# Prediction of Fashion News Popularity and Retail Companies Share Price

*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 13, 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 computational finance and machine learning tools for identifying investment opportunities in fashion retail assets. The dissertation consists of three key experiments:

### Experiment 1: Celebrity and brand influence on the popularity of the fashion news

The first experiment, which focuses on the collection and the basic analysis of the dataset, sets the background to the following chapters. More specifically, it investigates the correlation between the celebrities and brands mentioned on the fashion news and the popularity of the news on hypebeast.com and hypebae.com. The datasets for this experiment are collected by the self-developed crawler from hypebeast.com and hypebae.com, which are the most influential websites on men's fashion and culture. It contains 182,294 fashion news labeled by their categories, keywords which including the related celebrities and brands, and the index of popularity.

### Experiment 2: Text analysis to predict the popularity of the fashion news

The second experiment use NLP to create a classifier to analyse the news data, predict the popularity, and derive metrics, which could be used as the input for further modeling. This experiment explores various algorithms for text classification for fashion news data. In this experiment, the dataset is performed basic pre-processing and spitted into train and validation sets. The feature engineering is also implemented which transforms the raw data into flat features and also create new features from the existing data. The effectiveness of traditional models are evaluated and the best performing classifier is used in the final model in the following experiment.

### Experiment 3: Stock price movement prediction model

The third experiment proposes a framework to predict the stock price movement using various data sources, including e-commerce data, social media, financial data, and identify investment opportunities. The framework is in the from of a [*the introduction to the final algorithm I use in the model. As I did not start to implement the final model, I will keep it for later.*]. The implementation of the model is made available in the Python programing language.

## Contributions to Science

Due to lack of the understanding of how to analyze the fashion retailing, the investors often overlook the niche financial securities such as stocks of the public fashion companies. The main research contribution of this thesis is to explore new methodology to investment decision making based on different online data sources, such as e-commerce, social network, financial economic ,and fashion news.

This thesis contributes to the existing literature in a number of ways. First, this research proposes a new view to analyze the fashion news and investigates the correlation between the celebrities and fashion trends. Second, the research examines the effectiveness of different algorithm for fashion news popularity prediction and derive the metric for the stock movement prediction model. Third, the final framework is designed combining previous text analysis and other public online data sources, which are capable to undertake pre-investment analysis and recommend investment opportunities.

# 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 [4] developed a model which can predict a news tweets popularity based on the number of its retweets soon after being published. Lerman and Hogg [5] 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.[6] 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 [7]. 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 [8]. 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) [9].

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 [10]. Wu and Shen use the topology reconstruction and inferences of network diffusion to predict the popularity of the twitter news [4]. 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 [11].

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 [12]. 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 [13]. 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 [14]. 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 [15]. Another type of drift is the virtual concept drift which occurs when the target concepts remain the same but the data distribution changes [16]. 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 [17], support vector machine [18], 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.

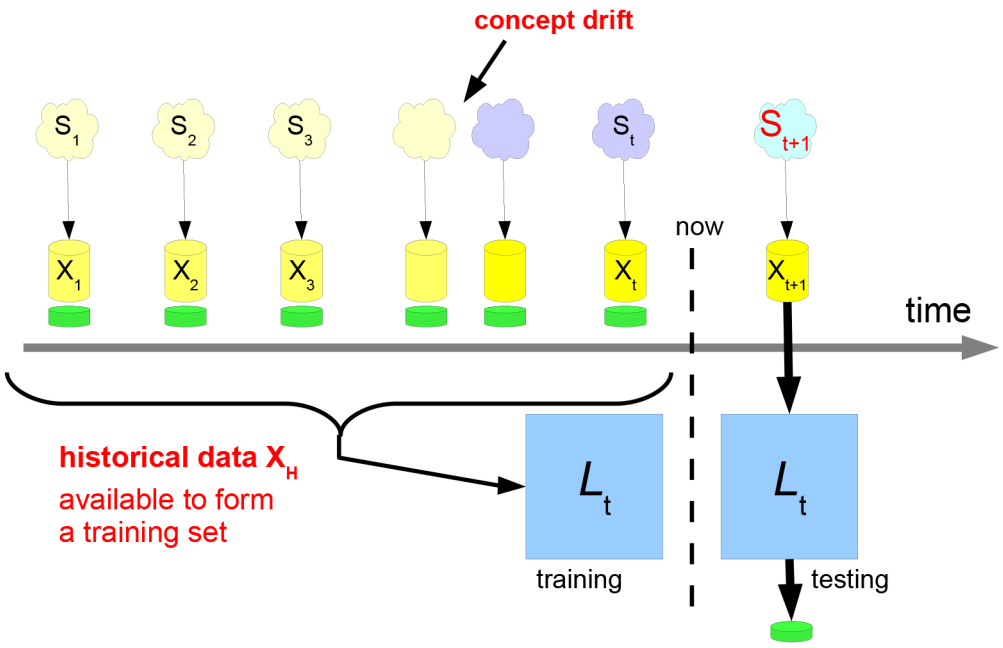
To decrease the training time, most of the concept drift detectors uses incremental learning framework, which show in Figure 2.1. Let $\mathbf{x}_t \in \mathbb{R}^p$ is the feature vector observed at time $t$ and $y_t \in \mathbb{R}$ us the corresponding label. We assume the stream $\mathbf{X}^H = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t)$ 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 [9], EDDM [**?**], PHT [19], ADWIN [20], Paired Learners [21], ECDD [22], DoF [23], and STEPD [24]. 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 [25], 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 $\alpha$ 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 [14].

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 $\alpha$ 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 [14].

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 $\mu_W$ is the average of $\mu_t$ 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 $\varepsilon_{cut}$ 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 $\varepsilon$ 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 [14].

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 $\theta$, 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.

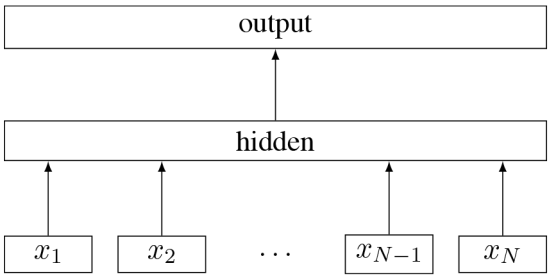
## 2.3 Deep Learning Models in Text Classification

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 [26]. 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 [27][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.

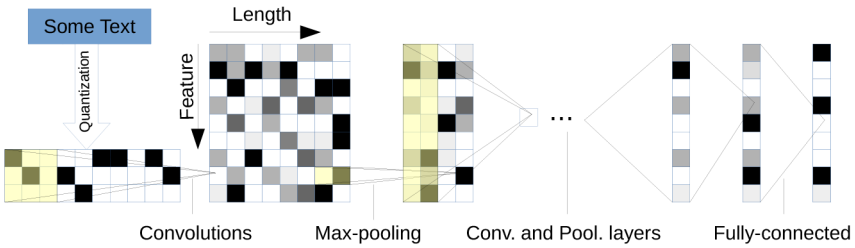
For a large number of classes, fastText uses hierarchical softmax to improve the efficiency in computing the linear classifier. As, in our experiment, the number of classes is so small that hierarchical softmax has no obvious advantages, we will not discuss it in this chapter.

### 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

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

| 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 |

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 [28]. 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

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

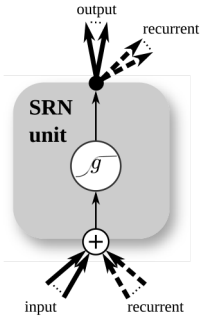
The configurations of the fully connected layers is shown in Table 2.2

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.

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

| Layer | Output units |
|---|---|
| 7 | 1024 |
| 8 | 1024 |
| 9 | Number of classes |

**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.

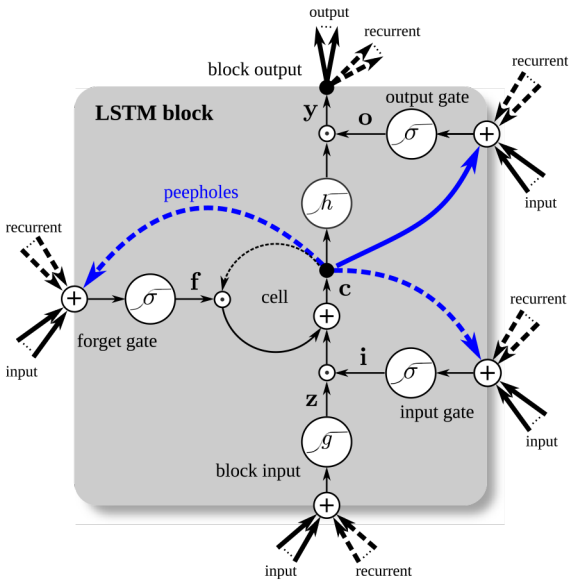
Compared with traditional machine learning method in text classification, such as bag-of-words [13], bag-of-ngrams, bag-of-means on word embedding [29], and some deep learning methods, such as word-level ConvNets, Long-short term memory, Character-level Convolutional Network is an effective method and 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 [30].

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 [31]. 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 $\sigma$, $g$, 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 [32], translation [33], speech synthesis [34].

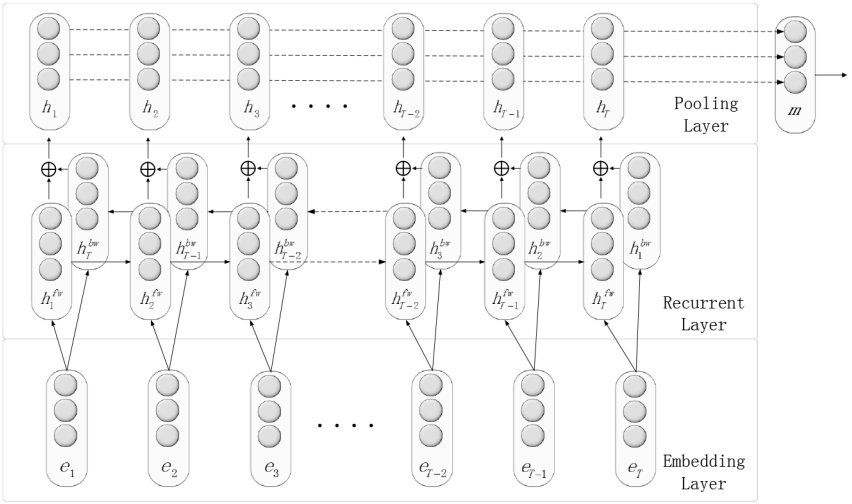
However, the disadvantage of LSTMs is that they ignore the future context as it only read the data in temporal order. To overcome this limit, Bidirectional LSTMs (BLSTM) introduces a second layer where the LSTM units process the sequence in opposite order. Based on this design, the model has the ability 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 [35]. 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 [36]. 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 [37].

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.

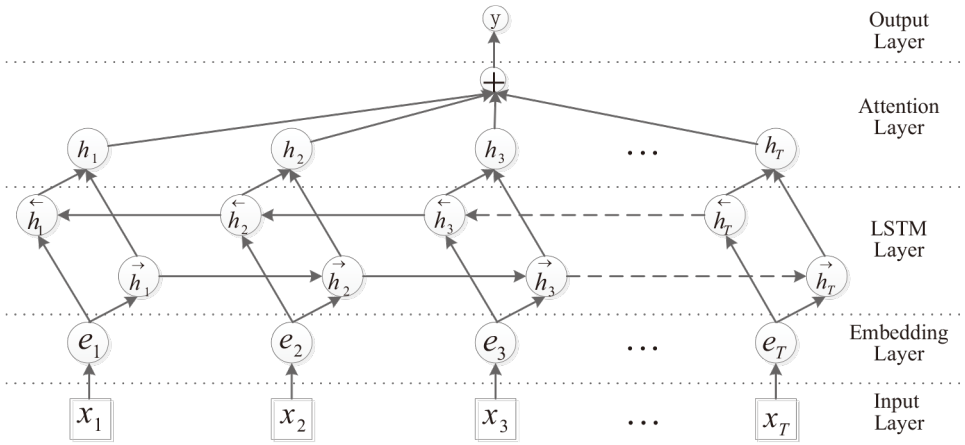
### 2.3.4 Attention Based Bidirectional Long Short-Term Memory Network

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 [38]. 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 $\lambda$ 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[39]. 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 [40]. Other weights are initialized randomly. Different with aforementioned models which trained using stochastic gradient descent, Att-BLSTM is trained using AdaDelta [41] 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 [42].

## 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

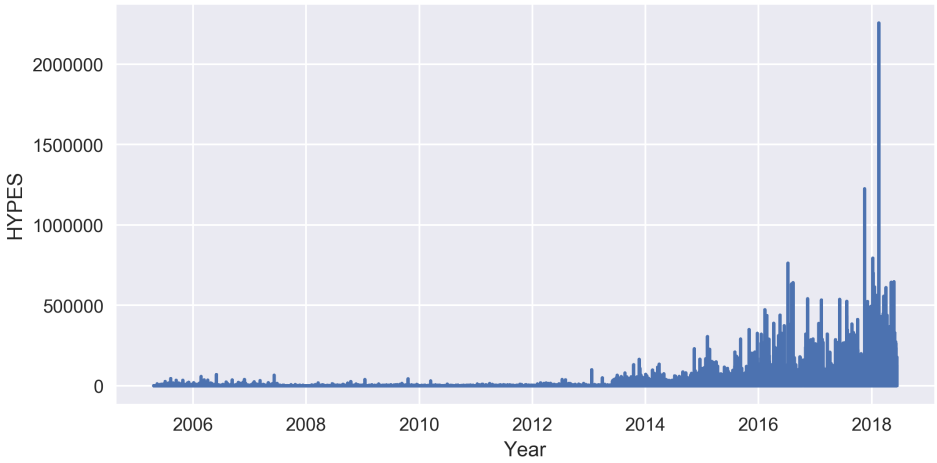



**Figure 3.1:** HYPES over years

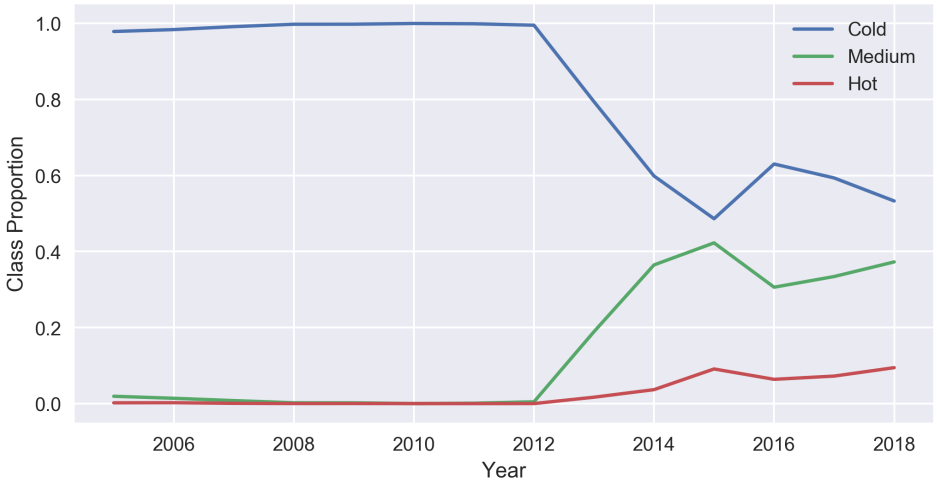



**Figure 3.2:** Proportion of each class over years

## 3.2 Model Design

## 3.3 Result

Chapter 4

# Text Analysis to Predict the Fashion News Popularity

## 4.1 Background

### 4.1.1 Motivation

### 4.1.2 Related Work

## 4.2 Experiment

### 4.2.1 Dataset

### 4.2.2 Experiment Design

### 4.2.3 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.

## Appendix A

# An Appendix About Stuff

(stuff)

## Appendix B

# Another Appendix About Things

(things)

## Appendix C

# Colophon

*This is a description of the tools you used to make your thesis. It helps people make future documents, reminds you, and looks good.*

*(example)* This document was set in the Times Roman typeface using LaTeX and BibTeX, composed with a text editor.

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

[5] 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.

[6] 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.

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

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.

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

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

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

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

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

[28] 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.

[29] 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.

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

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

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

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

[34] 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.

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

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

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

[38] 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.

[39] 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.

[40] 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.

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

[42] Dongxu Zhang and Dong Wang. Relation classification via recurrent neural network.