \mathbf{H}\alpha^{\intercal} \tag{2.24}$$

where $\mathbf{H} \in \mathbb{R}^{d \times T}$, d is the dimension of the word vector. The sentence-pair representation used for linear classification is in form of

$$\mathbf{y}^{*} = \tanh(\mathbf{r}) \tag{2.25}$$

To avoid overfitting, A-BLSTM uses the nagative log-likelihood with L2 regularization:

$$L(\theta) = -\frac{1}{m}\sum_{i=1}^{m} t_i \log(y_i) + \lambda \|\theta\|_F^2 \tag{2.26}$$

where $\mathbf{t} \in \mathbb{R}^m$ is the true label encoded as one-hot vector, $\mathbf{y} \in \mathbb{R}^m$ is the probability for each class produced by softmax function, m is the number of classes, and λ is the regularization hypeparameter which controls the penalty to the complex model.

In Att-BLSTM, dropout method is also used to do the regularization. During the training process, the dropout method is used on embedding layer and BLSTM layer. Dropout technique was first introduced by Hinton et al. in image classification[40]. It set the output of each neuron to zero with a specific probability. In this way, a part of neurons are dropped out and do not contribute to the forward and backward propagation. In Att-BLSTM, the dropout rate of embedding layer and LSTM layer is set as 0.3, 0.5 respectively.

The embedding layer is initialized by the pre-trained word embedding by Turian et

al [41]. Other weights are initialized randomly. Different with aforementioned models which trained using stochastic gradient descent, Att-BLSTM is trained using AdaDelta [42] with a learning rate of 1.0 and batch size of 10. The L2 regularization is set as 10^{-5}.

Att-BLSTM achieve a similar result to the BLSTM which used many features derived from NLP tools and lexical resources [**?**].

Chapter 3

Concept Drift Detection

3.1 Data Used

The whole dataset was collected by web crawler, containing 181903 fashion news in www.hypebeast.com from 20/04/2005 to 13/06/2018. A sample of the dataset including the date of the publication, the category, the keywords, the title, the context, the HYPES, and the class of the HYPES. The HYPES is the real time popularity metric informing reader of what is trending. The HYPES is computed based on the page-viewers, the comments, and the information flow on social network. The HYPES is classified into 3 classes which is shown in Table 1.1 of the introduction chapter.

During the development of the website, HYPEBEAST also changed the algorithm of computing the HYPES. Figure 3.1 shows the change of the HYPES from 2015 to 2018. And the distribution of each class over the past 13 years is presented in Figure 3.2. It is obvious that after the middle of 2013, HYPES have a significant jump and the present a rising trend until 2018. However, the proportions of classes remain stable after 2014. Based on this intuitive analysis, we select the data from 01/01/2014 to 13/08/2018 as the dataset for the experiment. It containing 95787 fashion news. The distribution of the classes is shown in Table 3.1

Popularity	Number	Proportion
Cold	52852	0.5518
Meidum	35944	0.3752
Hot	6991	0.0730

Table 3.1: Distribution of classes of the dataset

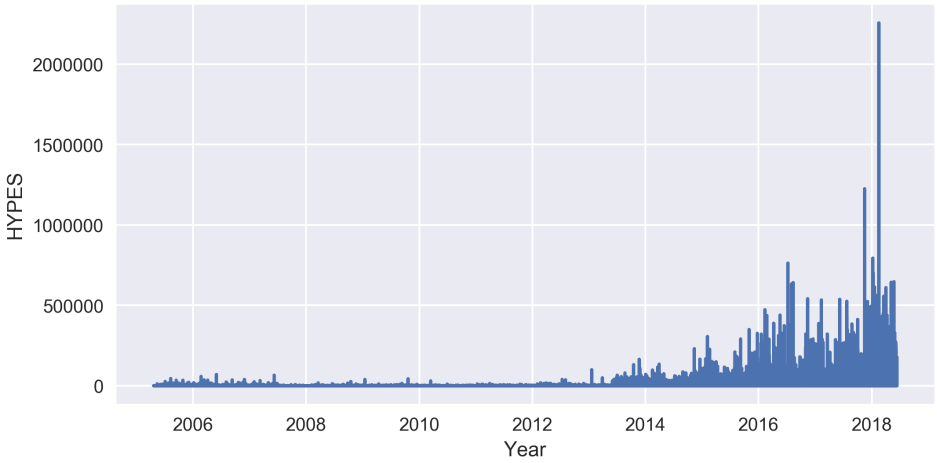


Figure 3.1: HYPES over years

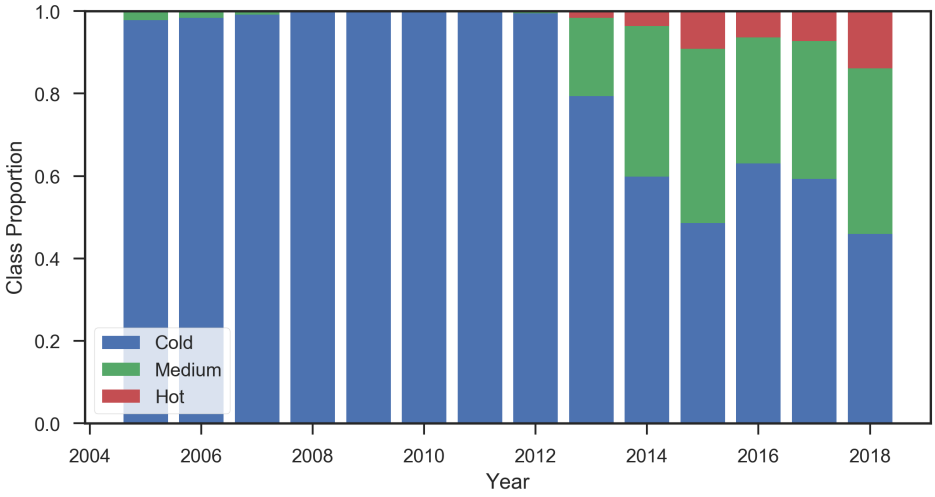


Figure 3.2: Proportion of each class over years

3.2 Model Design

As shown in Figure UNKNOWN, the model proposed in this experiment contains three components: (1) a pre-process part that extract features from raw text; (2) a base learner that gives a prediction with a subset of examples; (3) a concept drift detection algorithm which detects the drift and select the subset of the example to re-train the base learner. These components will be presented in detail in this section.

3.2.1 Preprocessing

Before the training data can be fed to the model, it needs to be transformed to a format the model can understand which is called preprocessing including tokenization, vectorization, and feature selection. The scheme of the pre-processing is shown in Figure UNKNOWN.

In natural language processing problem, tokenization is the procedure of splitting a text into words, phrases, or other meaningful parts. In the other words, tokenization is kind of segmentation. In conventional machine learning problem, Bag-of-words model is the most popular representation to tokenize the text. In this model, a raw article is represented as a bag of its words. It is a commonly used and also very effective model. However, bag-of-words model ignores the order of the words which leads to the same representations of the sentences "Kanye loves Kim." and "Kim loves Kanye.". As an alternative, the bag-of-ngam model is used to store the spatial information. To some extent, bag-of-words is a special type of bag-of-ngram which only counts the unigram.

For example, in a bag-of-ngrams model, text is represent as a collection of unique ngrams (groups of n adjacent tokens). Consider the text "The YEEZY BOOST restock gets postponed". Here, the word unigrams $(n = 1)$ are ['The', 'YEEZY', 'BOOST', 'restock', 'gets', 'postponed'] and the bag-of-bigrams $(n = 2)$ is ['The YEEZY', 'YEEZY BOOST', 'BOOST restock', 'restock gets', 'gets postponed'].

In this experiment, we tokenize the text into both unigrams and bigrams which not only consider the distribution of the words but also the local connection between the words. Thus, the tokenization result of the sentence aforementioned is

```
['The', 'YEEZY', 'BOOST', 'restock', 'gets', 'postponed', '
   The YEEZY', 'YEEZY BOOST', 'BOOST restock', 'restock gets
   ', 'gets postponed']
```

After the tokenization,the ngrams need to be truned into numerical vectors which can be processed by the machine learning algorithm. Firstly, every ngram is assigned a unique index which ranges from 0 to the number of ngrams in the corpus. The example below shows the indexes assigned to the unigrams and bigrams generated for two sentences.

```
Raw texts:
        'Where to buy the YEEZY BOOST.'
        'Your first look at the YEEZY BOOST.'
Index assigned for every ngrams:
        {'Where': 7, 'to': 6, 'buy': 2, 'the': 5, 'YEEZY':
           8, 'BOOST': 1, 'You': 9, 'first': 3, 'look': 4, '
           at'    0}
```

Once the indexes are siggned to the ngrams, we vectorize using the encoding methods.

There are three popular encoding methods. The first one is the one-hot coding by which the text is represented as a vector indicating the presence or absence of a token in the text. The second one is the count encoding by which the text is encodded as a vector indicating the count of a ngrams. The disadvantage of the above two methods is that common words such as 'the', 'and' are not pernalized. These words occurs in similar frequencies in all texts and are not particularly unique to the text samples in the dataset. Thus, in our model, we use the tf-idf encoding to represent the ngrams. tf means term frequency and idf means inverse document frequencey. The formula used to compute the tf-idf of ngrams is that

$$\text{tf-idf}(d,t) = \text{tf} * \text{idf}(d,t) \tag{3.1}$$

and the idf is computed as

$$\text{idf}(d,t) = \log(\frac{n}{\text{df}(d,t)}) + 1 \tag{3.2}$$

where n is the total number of documents and $\text{df}(d,t)$ is the document frequency which is defined as the number of documents d that contain term t. The constant '1' is added to make the terms with zero idf will not be entirely ignored. tf-idf helps to ajust for the fact that some words appear more frequently in general.

During above process, we also apply some tricks. One of them is the lowercase conversion. Since uppercase or lowercase forms words are assumed to have no difference. Another is the stop-words removal. The stop-words are usually assumed to be irrelevant in text classification [43].

The vectorized feature vectors are alway high-dimensional in text classification problem because the size of the corpus is very large. In addition, not every feature extracted from the text is useful for the classification. In this case, feature selection is a necessary step which is helpful (1) to improve the performance of the classifier and avoid the overfitting; (2) to reduce the memory storage and provide a more cost-effective model (3) to gain a deeper insight into the underlying processes that generated the data.

If the features are categorical, we normally calculate a chi-square statistic between each feature and the target vector. However, the encoded features are quantitative. Therefore, we select the features by computing the ANOVA F-value between each feature and the target vector. The F-val scores examine if, when we group the numerical features by the target vectors, the means for each group are significantly different. Based on F-val scores, we can

remove the features so as are statistically uncorrelated with class labels.

Suppose the encoded feature vectors $\mathbf{X} \in \mathbb{R}^{n \times m}$, where n is the number of examples and m is the number of features, and the labels $\mathbf{Y} \in \mathbb{R}^{n \times l}$, where l is the number of classes. We assume to select k relevant features from m features. Hence, there are $\binom{n}{k}$ combinations. Then, we compute the F-score for each combination with the ANOVA table and retain the k features from the group with the highest F-score.

3.2.2 Base Learner

In the experiment, the naive bayes classifier is used as base learner. It is because it naturally support incremental learning which can provide a fast and cost-efficient model.

For a given document d with class of y, the Bayes' theorem present the relationship as a follow:

$$P(y|d) = \frac{P(d|y)p(y)}{P(d)} \tag{3.3}$$

As what we mentioned in section of pre-processing, the document d is represented as features $(x_1, x_2, ..., x_k)$. The Naive Bayes method using a strong and naive assumption that the feature probabilities $P(x_i|y_j)$ are independent given the class y. Hence, the relationship is simplified to

$$\begin{aligned} P(y|d) &= p(y|x_1, ..., x_k) \\ &= \frac{P(y)P(x_1, ..x_k|y}{P(x_1, .., x_k)} \\ &= \frac{P(y)\prod_{i=1}^{k} P(x_i|y)}{P(x_1, ..., x_k)} \\ &\propto P(y)\prod_{i=1}^{k} P(x_i|y) \end{aligned} \tag{3.4}$$

The prediction result can be obtained by using the Maximum A Posteriori (MAP)

$$\hat{y} = \text{argmax}_y P(y)\prod_{i=1}^{k} P(x_i|y) \tag{3.5}$$

In the experiment, we use the multinomial naive Bayes algorithm which is commonly used in text classification. The data distribution is parametrized by the vectors $\theta_y = (\theta_{y_1}, ..., \theta_{y_k})$ for each class y, where θ_{y_i} is the probability $P(x_i|y)$ which is estimated as

$$\hat{\theta}_{y_i} = \frac{N_{y_i} + \alpha}{N_y + \alpha k} \tag{3.6}$$

where N_{y_i} is the number of times feature i appears in a sample of class y in the training set and N_y is the total count of all features for class y.

The hypeparameter α is the smoothing parameter which incorporate a small-sample correction. It is a type of regularization. When a given test point whose class and one of features never occur together in the training set, the estimated probability will be zero. Because of the multiplication, the posteriori will be zero. Thanks to the smoothing parameters, no probability is set to zero. When $\alpha = 1$, the smoothing is called Laplace smoothing, and Lidstone smoothing in the general case.

3.2.3 Concept Drift Detection Algorithm

In the experiment, we use Drift Detection Method (DDM) proposed by Gama et al. [10]. Previous experiments have prove that DMM has the best performance in both gradual and abrupt drift [15]. In addition, compared with other detector, DMM is relative fast and cost-efficient.

DMM was proposed by the probably approximately correct learning model [26] that if the distribution of the data is stationary, the error rate of the classifier will decrease with the increase of the number of the instances. Therefore, a great increase of the error rate indicates the change of the distribution, namely, the concept drift.

For every instance at time i, the error-rate p_i and the standard deviation $s_i = \sqrt{p_i(1-p_i)}$ are computed. p_{min} and s_{min} are stored when p_i and s_i reach their minimum. A warning level is reached if $p_i + s_i \geq p_{min} + w \cdot s_{min}$ and a drift occurs when $p_i + s_i \geq p_{min} + d \cdot s_{min}$, where w and d are number of standard deviation to warn (w) and detect (d) drift. With confidence levels for warning and drift set to 95% and 99% respectively, w is 2 and d is 3. When a warning level is reached, the instances after the warning level are stored. Once the drift occurs, the base learner will re-trained by the stored instance and the values of p_{min} and s_{min} will be reset. It often happens that the error rate increases and reach the warning level but decrease later. It is called false alarm. Once the false alarm occurs, the stored instances will be thrown away. Based on this mechanism, DDN keeps the base learner better adapted to the present data distribution.

DMM can be applied to any machine learning algorithm. It can be directly implemented inside the incremental learning algorithm and also can be implemented as a wrapper to batch learner. In this experiment, the DMM is implemented as a wrapper over the multinomial Bayes classifier that incrementally learns the dataset. Figure UNKOWN presents

the process at time i

Algorithm 1 DMM with Bayes Classifier

Require: Φ: Current Bayes Classifier, $\mathbf{x}_i, y_i$: Current example
  Let $\hat{y}_i \leftarrow \Phi(\mathbf{x}_i)$
  Compute p_i and s_i
  if $p_i + s_i < p_{min} + s_{min}$ **then**
    $p_min \leftarrow p_i$
  end if
  if $p_i + s_i < p_{min} + w \cdot s_{min}$ **then**
    Warning $\leftarrow$ False
    Update Φ with $\mathbf{x}_i, y_i$
  else
    if $p_i + s_i < p_{min} + d \cdot s_{min}$ **then**
      if Not Warning? **then**
        buffer $\leftarrow \{\mathbf{x_i}, y_i\}$
        Warning? $\leftarrow$ True
      else
        buffer $\leftarrow$ buffer $\cup \{\mathbf{x_i}, y_i\}$
      end if
    else
      Retrain Φ with the instances in the buffer
      Warning? $\leftarrow$ False Reset p_min and s_min
    end if
  end if

3.2.4 Evaluation Step

To evaluate the performance of the concept drift detector, we use an approach described by Elwell et al. [44]. For each time t, the instance from time $t+1$ is used as the test data. We assume T as the time step. For each time step T, there are specific amount of test instances. Then, a variety of metrics is computed for each time step to compared the performance between the concept drift detector and the base learner.

Firstly, we focus on the basic metric, the accuracy. For each test instance, the Bayes Classifier will return a boolean value indicating if it correctly classified the instance or not. For every time step T, the accuracy is defined as

$$\text{Accuracy} = \frac{\text{Number of correctly predicted instance}}{\text{Number of instances}} \tag{3.7}$$

However, the data set is imbalanced, the number of hot news is significantly less than other two types of news. High accuracy does not indicate good classification ability. Using accuracy to evaluate the models will drive the models focus on the majority class but ignore

the minority. In this case, accuracy alone is typically not enough information to do the comparison. In addition, we expect the hot news can be predicted correctly as much as possible. Thus, the recall is used to check if the model returns most of the relevant result.

For binary classification, recall is intuitively defined as

$$\text{Recall} = \frac{tp}{tp + fn} \tag{3.8}$$

where tp is the number of correctly predicted positive instances, and fn is the predicted correctly negative instance. However, our model is designed to deal with multiclass classification problem. The definitions of true positive (tf) and false negative (fn) are slightly different. The multiclass problem is separated into several binary classification problem. For class i, the true positive tf_i and false negative (fn_i) are computed by treating class i as the positive class and other classes are all negative. Then, we compute the micro-average recall as it aggregates the contributions of all classes to compute the average metric. The micro-average recall is defined as [45]

$$\text{micro-Recall} = \frac{\sum_i tp_i}{\sum_i (tp_i + fn_i)} \tag{3.9}$$

In addition, we also compute the micro average F1 score which is the weighted average of micro-average precision and micro-average recall.

$$\text{micro-F}_1 = 2 * \frac{\text{micro-Precision} * \text{micro-Recall}}{\text{micro-Precision} + \text{micro-Recall}} \tag{3.10}$$

where definition of the micro-average precision is similar to the micro-average recall which is shown as a follow

$$\text{micro-Precision} = \frac{\sum_i tf_i}{\sum_i (tf_i + fp_i)} \tag{3.11}$$

where fp is number of negative instances which are predicted as positive.

In addition to pay attention the prediction result of the instances at each time step, the overall performance of the model will be evaluated as well. Apart from accuracy, micro-average recall, and micro-average F1 score, the normalized confusion matrix will be computed and visualized.

Table 3.2: Parameter configuration of the drift detectors

Detector	n	w	d
1	30	2.0	3
2	30	1.5	2.5
3	50	2.0	3.0
4	50	1.5	2.5

Model	Average	Recall	F1
Detector 1	0.6018	0.4070	0.3921
Detector 2	0.6011	0.4066	0.3918
Detector 3	0.6128	0.4138	0.3985
Detector 4	0.6130	0.4139	0.3986
Base leaner	0.6135	0.4357	0.4264

Table 3.3: Averaged metrics of the detectors with different parameters

3.2.5 Parameters Configuration

The base learner only has a parameter, the additive smoothing parameter, which needs to be tune. A 5-fold cross-validation is applied and α is set to 0.1. In addition, based on 5-fold cross-validation, we keep the top 20000 features as the input.

The DDM has three parameters, including the minimum number of instances before permitting detecting change (n), the number of standard deviations to warn (w) and detect (d) drift. Testing is performed using following values for the parameters: $n \in \{15, 30\}$,$w \in \{1.5, 2.0\}$, and $d \in \{2.5, 3.0\}$. The detectors with different parameters are presented in Table 3.2

3.3 Experimental Results

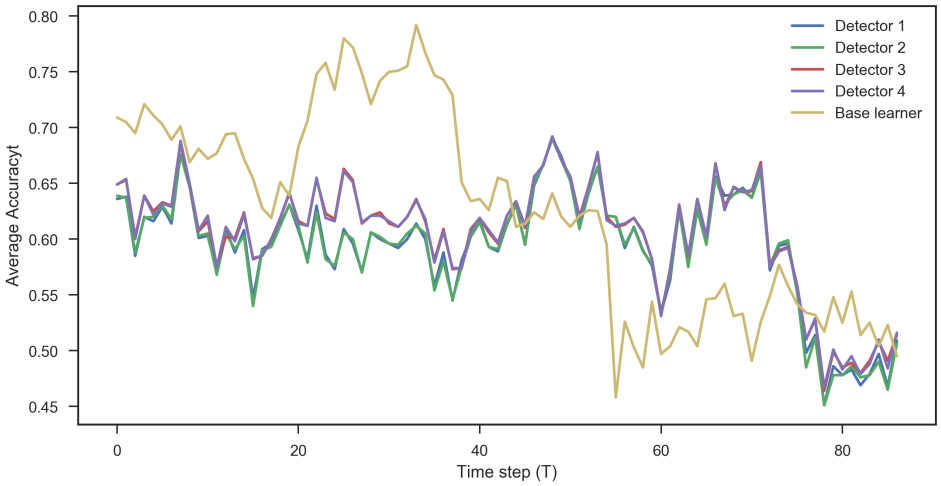
In this section, the results of the experiment with the drift detection framework, concerning the prediction performance and the drift detection ability, are presented.

Figure 3.3 shows graphics of accuracies, recalls, and F1 scores at each time step T in the dataset. Each time step, there are 1000 instance. Table 3.3 presents the averaged metrics of each detector and the base learner.

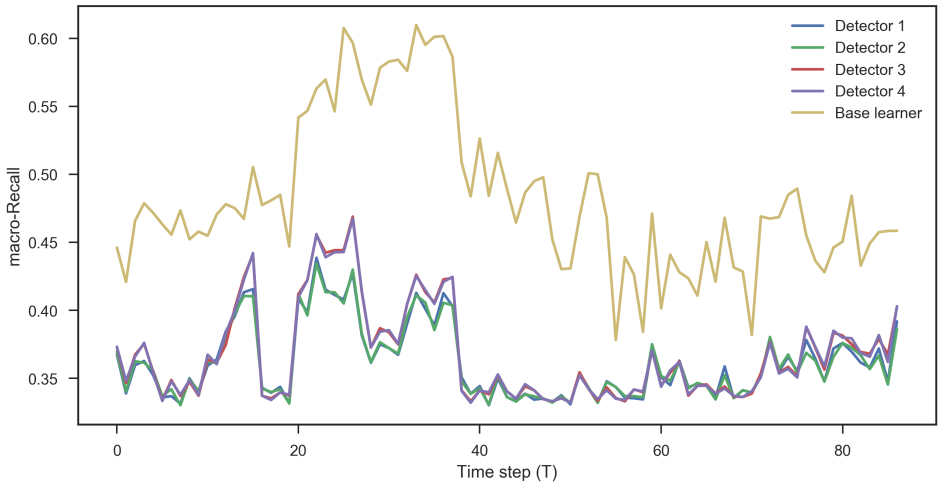
The confusion matrix

3.3.1 Drift Identification

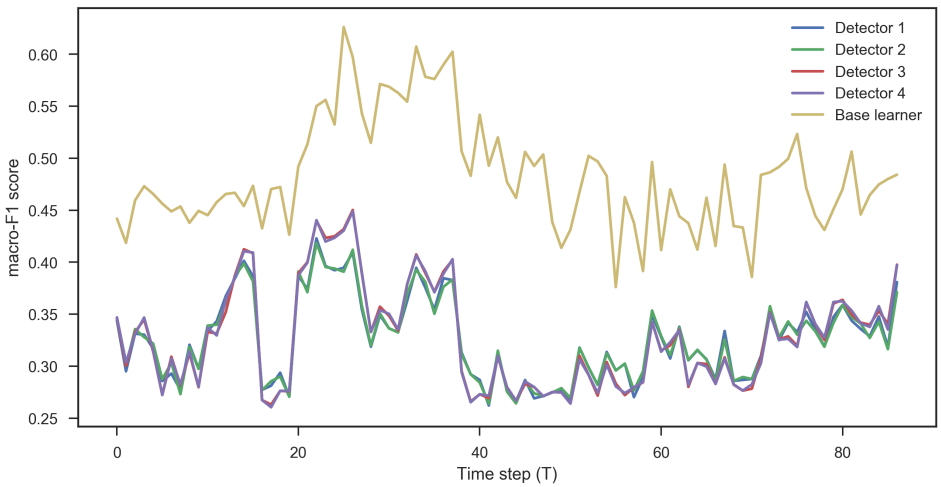
3.3.2 Predictive Accuracy



(a) Accuracy



(b) macro-average Recall



(c) macro-average F1 score

Figure 3.3: Metrics of the detectors with different parameters.

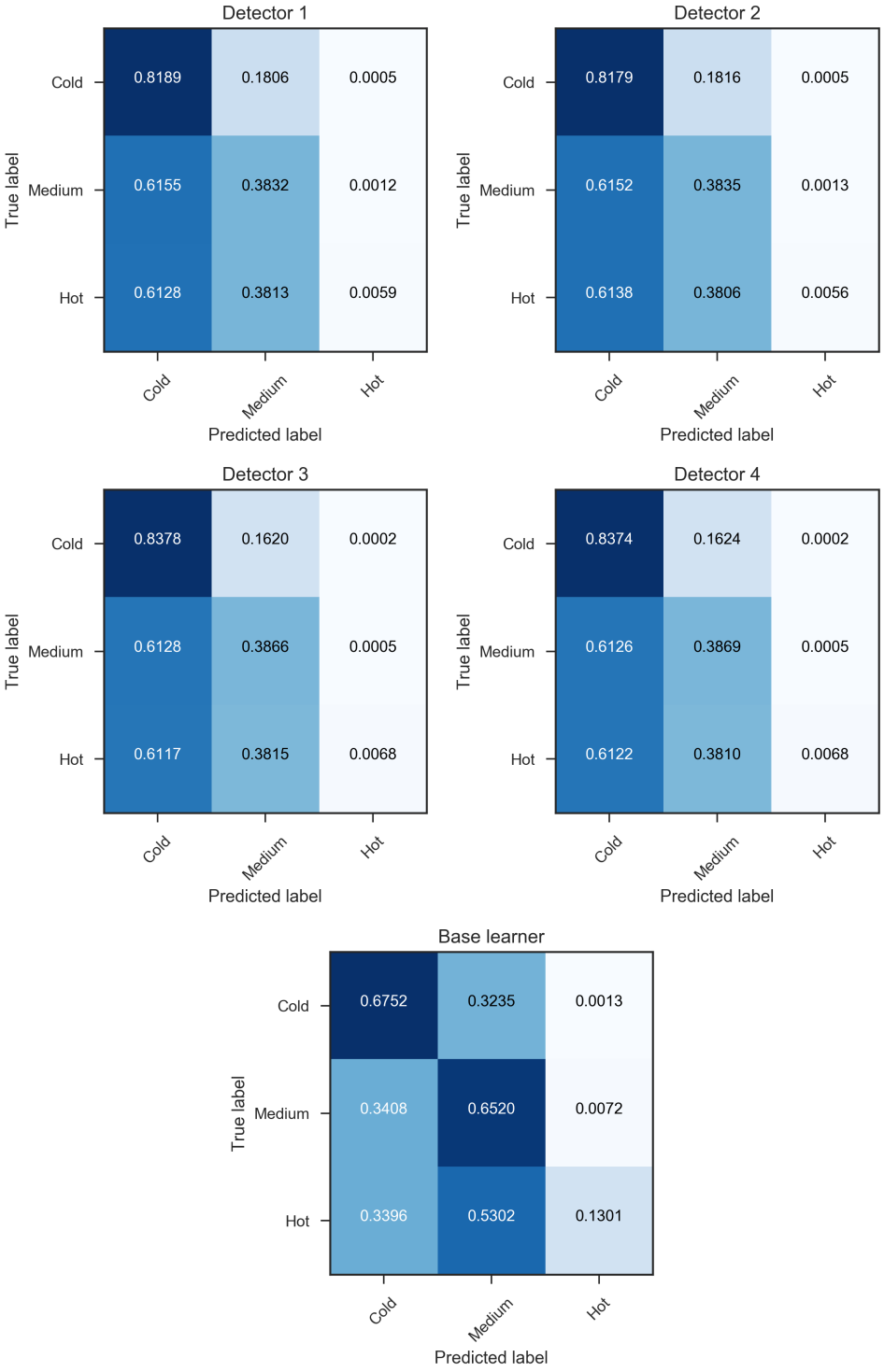


Figure 3.4: Confusion matrix of different detectors

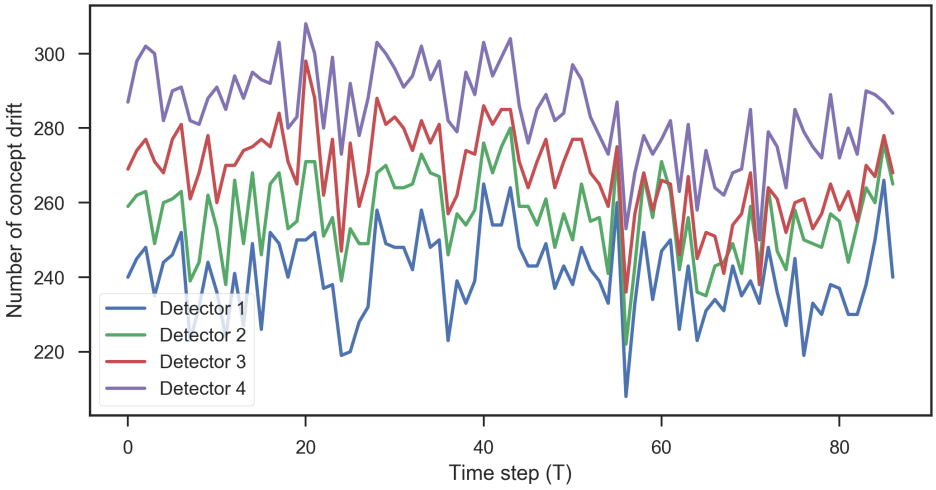


Figure 3.5: Number of concept drift detected at each time step

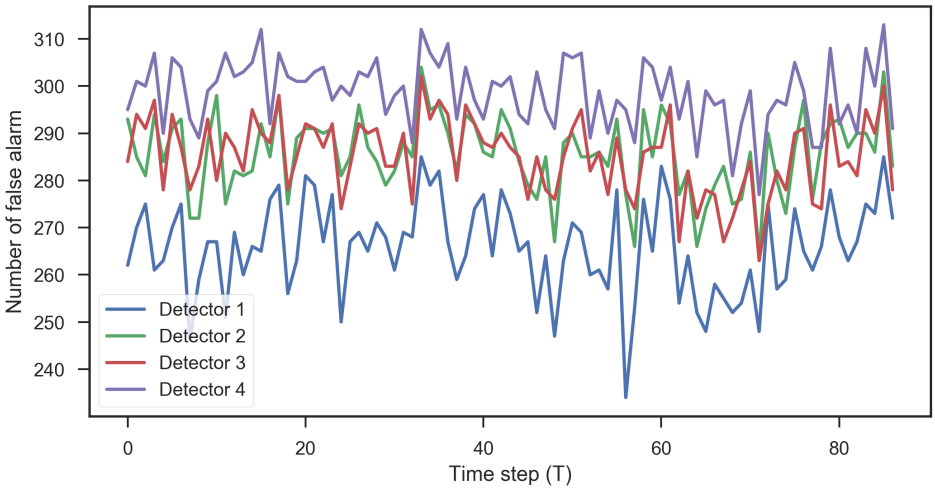


Figure 3.6: Number of false alarm at each time step

Chapter 4

Multi-Input Deep Neural Network in Fashion News Popularity Prediction

This chapter proposes a new deep learning model with two input layers to predict the fashion news popularity. The preprocessing of the deep learning model is introduced in detail. In addition, the techniques used in the model, including dropout, pooling, and bidirectional gate recurrent unit are discussed. This experiment further shows the prediction performance compared with four state-of-the-art model.

4.1 Data Used

In this experiment, we use similar dataset in experiment 1. The difference is that, as we do not use concept drift detection framework, we split the dataset into three parts. The fashion news from 1/04/2018 to 13/06/2018 are used as the test dataset. The dataset containing fashion news from 01/01/2014 to 31/03/2018 is split randomly into two subset. The subset containing 90% news is used to training the model and the subset containing rest 10% news is responsible to tune the hypeparameters. The number of examples in each dataset and the proportion of each class is shown in Table 4.1

To characterize the text classification problem and help tune the hypeparameters, the

Dataset	Number of news	Proportion of class		
		Hot	Medium	Cold
Trian	74996	0.0721	0.3794	0.5485
Validation	8333	0.0728	0.3760	0.5512
Test	4188	0.1445	0.3677	0.4878

Table 4.1: Properties of the datasets

	Number of words per sample		
	Median	Maximum	Average
Text	114	6258	158.59
Title	19	59	19.60
Keywords	3	36	3.24

Table 4.2: Number of words per sample

number of words in a sample from training set is collected and shown in Table 4.2

4.2 Model Design and Implementation

This section introduces the preprocessing of the raw data which is shown in Figure 4.1. Then, the structure of the model is discussed. As shown in Figure 4.2, the model proposed in this section contains two parts. The first part receives the unigram and bigram of the keywords and title as the input. The architecture of this part is similar to fastText model which uses a word embedding layers that maps each word in the input into a low dimension word vector. Then, the word vectors is averaged to produce a sentence-level representation. The second part received the preprocessed text and also produce the word vectors by an embedding layer. The difference is that a bidirectional GRU is applied to model the word sequence and output the word-level representation. Then both a max pooling layer and an average pooling layer are used to merges the word-level representations. The sentence-level features are produced by concatenate the outputs from these two pooling layers. Finally, the prediction result is given by the fully connected layers.

The basic idea behind this model is that the order of the words and the long distance pattern is important for modeling the text but unimportant for modeling the titles and keywords. The word distribution is feature that we expect to extract from the keywords and titles. Thus, we design two different feature extractors, one inspired by fastTex focus on the word vectors and one similar to BLSTM [36] uses RNN component to extract the long distance pattern. These components will be discussed in detail in this section.

4.2.1 Preprocessing

The procedure of the preprocessing for keywords, title, and text is shown in Figure 4.1.

In a sample, the keywords and title are concatenate to a new sentence. Though the long distance pattern of this new sentence is not the one of the features we want, it is useful to capture some participial information about the local word order. Thus, the bag of n-grams is used as the additional feature. Considering the efficiency and most of keywords that are

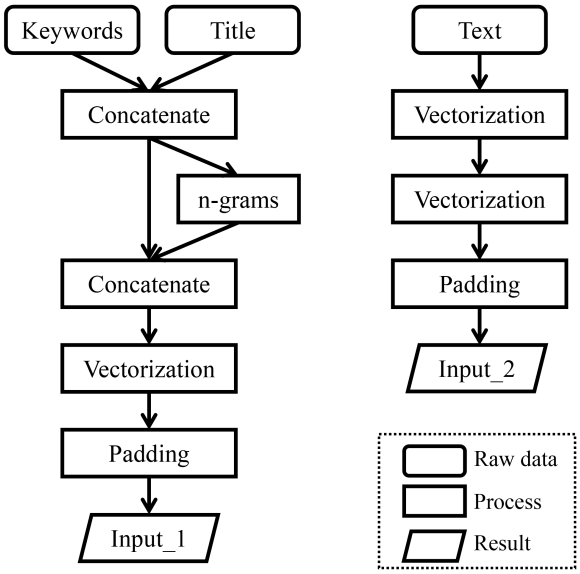


Figure 4.1: Preprocessing of the deep learning model

composed of two or fewer words, we use a bag of bigrams to keep the partial information. After this process, the sentence is represented as a sequence of unigrams and bigrams. Then the sequence is turned into numerical vector. A unique index is assigned to each unigram and bigrams. As keywords and title are the information extracted from the text, we keep all the words in our vocabulary.

However, not all words in the text contribute to the popularity predictions. In the process of tokening the text, the text is represented as a sequence of the unigrams and some rare and irrelevant words are discards from the vocabulary. In the experiment, we observe that using the most frequent 20,000 features is generally sufficient.

The final step for two inputs is the padding. Sequences that are shorter than max length are padded with 0 at the end while sequences longer than max length are truncated. In the experiment, the max length of the sequence is defined by the median number of words in each element. As shown in Table 4.2, the max length for title and keywords is sequence is 22 and that for text is 114.

4.2.2 Word Embeddings

The first component of the proposed model is the word embedding layers. Both inputs are projected to low-dimensional word vectors by the embedding layers. After that, each word gets mapped to a dense vector of real values representing that word's location in semantic

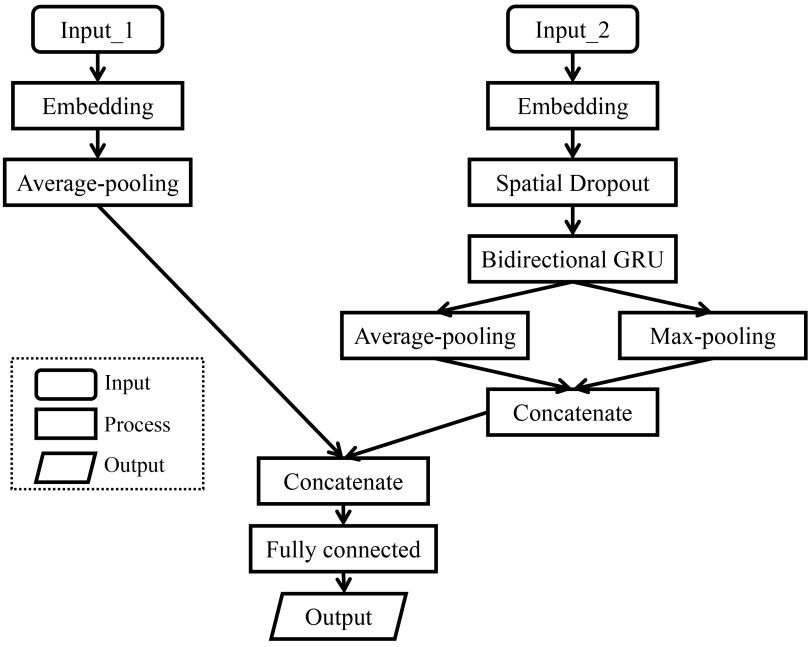


Figure 4.2: Structure of multi-input deep learning model

space. The advantage of using word vector is that the location and distance between vectors indicates how similar they are semantically.

A sequence of n indexes, where n is the max length of the input which defined at the process of padding, is firstly transfered to one-hot encoding matrix $\mathbf{X}$. Let $\mathbf{X} \in \{1,0\}^{n \times V}$ where V denotes the size of the vocabulary. The embedding layer transfers the one-hot encoding matrix into matrix containing composed of word vectors as follows

$$\mathbf{E} = \mathbf{XW} \tag{4.1}$$

where $\mathbf{W} \in \mathbb{R}^{V \times D}$ and D is the dimension of the word vector. Since $\mathbf{X}$ is one-hot, $\mathbf{W}$ is actually stores representations of all the words in the vocabulary.

There are two approach to initialize the projection weight $\mathbf{W}$. One is random initialization which will be updated during the training process. Another is using the pre-trained word vectors, by which the knowledge of general domains can be used. In our research, the embedding layer for the title and keywords is initialized randomly. Because, the keywords contains many celebrities' name, such as A$AP Rockey, Young Nudy, and brands, such as YEEZY, Suicoke, which are not commonly used words so that the pre-trained word vectors do not contain these words. The dimension of the word vectors is set to 20 as suggested by

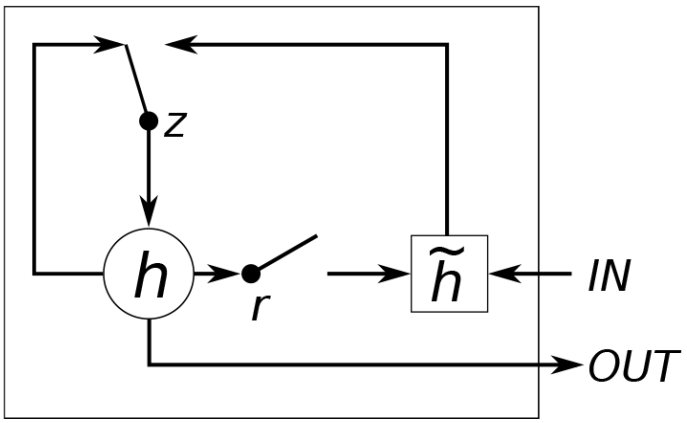


Figure 4.3: Graphical illustration of the GRU [4].

Joulin [28].

In contrast, the embedding layer for the text is initialized by the pre-trained weight. In order to compare with the work by Zhang [36] and Zhou et al. [39], the model uses the same word vectors proposed by Turian et al. [41], which are 50-dimensional, to initialize the projection matrix.

4.2.3 Bidirectional Gated Recurrent Unit Neural Networks

The second component of the model is the bidirectional recurrent layer which is the key component for modeling the sequence and capture the long-distance pattern.

We start from the gated recurrent unit(GRU). LSTM and GRU are both the variations of the RNN. But GRU can also be considered as a variation on the LSTM. The mechanism of the LSTM is introduced in Section 2.3.3. One of the difference between these two units is that GRU only has two gates which controls the information flow inside the units, however, without having an output gate that controls the exposure of the outputs. Another difference is the location of the input gate (reset gate). GRU controls the information from the previous activation when computing the new activation while LSTM controls it independently.

We assume $\mathbf{h}_t$ is the activation at time t of the GRU, which is defined as follow

$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j \tag{4.2}$$

where $\mathbf{h}_{t-1}$ is the previous activation, $\hat{\mathbf{h}}_t$ is the candidate activation, and $\mathbf{z}_t$ is the update gate which controls how much of the past information needs to be passed along to the future.

The update gate is computed by

$$z_t^j = \sigma(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1})^j \tag{4.3}$$

In addition, the candidate activation $\hat{\mathbf{h}}_t$ is computed by the equation which is similar to the traditional recurrent unit.

$$\hat{h}_t^j = \tanh(W \mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))^j \tag{4.4}$$

where $\mathbf{r}_t$ is the reset gate which controls how much of past information flow to forget. Its computation is similar to the update gate

$$r_t^j = \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1})^j \tag{4.5}$$

As we discussed in Section 2.3.3, the problem of the one-directional forward RNN is that the information of the future vectors are not fully utilized. Thus, we use the bidirectional architecture to extract the features from both past words and future words. The graphic illustration of bidirectional LSTM is shown in Figure 2.6. The bidirectional GRU is same to bidirectional LSTM, just change the type of the unit. Zhang [36] and Zhou et al. [39] both summed the outputs from forward and backward. In order to retain as much information as possible, we concatenate the outputs from both directions.

In the experiment, the size of the GRU unit is set to 80.

4.2.4 Pooling Layer

The third component of our model is the pooling layer. In the experiment, we use two types of pooling layers. The objective of using pooling layer is compress the word-level representations to sentence-level feature vectors. One of the compression method is the accumulation approach which use the feature vector produced at the last time step from RNN layer. However, we use the bidirectional RNN which can not defined what is last time step. Thus, most of RNN models resort to the pooling approach as in CNN models.

However, different from the pooling layer used in CNN models which specify the kernel size of the pooling layer, the pooling layer in RNN are mostly the global pooling. In

our research, the global max-pooling and global average-pooling are defined as follows:

$$m_i = \max_t\{(h_t)_i\} \tag{4.6}$$

$$m_i = \text{average}_t\{(h_t)_i\} \tag{4.7}$$

where m is the sentence-level feature vector and i indexes features dimensions [39].

For title and keywords, we use the global average pooling which is suggested by [28]. For the text, the novelty is that we use two pooling methods and the sentence-level representation is produced by concatenating the outputs from the pooling layers. The hypothesis is that some keywords play important a role in prediction but the overview of the sentence is also very important. So max-pooling extracts the keywords information while average-pooling provides the overview of the sentence.

4.2.5 Classifying

The sentence-level feature vectors from title&keywords and text are concatenated and passed to the final classifier. In the experiment, the fully connected neural network is classifier and softmax function is used to predict the label. For the concatenated feature vector $\mathbf{h}^* \in \mathbb{R}^d$, the prediction is given by

$$\mathbf{J} = W_j\mathbf{h}^* + b_j \tag{4.8}$$

$$\hat{(}p)(y|S) = \text{softmax}(W_S\mathbf{J} + b_S) \tag{4.9}$$

$$\hat{y} = \text{argmax}_y\hat{p}(y|S) \tag{4.10}$$

where W_j, b_j is the weights of the fully connected layer and W_S, b_S is the weights of the output layer.

As our dataset is very imbalance, traditional cost function, such as negative log-likelihood, does not punish the majority class. To solve this problem, we resort the focal loss in image classification. Focal loss was proposed by Lin et al. for dense object detection [46]. The definition of the focal loss can address the class imbalance by down-weighting the loss assigned to well-classified examples.

We assume $t \in \mathbb{R}$ is the ground truth, p_t is defined as

$$p_t = \hat{p}(t|S) \tag{4.11}$$

The focal loss is defined as

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t) \tag{4.12}$$

where γ is the focusing parameter adjusting the rate at which easy examples are down-weighted. And when $\gamma = 0$, FL is equivalent to cross entropy. Focal loss shows its advantage when $p_t \to 1$. $(1-p_t)^{\gamma}$ goes to 0 by which the easy example is down-weighted.

4.2.6 Regularization

We punish the complex model by adding an additional dropout layer after embedding layer. The role of dropout layer is preventing the activations from becoming strongly correlated so that the generalization of the model can be approved. It is an effective regularization method in deep learning model.

Instead of using standard dropout, we apply the spatial dropout to our model. The standard dropout set individual 'pixel' to 0randomly, which works well on images because adjacent pixels are highly correlated. However, the correlation between the feature vectors are more important on texts. Thus, Spatial dropout randomly omits the entire feature maps by setting them to 0, which works better in our model [47].

In the experiment, the dropout rate is set to 0.2, which indicates 20% of the word vectors will be dropped.

4.2.7 Evaluation Metrics

The evaluation metrics in this experiment is similar to the experiment 1, including accuracy, macro-F1, macro-Recall, and the confusion matrix. The difference is that we use Area Under the macro averaged Receiver Operating Characteristic Curve (ROC AUC) score to evaluate the prediction performance.

A ROC curve is a graphical plot that illustrate the prediction ability of a binary classifier. The x-axis of the curve is the false positive rate (FPR) with different threshold. The FPR is defined as

$$\mathrm{FPR} = \frac{fp}{fp+tn} \tag{4.13}$$

where fp is number of positive instances which are incorrectly predicted, and tn is the number of negative instances which are correctly predicted. The y-axis of the curve is the true positive rate (recall) with different threshold.

ROC AUC score is obtained by computing the area under the ROC curve. ROC AUC

store is 1 indicating perfect classifier and between 0 to 1 indicating the classifier is better than random guess.

The definition of macro averaged ROC AUC score is same to macro-F1 and macro-Recall introduced in Section 3.2.4, which calculates the metrics for each label and outputs the unweighted mean.

4.2.8 Model Training

The model is trained by training dataset and validated by the validation dataset, which described in Section 4.1. The focal loss function is optimized by Adam optimization algorithm [48]. Adam combines the advatanges of adaptive gradient algorithm (AdaGrad) and root mean square propagation (RMSProp) which can handle sparse gradients on noisy problem. In the training process, the Adam algorithm is applied with learning rate of 0.1 and mini batch size of 32.

To prevent overfitting and find the best model, the validation-based early stopping technique is used in the training process. At the end of each epoch, the ROC AUC score on validation dataset will be computed and the current weights of the model will be stored. In the experiment, the patience of the early stopping is set to 5. After 5 epochs with no improvement of the ROC AUC score, the training will be stopped. The best model will be selected from the stored model with the highest ROC AUC score.

In addition, except for the embedding layer for text inputs which is initialized with pre-trained word vectors, all the weights are initialized randomly.

4.2.9 Comparison Models

Recently, a variety of the deep learning model have been applied to text classification. We choose four stat-of-the-art and representative models for comparison. These four models have different functional components and mechanisms which are discussed in Section 2.3.

fastText. fastText only uses an embedding layer to extract word vectors and an global average pooling layer to produce sentence-level representation. It is a fast and effective classification model which widely used as a baseline model. The parameters of used in comparison is same to the model described in the original paper [28]. The embedding layer is initialized randomly and the dimension of the word vector is set to 20. The inputs the sequence concatenating the title and text.

Character-level Convolutional Network The char-level CNN is only one comparison model whose functional layer is the convolutional layer. The advantage of the char-level

CNN is that it has very high fault tolerance. As it classify the text in character level, it can easily deal with the typos or the unseen words. The model structure used in comparison is same to what is described in Table 2.1 and Table 2.2. We use the same low-case alphabet presented in Section 2.3.2.

Bidirectional Long-Short Term Memory Network. The BLSTM used in comparison is the model introduced in Section 2.3.3. The size of LSTM unit is set to 256 as suggested by the author. To ensure fair comparison, the embedding layer is initialized with the same pre-trained word vectors proposed by Turian et al. [41]. The model is trained by stochastic gradient descent algorithm.

Attention Based Bidirectional Long Short-Term Memory Network. The Att-BLSTM has same bidirectional RNN layer with BLSTM. In addition, its embedding layer is also initialized with the Turian's pre-trained weights. The difference is that Att-BLSTM is trained by AdaDelta and regularized the model parameters with L2 regularization strength of 10^{-5}.

4.3 Experimental Results

Chapter 5

Conclusions and Future work

5.1 Summary of Contributions

5.2 Future Work

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Bibliography

[1] Indrė Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.

[2] 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.

[3] Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, oct 2017.

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling.

[5] Bo Wu and Haiying Shen. Analyzing and predicting news popularity on twitter. *International Journal of Information Management*, 35(6):702–711, 2015.

[6] Kristina Lerman and Tad Hogg. Using a model of social dynamics to predict popularity of news. In *Proceedings of the 19th international conference on World wide web*, pages 621–630. ACM, 2010.

[7] Georgios Rizos, Symeon Papadopoulos, and Yiannis Kompatsiaris. Predicting news popularity by mining online discussions. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 737–742. International World Wide Web Conferences Steering Committee, 2016.

[8] Dorothy Behling. Fashion change and demographics: A model. *Clothing and Textiles Research Journal*, 4(1):18–24, 1985.

[9] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[10] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[11] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[12] Ali Selamat, Hidekazu Yanagimoto, and Sigeru Omatu. Web news classification using neural networks based on pca. In *SICE-ANNUAL CONFERENCE-*, number 4, pages 2389–2394. SICE; 1999, 2002.

[13] Helen Goworek, Patsy Perry, and Anthony Kent. The relationship between design and marketing in the fashion industry. *Journal of Fashion Marketing and Management*, 20(3), 2016.

[14] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[15] Paulo M. Gonçalves, Silas G.T. de Carvalho Santos, Roberto S.M. Barros, and Davi C.L. Vieira. A comparative study on concept drift detectors. *Expert Systems with Applications*, 41(18):8144–8156, dec 2014.

[16] 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.

[17] Sarah Jane Delany, Pádraig Cunningham, Alexey Tsymbal, and Lorcan Coyle. A case-based technique for tracking concept drift in spam filtering. In *Applications and Innovations in Intelligent Systems XII*, pages 3–16. Springer, 2005.

[18] 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. AcM, 2003.

[19] Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *ICML*, pages 487–494, 2000.

[20] Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[21] 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.

[22] Stephen H Bach and Marcus A Maloof. Paired learners for concept drift. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 23–32. IEEE, 2008.

[23] Gordon J Ross, Niall M Adams, Dimitris K Tasoulis, and David J Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, 33(2):191–198, 2012.

[24] Parinaz Sobhani and Hamid Beigy. New drift detection method for data streams. In *Adaptive and intelligent systems*, pages 88–97. Springer, 2011.

[25] Kyosuke Nishida and Koichiro Yamauchi. Detecting concept drift using statistical testing. In *International conference on discovery science*, pages 264–269. Springer, 2007.

[26] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[27] Hang Li. Deep learning for natural language processing: advantages and challenges. *National Science Review*, 5(1):24–26, sep 2017.

[28] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. 2016.

[29] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[31] John A Bullinaria. Recurrent neural networks. *Neural Computation: Lecture*, 12, 2013.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] Alex Graves. Generating sequences with recurrent neural networks.

[34] Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation.

[35] Yuchen Fan, Yao Qian, Feng-Long Xie, and Frank K Soong. Tts synthesis with bidirectional lstm based recurrent neural networks. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[36] Dongxu Zhang and Dong Wang. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*, 2015.

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space.

[38] David C Plaut and Geoffrey E Hinton. Learning sets of filters using back-propagation. *Computer Speech & Language*, 2(1):35–61, 1987.

[39] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212, 2016.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[41] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[42] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[43] Alper Kursat Uysal and Serkan Gunal. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112, 2014.

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

[45] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[46] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollr. Focal loss for dense object detection.

[47] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015.

[48] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization.