

AN ADVANCED FRAMEWORK TO OPTIMIZE MULTI-RESIDUAL RECURRENT
NEURAL NETWORK FOR BETTER SEQUENCE LEARNING

A Dissertation

by

YE WANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Chair of Committee, Mi Lu
Committee Members, Yoonsuck Choe
Jiang Hu
Xiuquan Ji
Head of Department, Miroslav M. Begovic

December 2019

Major Subject: Computer Engineering

Copyright 2019 Ye Wang

ABSTRACT

Sequence Learning is the cornerstone of data mining, and is significant in extracting useful information, from sequencing sounds in a speech to sequencing semantics in linguistics.

Before sequence learning, finding a proper approach to collect sequential data is essential. The main purpose of the proposed recognition system is to defeat the CAPTCHA (Completely Automated Public Turing test to Tell Computers and Humans Apart) because we need to collect the data for the sequence learning. Besides, defeating the CAPTCHAs is also beneficial to improving the safety when we expose the CAPTCHAs' deficiency. As an effective way to protect the security and preserve the privacy of the network data, CAPTCHA is widely used in recent years. Normally, three steps are utilized to defeat the CHAPCHAs - Preprocessing, Segmentation and Recognition. Since there is not a universal segmentation framework that is adaptive to all the possible CAPTCHA characters, each individual character requires separate segmentation which makes the segmentation complicated. In this dissertation, we present a self-adaptive algorithm in optimally segmenting different CAPTCHA characters. Current classifiers including Template Matching (TM), Optical Character Recognition (OCR) and Convolutional Neural Networks (CNN) are utilized in classifying these segmented CAPTCHA characters. The CAPTCHAs experimental results show the outperformance of the proposed recognition system in defeating the CAPTCHA.

In the currently existing financial related Chinese text classification task, the data quality of those tasks is not ideal because labeled Chinese datasets are not large enough. Besides the text-based CAPTCHA recognition, short-term text classification also plays an important role in sequence learning. After obtaining the titles of Chinese commercial news, a new Chinese financial related Short-term Text Classification Task (STCT) is introduced and its corresponding benchmark is provided.

As a popular solution for STCT in sequence learning, recurrent neural networks (RNNs) have proven its efficiency in processing sequential information. However, the traditional RNNs have suffered from the gradient diminishing problem until the advent of Long Short-Term Memory

(LSTM). The LSTM, though, is still weak in capturing long-time dependency in sequential data due to the inadequacy of memory capacity in LSTM cells. To address this challenge, we propose an Attention-augmentation Bidirectional Multi-residual Recurrent Neural Network (ABMRNN) to overcome this deficiency. The proposed ABMRNN integrates both past and future information at every time step with an omniscient attention model. The multi-residual mechanism has also been proposed in our model targeting the pattern of the relationship between the current time step and further distant time steps instead of only one previous time step. The experimental results show that the proposed model outperforms the traditional statistical classifiers and other state-of-the-art variations of RNN architectures in both the STCT and other public tasks, such as AG news, Sequential-MNIST, and IMDB.

DEDICATION

To my dear family and all the important people in my life.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Professor Mi Lu, for her massive guidance, encouragement and patience. Her splendid experience in terms of life and academic work has inspired me significantly. Without her constant support and timely advice this dissertation would not have been possible. I feel grateful to be her student and work with her over the six years.

I would like to thank Professor Yoonsuck Choe not only for his machine learning and neural network courses which guide me around this area, but also for his help with my research.

I would like to thank Professor Jiang Hu and Jim Ji for their insightful comments and feedback on my work.

I also feel thankful to all my friends for the support during my graduate life. I would like to thank Yue Wu, Dan Tian, Xinxiang Zhang, Han Wang, Yingyezhe Jin, Zhengcong Yin, Tiben Che, Tian Lan, Zhiyuan Zheng and Binghan Li for their wonderful friendship that makes my life in the United States amazing.

Finally, I would like to thank my parents, my sister and my girlfriend for their unconditional love, support and encouragement that have enriched and enlightened my life.

CONTRIBUTORS AND FUNDING SOURCES

This work was supported by a dissertation committee consisting of Professors Mi Lu, Xiuquan Ji and Jiang Hu of the Department of Electrical & Computer Engineering; and Professor Yoonsuck Choe of the Department of Computer Science & Engineering.

The CAPTCHA part of data collection for Section 2 was received with the help during the intern. All other work including RNN design and benchmark conducted for the dissertation was completed by the student.

NOMENCLATURE

ABMRNN	Attention-augmentation Bidirectional Multi-residual Recurrent Neural Network
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
DT	Decision Tree
GNB	Gaussian Naive Bayes
KNN	K-Nearest Neighbor
LR	Logistic Regression
LSTM	Long Short-Term Memory
MNB	Multinomial Naive Bayes
NB	Naive Bayes
NLP	Natural Language Processing
NN	Neural Network
OCR	Optical Character Recognition
OOV	Out-Of-Vocabulary
PN	Proper Names
RNN	Recurrent Neural Network
STCT	Short-term Text Classification Task
SVC	Support Vector Classifier
SVM	Support Vector Machine

TABLE OF CONTENTS

	Page
ABSTRACT	ii
DEDICATION	iv
ACKNOWLEDGEMENTS	v
CONTRIBUTORS AND FUNDING SOURCES	vi
NOMENCLATURE	vii
TABLE OF CONTENTS	viii
LIST OF FIGURES	x
LIST OF TABLES	xii
1. INTRODUCTION.....	1
2. AN OPTIMIZED SYSTEM TO SOLVE TEXT-BASED CAPTCHA.....	6
2.1 CAPTCHA overview	6
2.2 State-of-the-art CAPTCHA algorithms	8
2.2.1 Data processing	10
2.2.1.1 Thresholding.....	10
2.2.1.2 Median filter	11
2.2.1.3 Affine transformation	11
2.2.2 Segmentation	12
2.2.3 Recognition	13
2.3 Proposed scheme	13
2.3.1 Denoise filter.....	13
2.3.2 Adaptive algorithm	14
2.3.3 Convolutional neural network construction	21
2.3.3.1 The Architecture.....	21
2.3.3.2 Pooling layer.....	22
2.3.3.3 Hyper-parameters	22
2.3.3.4 Datasets	24
2.4 Summary	24
3. COMPARISONS AND SELECTIONS OF FEATURES AND CLASSIFIERS FOR SHORT TEXT CLASSIFICATION.....	29

3.1	Task motivation	29
3.2	General approach description.....	30
3.2.1	Data processing	30
3.2.1.1	Segmentation	31
3.2.1.2	Feature selection.....	31
3.2.2	Different types of classifiers	33
3.2.2.1	Naive bayes	34
3.2.2.2	Decision tree.....	34
3.2.2.3	Logistic regression	35
3.2.2.4	Support vector machine	35
3.3	Specific system implementations	35
3.4	Summary	35
4.	ATTENTION-AUGMENTATION WITH MULTI-RESIDUAL IN BIDIRECTIONAL LSTM	40
4.1	Background.....	40
4.2	Recurrent neural networks preliminaries.....	43
4.2.1	Long short-term memory (LSTM).....	43
4.2.2	Recurrent residual network	44
4.3	Proposed scheme	45
4.3.1	Attention-augmentation mechanism.....	46
4.3.2	Multi-residual LSTM.....	47
4.3.3	Bidirectional multi-residual network	49
4.3.4	Training procedure	50
4.4	Summary	51
5.	CONCLUSION.....	57
	REFERENCES	59

LIST OF FIGURES

FIGURE	Page
2.1 CAPTCHA applications	7
2.2 Online samples in the datasets	7
2.3 CAPTCHA datasets	9
2.4 Basic flow chart to defeat the CAPTCHA	9
2.5 A classic example of affine transformation	11
2.6 The example of adaptive segmentation	15
2.7 The detailed flow chart of adaptive segmentation	16
2.8 Traditional flow chart of segmentation	20
2.9 Sequentially defeating the CAPTCHA	21
2.10 Neural network structure	23
2.11 Rotate the image to diversify training data	24
2.12 The full process of defeating the CAPTCHA	26
2.13 The accuracy rate for each CAPTCHA	27
2.14 Overall CAPTCHA accuracy rate	27
2.15 Accuracy rate of individual character	28
3.1 Announcement examples in dataset	30
3.2 Basic flow chart	30
3.3 Processing the data	31
3.4 Two models of word2vec	32
3.5 Paragraph2vec model	33
3.6 Result	36

3.7	1/2 datasets of overall	37
3.8	1/2 datasets of word2vec	38
3.9	1/2 datasets of doc2vec	38
3.10	1/2 datasets of tf-idf/counter	39
4.1	LSTM cell	44
4.2	Residual recurrent network	45
4.3	Attention model	47
4.4	Multi-residual LSTM with attention Model, unrolling our model along the time axis. The dashed box indicates the updated state in current time step.	47
4.5	Bidirectional multi-residual LSTM with attention model	48
4.6	Error rate.....	53
4.7	Training loss.....	54

LIST OF TABLES

TABLE	Page
2.1 All possible cases	17
4.1 Classification datasets	51
4.2 Accuracy in classification results	52
4.3 The number of parameters	56

1. INTRODUCTION

Sequence Learning (SL) is the cornerstone of data mining, which is significant to extract useful information. In particular, sequence learning plays a vital component of learning in numerous task domains - natural language processing, sequencing sounds in speech, sequencing actions in driving an automobile and so on.

Nowadays, plentiful approaches towards sequence learning have been proposed, resulting from different perspectives taken in disparate task domains [1]. Sequence learning is not a simple task because of these aforementioned different domains. To find more advanced algorithms, training procedures, and theories are to comprehensively understand the sequence and achieve state-of-the-art performance in given topics. To develop better and more powerful algorithms, it is necessary to compare, contrast, and combine different existing methods, approaches and paradigms.

Before learning the sequence, the first part is fetching the data. Although tons of data exist among the internet, how to collect the data we need for the sequence learning properly is essential. CAPTCHA (Completely Automated Public Turing test to Tell Computers and Humans Apart) is widely used to protect data from auto bots. Furthermode, defeating the CAPTCHAs is also beneficial to improving safety when we expose the CAPTCHAs' deficiency. CAPTCHA designers change the combination of coloring numbers and characters and so on which can be recognized by people but not the automated bots. Besides, both companies and individuals apply text-based CAPTCHAs most frequently because of the convenience.

CAPTCHA is used to prevent the auto bots, but it is impossible to manually collect the data we need for sequence learning by human beings. Therefore, we have to defeat text-based CAPTCHA automatically first. Three steps are normally needed: preprocessing by denoising, segmentation to get individual characters and recognition to identify each character. Those three steps are treated as equally important. Regarding the preprocessing step, since there may be a lot of noise affecting the performance, we must decrease the effect to obtain clear images for higher performance. Nowadays, a lot of methods are proposed to achieve the goal, like certain image processing ways

and machine learning algorithms such as median filter [2], neighborhood filter, wavelet threshold, universal denoising, K-nearest neighbors algorithm, support vector machine and so on. However, only when the appropriate denoising method is selected, we can acquire the clearest output [3][4].

After preprocessing, we should segment the image into characters individually. This is because if the image can be divided perfectly, it will be helpful for the next step about the recognition accuracy. As we mentioned before, the CAPTCHAs' images are not always the same, thus defeating the CAPTCHAs itself heavily depends on the detailed weakness. After numerous methods about segmentation experiments, image intensity histogram and color clustering [5] are the two most effective ways. Last we present our novel adaptive algorithm to optimize the segmentation in defeating the CAPTCHAs which will be further discussed.

Recognition is the last step to get the output of defeating the CAPTCHAs. Through a lot of explorations, we conclude three fast and reliable recognition ways, they are Optical Character Recognition(OCR), Template Matching(TM) and convolutional neural network(CNN). First of all, OCR is used to convert words in pictures to the formal printed text [6]. Actually, OCR has been applied in a wide range of areas such as documents like contracts, passport, receipts, even business and credit cards. Besides, many commercial applications also have been developed for the task of identifying machine typed text. While OCR has failed to defeat the CAPTCHAs due to various reasons. Some commercial OCR methods can only work well on black-and-white images, part of them are even based on free-noise text. However, the majority of the CAPTCHAs we need to deal with in this paper are against free-noise text, they are always randomly noised, colored and even rotated [3][7]. Moreover, the examples of our datasets suffer from various fonts, directions, and even other distortions, so that the combination of those existing problems can not be easily solved by just employing OCR.

To overcome OCR's shortage, we would like to introduce TM. Long long ago, before OCR, this is the most original machine learning way to perform recognition [8]. TM is straightforward and easy to implement. Specifically, when we defeat some CAPTCHAs with rotated examples, TM can achieve a higher accuracy rate than OCR. With the increasingly complicated circumstances

designed, in terms of heavily rotated, overlapped and twisted, CNN overwhelmingly works better than any others in terms of accuracy and time complexity [9]. However, constructing a CNN is also huge, which needs to pre-train numerous well-prepared samples [10]. Our own CNN conquers this problem with a lot of manual efforts aiming at data collection. We finally obtain state-of-the-art performance to defeat the CAPTCHAs.

After defeating the CAPTCHAs and obtaining the data, the task has been proposed for learning the sequence. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are the two primary architectures in neural networks for sequence learning. RNNs are often applied to sequential data such as natural language processing and speech processing [11, 12], while CNNs are more employed in image processing areas [13, 7, 14]. Among the existing RNN models, LSTM [15] is one of the most popular approaches since it initially solves gradient vanishing and exploding problems during RNN training by introducing forget gates and memory cells. After the literature review, we found that numerous RNN variations have been proposed to achieve state-of-the-art performance in different tasks, where LSTM is the cornerstone of those structures. However, due to the limited memory cell in LSTM, when the time sequence is long, the LSTM performance is heavily influenced.

With the increase of the depth of layers and the length of the sequences, residual networks have proved their advantages in both CNNs [16, 17] and RNNs [18]. Residual networks connect current and distant previous time steps for optimizing of the layer information. [16] and [19] propose similar residual ideas to randomly connect one previous distant time step to the current time step, where the problem of long time dependency is solved partially. Therefore, the residual networks motivate us to combine the residual network with LSTM, where the information in current time step has been updated dynamically based on the attained correlation between previous time steps and current time step.

To better attain the correlation between current time step and previous time steps, the attention model is widely applied in image processing, speech processing, and natural language processing. The objective of the attention model is to ultimately optimize the training procedure when the

amount of attention is limited. [20] initially leverages the attention model from image processing to natural language processing. [21] proposes a model as a decoder network between previous states and current state. [22] simplifies the model as an attention-based weighted pooling RNN to acquire utterance representation in speech processing. Since the attention is limited, the way to effectively distribute those attention becomes considerably important. Inspired by the attention-based approaches, we leverage the attention model [23] to strengthen the correlation between the current state and both previous and future distant states. As for [23], they focus on the relationship between the current time step with previous information. Since the objective is to allocate the attention properly, we regard the past and the future as the same importance, which means we integrate both the previous and the future time steps to refine the information of current time step instead of only relying on previous time steps.

Therefore, to address the aforementioned challenges in long time dependency and optimizing the text correlations, this paper develops an Attention-augmentation Bidirectional Multi-residual Recurrent Neural Network (ABMRNN). The proposed ABMRNN achieves state-of-the-art performance among several existing sequential classification tasks. The main contributions of this work are summarized as follows:

- Our algorithm overcomes the deficiency of LSTM in weak modeling the long-time dependency, so we can handle much longer sequential data and obtain higher accuracy rate in longer sequential tasks. In this algorithm, we design a novel bi-directional layer to dynamically acquire and allocate attention from both previous and future time steps. Bi-directional layers help us focus not only on the past but also in the future so that we can attain a better correlation between current steps and distant time steps.
- To better supplement the acquired attention from bi-directional layer, we also leverage the multi-residual mechanism to recurrent networks. Compared with traditional residual networks, the advantages of ours are more obvious because the proposed multi-residual mechanism is more rational than previous residual networks in sequence learning since the traditional residual networks connect current time step with only one randomly previous time

step.

- The proposed model contains fewer parameters than current popular models such as [16] and [24], which indicates that the architecture of the proposed model is less complicated than those popular models. The proposed model also achieves the state-of-the-art performance in sequence learning of STCT.

2. AN OPTIMIZED SYSTEM TO SOLVE TEXT-BASED CAPTCHA ¹

2.1 CAPTCHA overview

Before learning the sequence, the first part is fetching the data. Although tons of data exist among the internet, it is essential to collect the proper data for sequence learning. CAPTCHA (Completely Automated Public Turing test to Tell Computers and Humans Apart) is widely used to protect data from auto bots as shows in Figure 2.1. Furthermore, defeating the CAPTCHAs is also beneficial to improving safety when we expose the CAPTCHAs' deficiency. CAPTCHA designers change the combination of coloring numbers and characters and so on which can be recognized by people but not the automated bots. Besides, both companies and individuals apply text-based CAPTCHAs most frequently because of the convenience.

We rely on CAPTCHA increasingly heavily in distinguishing between human beings and computer programs automatically. Because of the combination of distorting characters and obfuscation techniques which can be recognized by people but may be hard for automated bots [25][26], and text-based CAPTCHAs are most widely used by both companies and individuals.

We divide text-based CAPTCHA defeating into three parts: denoising, segmentation and recognition. They are equally vital, regarding preprocessing step. The purpose is to decrease the noise influence for ensuring the correct segmentation which can increase the recognition accuracy rate. Actually, there exist a lot of methods, including the combination of image processing and artificial intelligence algorithms, like median filter [2], neighborhood filter, wavelet threshold, universal denies, K-nearest neighbors algorithm, support vector machine and so on. However, how to choose those candidate ways is the key to perform the right denoising [3][4]. Actually, if we could directly utilize OCR, segmentation is not an essential step. While after preprocessing, most of our

¹Part of this section is reprinted with permission from (1) "A self-adaptive algorithm to defeat text-based CAPTCHA" by Ye Wang and Mi Lu, 2016. *Proceedings of the 2016 IEEE International Conference on Industrial Technology (ICIT)*, Page 720-725, ©2016 IEEE. (2) "Combining convolutional neural network and self-adaptive algorithm to defeat synthetic multi-digit text-based CAPTCHA" by Ye Wang, Yuanjiang Huang, Wu Zheng, Zhi Zhou, Debin Liu and Mi Lu, 2017. *Proceedings of the 2017 IEEE International Conference on Industrial Technology (ICIT)*, Page 980-985, ©2017 IEEE. (3) "An optimized system to solve text-based CAPTCHA" by Ye Wang and Mi Lu, 2018. *arXiv preprint arXiv:1806.07202*, ©2018 arXiv.org.



Figure 2.1: CAPTCHA applications



Figure 2.2: Online samples in the datasets

CAPTCHAs cannot be recognized by OCR. Thus, we should use the segmentation to divide the image into characters for better performance. Towards the segmentation, it heavily depends on the feature of the images. Image intensity histogram and color clustering are the most frequently used techniques [5]. For some special CAPTCHAs to be further discussed later, we present our novel adaptive length of characters to implement the segmentation. Recognition is the last step to get outputs. First of all, Optical Character Recognition (OCR) is the way to convert words in pictures or other language related in images to the typed text [6]. It is widely used as a form of data entry from the printed records, whether documents like bank statements, contracts, receipts, even business or credit cards and so on. Currently, the task to identify machine typed text has

already been absolutely solved. Many business commercial applications are now in the markets but the methods failed to CAPTCHAs due to various reasons. Some applications work well on black-and-white images, some based on free-noise text, while the majority of the CAPTCHAs we will discuss in this section are against the rules - they are randomly noised, colored and even rotated [3][4]. Moreover, our CAPTCHAs images suffers from inconsistent lighting conditions, and different fonts, directions, and other distortions. The combination of those complicated problems can not be easily solved by employing only OCR. The second potential way is Template Matching, before OCR, this is the most original artificial intelligence method to execute recognition [8]. The review of TM is considerably straightforward and convenient. At the mean time, for some rotated CAPTCHAs, it can get kind of higher accuracy rate than OCR. We need to figure out that with the development of increasingly complex circumstances, including heavily rotation, overlapping and twisted, CNN (convolutional neural network) overwhelmingly works better than any others in terms of accuracy and time complexity [9]. Every coin has two sides, so does the CNN. It needs to be pre-trained with a lot of well prepared samples, which is the essential contribution part of this CNN [10]. Our CNN conquers this problem with a lot of human being efforts aiming at data collection. So that, we finally achieve state-of-the-art performance to defeat the CAPTCHAs.

The rest of this section is organized as follows: Section 2.2 describes the background on the three steps in terms of denoising, segmentation and recognition. Section 2.3 proposes the scheme of adaptive segmentation and how we construct our convolutional neural network. Section IV shows the experiment. The section concludes in Section 2.4.

2.2 State-of-the-art CAPTCHA algorithms

CAPTCHAs can be found almost in every websites [26][27][28]. Each type of CAPTCHA should be solved accordingly because of the unique encoding algorithms. Correspondingly, a lot of deCAPTCHA algorithms have been proposed for most famous companies such as Google, Microsoft, and Facebook. A low-cost attack on a Microsoft CAPTCHA was presented for solving the segmentation resistant task, which achieved a segmentation success rate of higher than 90% [29]. Moreover, another projection-based segmentation algorithm for breaking MSN and YAHOO

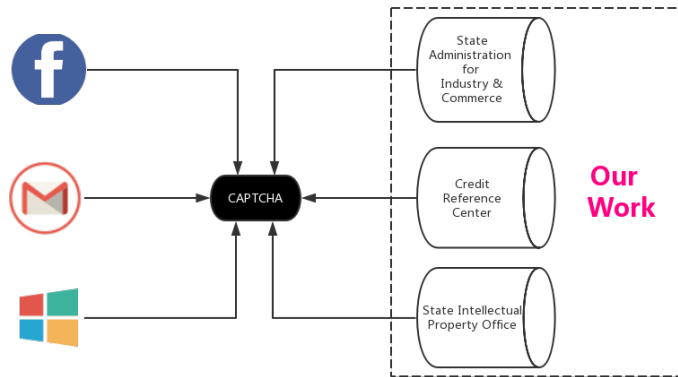


Figure 2.3: CAPTCHA datasets

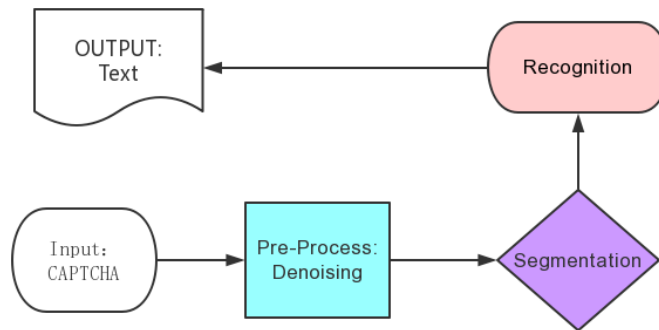


Figure 2.4: Basic flow chart to defeat the CAPTCHA

CAPTCHAs proves to be effective, which doubled the corrected segmentation rate over the traditional method [30]. Besides, an algorithm using ellipse-shaped blobs detection for breaking Facebook CAPTCHA also presents to be useful [31].

As can be seen, Figure 2.2 presents some examples of text-based CAPTCHAs which will be discussed later. Figure 2.3 describes the system of our datasets. Figure 2.4 shows the basic flow chart to defeat the CAPTCHAs. In this section, we will demonstrate the whole process.

2.2.1 Data processing

There are several possible techniques to denoise the images based on various types of each other. After our efforts, we find out the appropriate ways according to the different types of examples. Median filter is effective for nonlinear digital method to filter the noise. Affine transformation plays an equivalently important role in preprocessing. Actually affine transformation does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line [32]. After an affine transformation, the sets of parallel lines can still remain parallel, which also preserve points, lines and planes.

2.2.1.1 Thresholding

In an image, the main difficulty of finding the optimal intensity of the whole image between real image and noise is thresholding, because there are not any relationships between the pixels. No one can guarantee that the pixels between each others identified by the thresholding process are contiguous. We can easily include irrelative pixels that aren't part of the desired region in real image we actually need, and we can easily miss isolated pixels within the region as well (especially near the boundaries of the region). These effects get increasingly worse as the type of noise becomes more and more complicated, simply because it's more likely that a pixel intensity cannot represent free-noise image intensity in the region [33]. When we use thresholding method, we typically have to balance with the tradeoff in terms of losing too much of the region informations and getting too many background pixels with noise. (Shadows of objects in the image are also a real pain - not just where they fall across another object but where they mistakenly get included as part of a dark object on a light background.)

Right here, we utilize the Automated Methods for Finding Thresholds: To set a global threshold or to adapt a local threshold to an area, we usually look at the histogram to see if we can find two or more distinct modes, one for the foreground and one for the background [34].

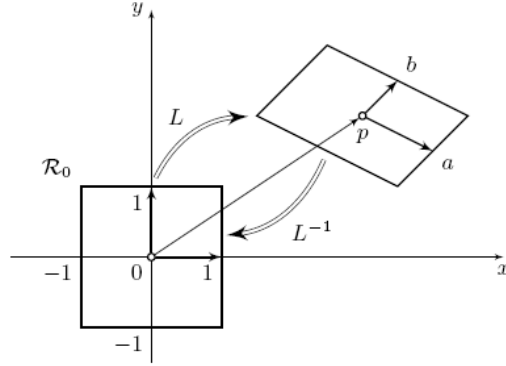


Figure 2.5: A classic example of affine transformation

2.2.1.2 Median filter

Median filter [2] is effective for nonlinear digital signal to remove noise as well as preserve edges while removing noises. Noise reduction is a typical pre-processing step to improve the final results of later processing (for example, edge detection on an image). The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries. For example, for every window slides, $y[1] = \text{Median}[2 \ 2 \ 80] = 2$.

2.2.1.3 Affine transformation

In geometry, an affine transformation, affine map or an affinity is a function between affine spaces which preserves points, straight lines and planes. An affine transformation may not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line [32].

An affine map is made up of two functions: a translation and a linear map. The ordinary vector algebra represents linear maps by matrix multiplication, and represents translations by vector addition. If the linear map is expressed as a multiplication by a matrix A and the translation as the addition of a vector \vec{b} , an affine map f acting on a vector \vec{x} can be represented as

$$\vec{y} = f(\vec{x}) = A\vec{x} + \vec{b} \quad (2.1)$$

In Figure 2.5, we can easily understand the effect of affine transformation. In short, we need to retain the ratio among graphics while the angles cannot be guaranteed.

2.2.2 Segmentation

Segmentation is quite important, which can heavily influence the performance. The most common way for analyzing segmentation is to divide the CAPTCHAs' image into multiple single parts [5][14]. Segmentation is used to detect the examples of CAPTCHAs' boundaries like lines or curves, with consideration of the rotation, noise and even twisted characters.

Histogram-based and K-means clustering are two effective methods in segmentation. Histogram-based is to count the quantity of pixels in each row or column in grey level. The basic algorithm would be illustrated as y_0, y_1, \dots, y_n , where y_i is the number of pixels in the image with gray-level i , and n is the maximum gray-level attained. Imagine that if the distance in histogram between the characters is very far, it would be easy to separate the characters [35]. While K-means clustering is another method for segmentation, segmentation is a very dependent method since the CAPTCHAs vary considerably. So far, there aren't any useful algorithms to segment the affixed, bended, even twisted characters. There are several generations of K-means methods. Following is the basic pseudocode we would need to exploit.

Algorithm 1: K-Means Algorithm

1. *Select* K points as the initial centroids
 2. *Repeat*
 - Form* K clusters by assigning all points to the closest centroid
 - Recompute* the centroid of each cluster
 3. *Until* The centroids don't change
-

In the next section, we will focus on an algorithm which contains a lot of CAPTCHAs by using adaptive length segmentation.

2.2.3 Recognition

The last step is to recognize the characters automatically, getting printed message to fulfill the whole process of defeating the CAPTCHAs. We implement the recognition in three ways, Optical character recognition (OCR), Template Matching (TM) and Convolutional Neural Network (CNN). For OCR, we choose an open source software named Tesseract by Google [6] [36]. If the format is rigid, the recognition accuracy performance would be the best. While not all the CAPTCHAs can be similar, for some irregular cases, TM is an option, which is a pattern-oriented method to find the most similar candidate characters. The theory is to slide the template image over the input image, then get the similarity matrix of each other to obtain the best candidate.

CNN is a more complicated artificial intelligent technique compared with OCR and TM [37][9], since it needs much more pre-defined data to train the classifier for a system in terms of higher accuracy rate and less time complexity. This is because the more data we can use, the better classifier we would get. The limitation of CNN is also apparent in terms of the size of datasets, since we need to manually denoise the image first and then segment them into individual characters, thus the working procedure is comparably complicated than the others.

2.3 Proposed scheme

2.3.1 Denoise filter

There are many types of CAPTCHA examples and each type of them is not the same as others. Generally, we need to first denoise the examples for clear images as we mentioned before. There are two major denoise methods regarding image processing, one is to carry on in frequency domain and the other is to process in time domain. In the frequency domain, we usually transfer the image from time series to frequency series, utilizing a corresponding transformation such as Fourier Transformation (FT) which can reflect image characteristics to denoise the image. In FT spectrum, the noise always exists around high frequency areas while the image entity itself exists in low frequency areas. So some particular low pass filters can be adopted to eliminate the noise and to retain the filtered real image. Normally, the filters are divided into linear and non-linear one for

denoising. Actually, linear filters such as mean filter and gaussian filter can be considered as a result of raw data and the filters in arithmetic computations like addition, subtraction, multiplication and division. Due to the limitation of arithmetic computation in linear filters, the transformation function is determinate and unique.

In the time domain, compared with the linear filter, the process of non-linear filtering can be viewed as passing raw data through specific filters such as minimum filter, maximum filter and median filter. As to the implementation, nonlinear filters act like to slide a window to make the decision about minimum, maximum or median value within the window frame. Unlike linear filters, this operation is nondeterminate because of the logic relationship in window sliding. What's more, compared with linear filters, the function of nonlinear filters is not limited to protecting the margins but also removing the noise much better. The speed in non-linear filters would be slower than that in linear filters, because we have to slide the window over the whole image. Median filter is a typical nonlinear technique. The basic idea is to get the median value of a pixel in grey level within the window. Since the noise pixels are always higher or lower than the image entity itself, this method is significantly effective in denoising and margin protection.

2.3.2 Adaptive algorithm

There are various kinds of similar notations defeating CAPTCHAs. As an example, we investigated the popular query involving the following triplet structure, < first operand, operator, second operand >.

One may need to know that the length of operator is variant here. Also, the operands can be of different fonts, like traditional Chinese characters, simplified Chinese characters and Arabic numbers. Furthermore, the font of the operator is orthogonal to that of the operands. There are three kinds of operators, symbol operator, single character and double characters operators, each of them also includes several types of operations. The ones given in Figure 2.6 show the challenge involving Chinese characters. For example, in the first row, it asked you to answer the question: 9 + 5 = ?. The first operand and second operand are traditional Chinese characters, and the operator is made up of two characters. While in the third row, there is a symbol operator with Arabic number

玖加上伍等于几	Traditional Chinese operands with double Chinese characters operator
肆+伍等于几	Traditional Chinese operands with symbolic operator
9-5等于几	Arabic numeric operands with symbolic operator
九乘以八等于几	Simplified Chinese operands with double Chinese characters operator
肆乘陆等于几	Traditional Chinese operands with single Chinese character operator
9加上8等于几	Arabic numeric operands with double Chinese characters operator
九-八等于几	Simplified Chinese operands with symbolic operator
0加4等于几	Arabic numeric operands with single Chinese character operator

Figure 2.6: The example of adaptive segmentation

operands which means $9 - 5 = ?$.

Let's start with the operand first. The first operand can be made up with any digits, so can the second operand. We assume that the numbers of the first operands, the second operands and the operator are M , N and O respectively. Since they are pairwise orthogonal, the total number of combinations of those notations can be up to $M * N * O$ which is very massive. Fortunately, in our cases, some rules can be applied to reduce the number of possible combinations.

For example, given one operand with values zero to nine to represent: there are three fonts like traditional Chinese characters, simplified Chinese characters and Arabic numbers, resulting in $10 * 3 = 30$ potential choices in total.

Moreover, the operators representing addition, subtraction, multiplication or division can be of three fonts, symbolic operators, single Chinese characters and double Chinese characters. The total number of operator candidates is $4 + 4 + 4 = 12$. With the two operands and one operator taken into consideration, the total number of combinations is $30 * 30 * 12 = 10800$.

As the last but not the least important issue, Table 2.1 shows all the possible combinations in our cases. S represents simplified Chinese, T is the traditional Chinese, and D stands for Arabic

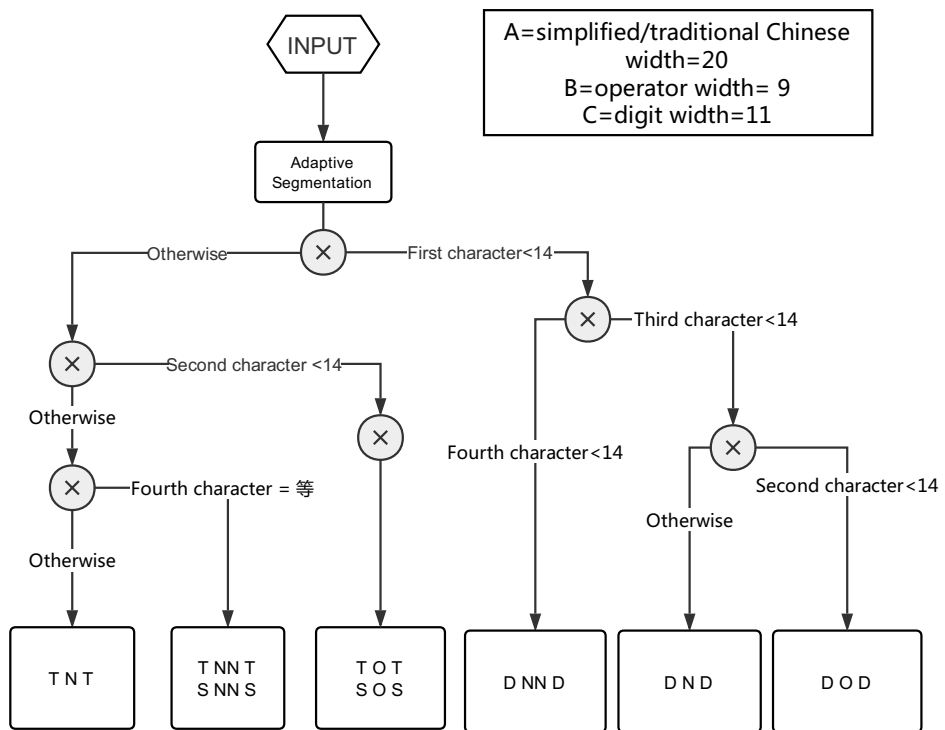


Figure 2.7: The detailed flow chart of adaptive segmentation

numbers. Regarding the operator, we use O for symbolic operator, and N and NN for the single and double Chinese characters symbols respectively. For all the possible combinations, please refer to Table 2.1. We divide all the combinations into three types based on the different fonts of operands. The simplified Chinese operands belong to Type 1, traditional Chinese operands belong to Type 2 and Arabic numeric operands belong to Type 3 respectively.

Since the operators needs be paired with different kinds of operands, the combinations of operand and operator examples are enumerated in Table 2.1. Nine groups among the operands and operators in total are shown in Table 2.1. For example, single character operator can be paired with traditional Chinese operands, simplified Chinese operands and Arabic numbers. One thing needs to be mentioned here. According to the shortage of our datasets, one type of notation is missing, SNS in the second row of Type 1. So we have, actually, only eight kinds of notations as shown in Figure 2.6.

As we introduced before, the segmentation for each type of CAPTCHAs is a very critical operation. Since there are eight types of notations, we have to try eight times to identify which font type it belongs to if using the traditional sequential method. We propose our algorithm to identify the font type, with the higher speedup and accuracy.

After preprocessing, we cannot directly employ the histogram-based method to apply segmentation because of the characters' rotation. Besides, the neighboring characters will affect close-by characters of each others. Therefore we have to use Affine Transformation to turn the rotated

Type 1			Type 2			Type 3		
S	O	S	T	O	T	D	O	D
S	N	S	T	N	T	D	N	D
S	NN	S	T	NN	T	D	NN	D

Table 2.1: All possible cases

characters into vertical. Following is the representation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ax + by \\ dx + ey \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.2)$$

One attractive feature of this matrix representation is that we can use it to factor a complex transformation into a set of simpler transformations. Because the rotation transformation is utilized, the following rotation matrix is essential: $\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$, where θ is an angle of counterclockwise rotation around the origin.

Whenever loading an image we first perform the regular affine transformation to make it vertical. The good thing in our application is that the rotated angles of the characters in CAPTCHA examples are fixed, so our task would be much simpler. After the optimization of the overall procedure, the whole process is exhibited in Figure 2.8. We just need to check the first four characters, because the citation we deal with is the operation of the operands. We need the first operand, the operator and the second operand where the operator includes one or two characters. Then we need to identify those operators, checking the first four characters are sufficient.

Once the notation comes into our system, we check the first character. If it is an Arabic number, we know that the notation belongs to Type 3 in Table 2.1, possibly in $\langle D \ - \ - \rangle$ form. Next we go to the third character where the second operand is possibly located. If that character is a D, we are sure that $\langle D \ O \ D \rangle$ or $\langle D \ N \ D \rangle$ is the form of the notation. We can then go back to check the second character to confirm whether it is an operator represented in N or O. If the third character is not an Arabic number, indicated is that it is not the second operand, because the second and first operand should be of the same type. The option left is that the operator between the two operands is of NN form. In that case, the second operand must be located in the position of the fourth character. Since the first and the second operands must be of the same type as we mentioned before, the fourth character will be an Arabic number, then the notation is represented in $\langle D \ NN \ D \rangle$ form.

If the first character is not an Arabic number, the notation could belong to Type 1 or Type 2 listed in Table 2.1 in <S - S> or <T - T> form. Next we go to the second character to verify whether it is a symbolic operator or not. If it is, the notation is then represented in <S O S> or <T O T> form. If the second character is not symbolic, then it can be a Chinese character, representing the operator by itself. The other case is that the operator is represented by double Chinese characters. Let's move to the third character to check. For the former case, the third character is where the second operand stands, and the second operand should have the same style as the first one. So we get <S N S> or <T N T> as listed in Table 2.1. For the latter case, operator being double Chinese characters case, NN will extend to the third character position. The simplest way is to directly check the fourth character, to see whether it is equal to one particular ideograph or not.

We take the third row in Figure 2.6 as an example, the meaning of which is $9 - 5 = ?$. The first character is a digit such as D, then we move to check the third one. The third character is a digit like the first operand. Then as the last examination, the second character is to be verified to confirm the operator type. It turns out that the operator is a symbolic operator, so we conclude that the notation type is <D O D>.

We look at another example next, from the 6th row of Figure 2.6. We check the first character and find that it is a digit. We move to the third one and find that it is not a digit but a Chinese character. Quickly we understand that the operator is a double-character operator and the notation is in <D NN D> form. Actually we don't have to examine the fourth character because it must be the same type of operand as the first one.

In Figure 2.6, we check a character by its width instead of its content, which is much faster and more reliable. We found that the width of both the simplified Chinese characters and the traditional Chinese characters are 20. Besides, the width of a symbolic operator is 9 and that of a digit is 11. We set the threshold of the width as 14.

Figure 2.9 shows the basic flow chart of segmentation which is a basic sequential test procedure. Figure 2.9 illustrates the traditional serialized method. If we use the serialized segmentation shown

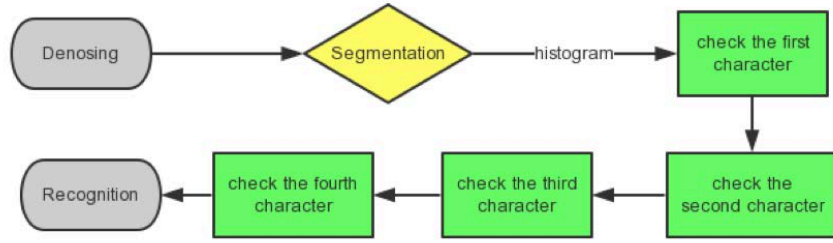


Figure 2.8: Traditional flow chart of segmentation

in Figure 2.9, the total time to identify which CAPTCHA type regarding the input image belongs to in Figure 2.6 is: $T1 = 8 * 4 * t$, where t is the time to check a single character. Regarding our algorithm, as Figure 2.7 illustrated, the whole procedure is optimized into at most three steps, thus the time to identify the correct CAPTCHA type is $T2 = 3 * t$. The speedup between ours and serialized segmentation is:

$$T1/T2 = 10.67$$

We denote the single character recognition rate as r . In the serialized segmentation, the successive recognition rate of the whole eight types $R1$ would be:

$$R1 = r^{4*8}$$

In our algorithm, the successive recognition rate $R2$ is:

$$R2 = r^{3*8}$$

Ideally, compared with the traditional algorithm, the improvement we obtained is

$$R1/R2 = r$$

One can reference to Section 2.4, Experiment, in the later part of this section to find the improvement of the single character recognition rate of our algorithm over the traditional algorithm,

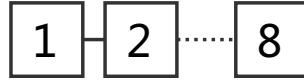


Figure 2.9: Sequentially defeating the CAPTCHA

which is roughly from 75% to 99% depending on the types of CAPTCHA.

2.3.3 Convolutional neural network construction

2.3.3.1 *The Architecture*

The CNN architecture is shown in Figure 2.10, which is adapted with one of the most classic LeNet-5 [38][39]. In Figure 2.10, the top is our image input layer and the image size is 32×32 . The net contains six layers: the first four layers are convolutional layers and the remaining two are fully connected layers. The general strategy of CNN is to extract simple features at a higher resolution by the first convolutional layer, and then convert the features in higher resolution into more complex features at a coarser resolution [40]. To obtain the coarser resolution, the most frequently used method is to sub-sample a layer by a factor of two, which can be utilized for the size of convolutional kernel as shown in the second layer in Figure 2.10. Regarding the kernel's width, the criteria is to have enough overlap to keep useful information but not too much redundant computation. Thus we choose five by five as the kernel size because three by three is too small with only one unit overlapping, and seven by seven is too big with 5 units overlapping. After the first convolutional layer extracting features, we found that if the kernel size is 5. The overall performance is better than the case with a kernel size less than 5, which means that increasing the kernel size cannot improve the performance significantly. Then we subsample those features to decrease the data size, prevent over-fitting as well as preserve the important information such as margins. Similarly we repeat the first and second layer to construct the third and fourth layer, so

that we can carry enough information to the classification layer. Finally we add a trainable classifier to the extractor which is of two fully connected layers. The output of the last fully connected layers is fed to a 82-way (10 for digits 0-9, 26 for lower case letters a-z and 26 for capital letters A-Z, 10 for simplified Chinese digits and 10 for traditional Chinese digits) softmax which produces a distribution over 82 class labels.

2.3.3.2 *Pooling layer*

Pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map. Traditionally, the neighborhoods summarized by adjacent pooling units do not overlap [41]. A pooling layer is to construct a grid of pooling units with pixels s spaced apart, each summarizing a neighborhood of size $z \times z$ centered at the location of the pooling unit. Regarding the pooling size, typical values are 2×2 or no max-pooling. Very large input images may warrant 4×4 pooling in the lower-layers. However, such a large 4×4 pooling layer will reduce the dimension of the signal by a factor of 16, and may cause throwing away too much information [37][9].

2.3.3.3 *Hyper-parameters*

CNNs' parameters are especially tricky to train, as it contains more hyper-parameters than a standard MLP (Multilayer perceptron). While the general rules of thumb for learning rates and regularization constants still apply, the following should be kept in mind when optimizing CNNs [41]. Since the feature map size decreases as the depth of layers increasing, layers near the input layer will tend to have fewer filters, while layers higher up can have much more filters. In fact, to equalize computations at each layer, the product of the number of features and the number of pixel positions is typically picked to be nearly constant across layers. To preserve the information about the input, it is required to keep the total number of activations (number of feature maps times number of pixel positions) to be non-decreasing from one layer to the next. The number of feature maps directly controls the capacity, while the features are dependant on the number of available examples and the complexity of the task [42][9].

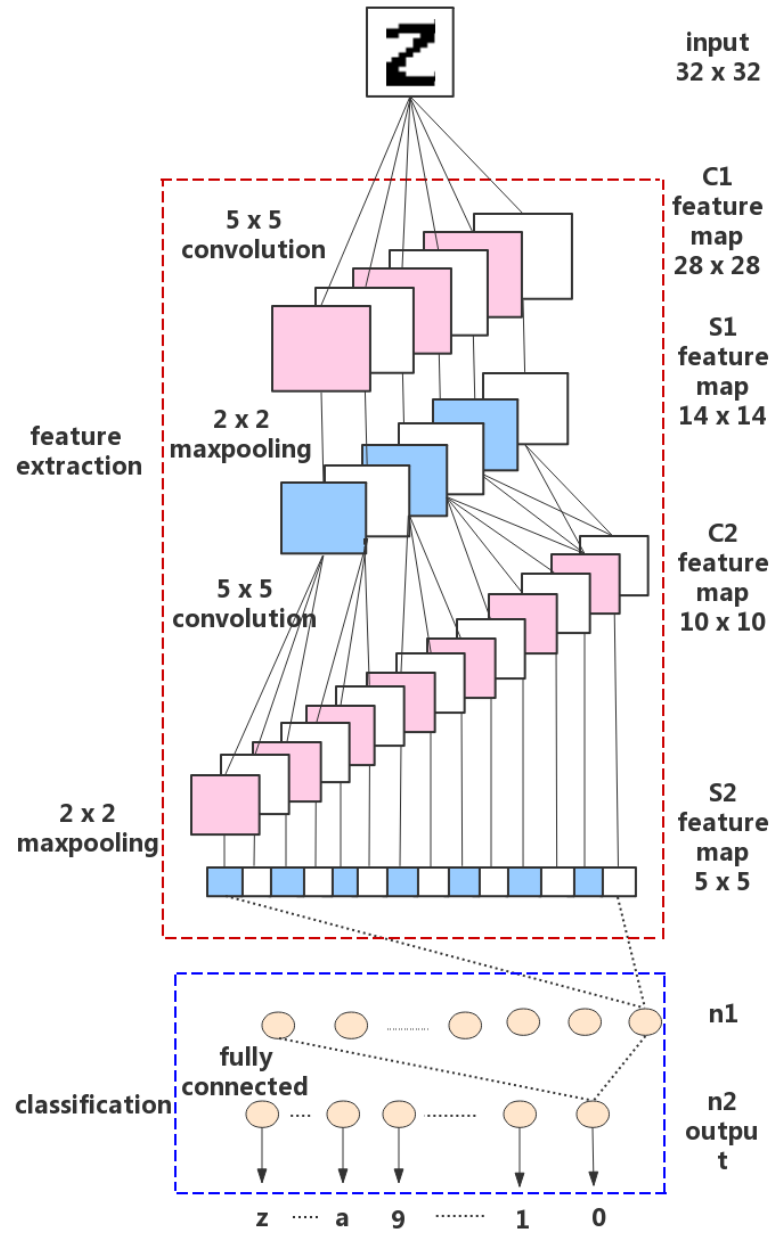


Figure 2.10: Neural network structure

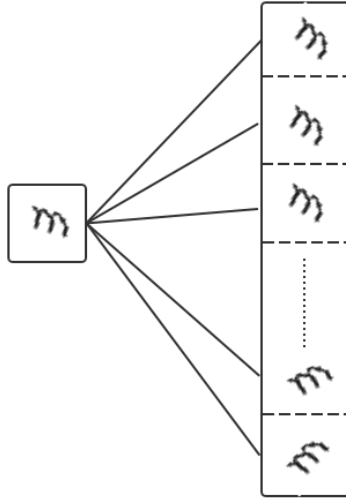


Figure 2.11: Rotate the image to diversify training data

2.3.3.4 Datasets

Our datasets are mentioned in Figure 2.2, which come from three major sources to provide CAPTCHA examples. The challenge is the insufficient size of datasets. However, with the increase of training data size, the performance of accuracy will correspondingly improve. We propose a method to gain data with larger size as shown in Figure 2.11. Through careful observation, we found that the majority of rotation angles in examples vary from -50° to $+50^\circ$. However, our manually training samples are insufficient, which means that training samples cannot cover the comprehensively rotated CAPTCHA examples. Thus we rotate our data every 5° to get the complete samples.

2.4 Summary

More than 7,000 testing samples were selected in our experiment work. The whole process of defeating the CAPTCHA is illustrated in Figure 2.12 including denoising, segmentation and recognition. We present the result of accuracy rate by our methods in Figure 2.13. Overall, CNN is much more robust than OCR/TM. However, the way to obtain the training data is really painful

because we have to manually decompose the examples into single characters and label them. Besides, we rotate the training data from -50° to 50° in every 5° for enlarging the data size ten times more than before. Rotating the training sample also brings more comprehensively circumstances for potential testing data, which proved to be more useful. This is because our raw data size is limited and cannot cover the whole rotated cases. However, we can fulfill the rotation completeness by our method.

Shown in Figure 2.13 are the results for comparison. It can be seen that the recognition rate for the particularly rotated CAPTCHAs such as the 9th and 10th row are much lower than the others. The best rotated recognition rate is 71.6% in the sixth row in Figure 2.13, and the lowest accuracy rate of rotated fonts is as low as 33%. With the increase of rotated angle, the accuracy rate is correspondingly decreasing. Therefore, the rotation hinders the accuracy of segmentation and recognition. However, CNN can always perform robustly than TM/OCR especially in rotation types. The average performance in CNN is approximately 10% better than in TM/OCR.

In fact, in some simple cases like first-row in Figure 2.13, both TM/OCR and CNN perform similarly regarding the accuracy rate. With the rotated angles increasing, more advantages appear on CNN than on TM/OCR. Figure 2.14 shows the accuracy rate much clearer.

Our CAPTCHA examples are always constructed by four characters. If and only if all of them have been correctly recognized, we consider it as a successful defeat as we introduced before. More precisely, Figure 2.15 shows the single character recognition accuracy rate. As we can see, the lowest accuracy rate is 75.96% in TM/OCR and 85.72% in CNN. The highest accuracy rate in Figure 2.15 is 98.98% in TM/OCR and 99.65% in CNN, which indicates that our method has achieved the state-of-the-art performance.

Original CAPTCHA	Denoising	Segmentation	Output
			<code>['outcome=', 1]</code>
			<code>['outcome=', 5]</code>
			<code>['outcome=', 2]</code>
			<code>['outcome=', 24]</code>
			<code>['outcome=', 36]</code>
			<code>['outcome=', 9]</code>
			<code>['outcome=', 4]</code>
			<code>['outcome=', 'qE9tAR']</code>
			<code>['outcome=', '4tzfrm']</code>
			<code>['outcome=', '4820']</code>
			<code>['outcome=', '3774']</code>
			<code>['outcome = ', '0']</code>
			<code>['outcome = ', 72]</code>
			<code>['outcome = ', -4]</code>
			<code>['outcome = ', 11]</code>

Figure 2.12: The full process of defeating the CAPTCHA




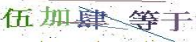







The type of CAPTCHA	Accuracy rate	
	TM/OCR	CNN
	288/300=96.0%	296/300=98.6%
	264/300=88.0%	289/300=96.3%
	244/300=81.3%	273/300=91.2%
	241/300=80.3%	260/300=86.7%
	235/300=78.3%	255/300=85%
	215/300=71.6%	263/300=87.6%
	166/300=55.3%	192/300=64%
	162/300=54.0%	195/300=65%
	100/300=33.3%	162/300=54%
	105/300=35.0%	166/300=55.3%
	192/300=64%	200/300=66.6%

Figure 2.13: The accuracy rate for each CAPTCHA

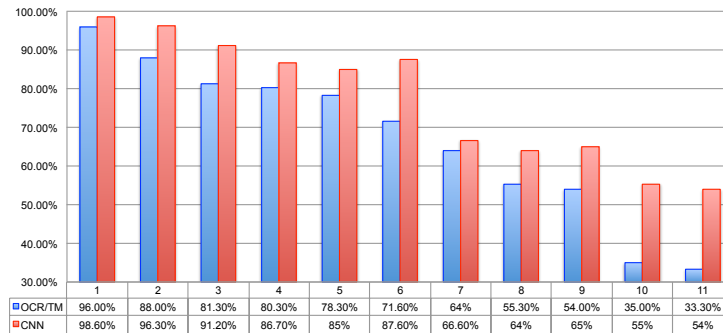


Figure 2.14: Overall CAPTCHA accuracy rate

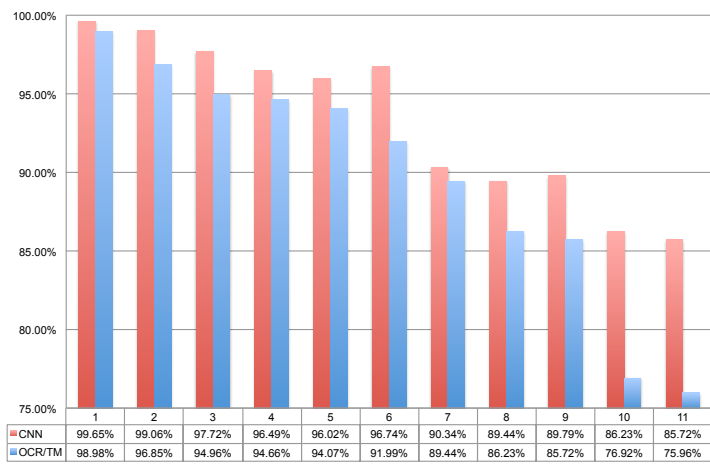


Figure 2.15: Accuracy rate of individual character

3. COMPARISONS AND SELECTIONS OF FEATURES AND CLASSIFIERS FOR SHORT TEXT CLASSIFICATION ¹

3.1 Task motivation

After obtaining the data as we introduced in Section 2, in this section, we will illustrate the task we proposed. Currently, there isn't a good quality task about Chinese short-term text classification unlike English text classification task such as AG_NEWS and IMDB. Therefore, we propose a Chinese short-term text classification task and we provide a baseline performance for evaluation.

Short text classification is unlike traditional long text documents, due to its own characteristics in terms of shortness and conciseness. The objective of this section is to compare existing non-AI methods for short-text classification. Each short-term text is constructed by less than 20 words, and we manually labeled it into two class of tags, there are eight types of labels in the big class and fifty-nine types of labels in the small class. The total number of short text files is about 400,000, and the number of labels' short-term text is equal. In other words, we can feed about 50,000 short text files to train each label in the big class, and around 6,700 files to train each label in the small class. There exist some challenges specific to short text classification. Shortness is a synonym for simplicity, which can confuse when we try to classify 59 labels with files of no more than 20 words. Besides, we have more than 400,000 short text files, which amount to approximately 5,349,348 words and would certainly create a lot of sparsities in the vectorization process. Sparsity would in turn lead to several problems such as data redundancy, sparsity matrix, etc. We will introduce them in the next few sections.

The rest of this section is organized as follows: Section 3.3.2 presents a high-level description of the approach in terms of data processing and classifier training. Section 3.3.3 introduces the system implementing our task and how we handle the challenges. Section 3.3.4 discusses the experiment results, and the section concludes with Section 3.3.5.

¹Reprinted with permission from "Comparisons and selections of features and classifiers for short text classification" by Ye Wang, Zhi Zhou, Shan Jin, Debin Liu and Mi Lu, 2017. *Proceedings of the 2017 International Conference on Artificial Intelligence Applications and Technologies (AIAAT 2017)*, Page 012-018, ©2017 IOP PUBLISHING.

Company Announcement	Big Class	Small Class
昆药集团:关于 2016 年度股权激励计划所涉限制性股票授予的进展公告 KPC Pharmaceuticals, Inc.: Announcement on the Progress of the Restricted Stock Grant in the 2016 Equity Incentive Plan	重大事项 Major Events	股权激励 Equity Incentive Compensation
新奥股份:2016 年半年度报告 ENN Ecological Holdings Co. Ltd.: 2016 Semi annual report	财务报告 Financial Report	半年报告 Semi Annual Report
京东方 A:关于回购公司部分股份并注销债权人通知的公告 BOETECHNOLOGYGROUPCO.,LTD: Announcement on repurchasing part of company shares and canceling the creditor	股权股本 Equity and Capital Stock	回购股权 Repurchasing Equity
Total: 409,871 short text announcement	8 labels	59 labels

Figure 3.1: Announcement examples in dataset

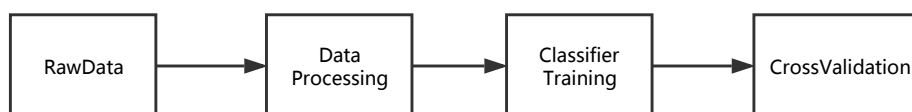


Figure 3.2: Basic flow chart

3.2 General approach description

As can be seen, Figure 3.1 lists some examples of our data samples which will be used later. Fig.2 shows the basic flow chart which can be regarded as top level description of our approach. In this section, we will discuss in details about the entire process.

3.2.1 Data processing

There are many different data processing techniques. We should be aware of their characteristics and choose the appropriate method.

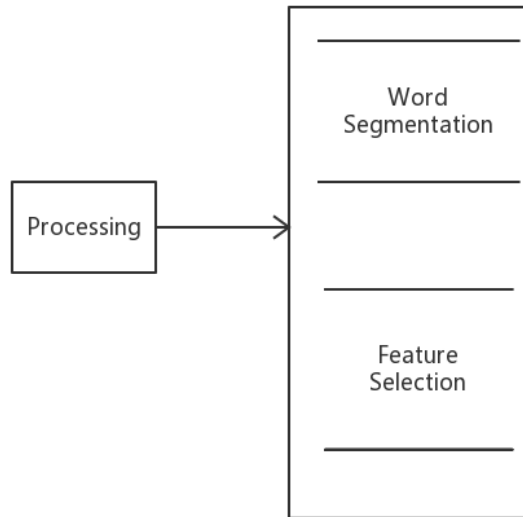


Figure 3.3: Processing the data

3.2.1.1 Segmentation

According to [43], segmentation is becoming increasingly more important in Chinese, Japanese and many other Asian language processing tasks. Unlike English, Chinese words are not delimited by whitespace characters, so word segmentation is a fundamental first step in processing these languages. Several algorithms have been proposed for Chinese word segmentation [44], and the study in automatic Chinese word segmentation has made significant progresses in recent years. For the purpose of our study, we just chose the currently most popular segmentation method which is based on prefix-trie and the Viterbi algorithm.

3.2.1.2 Feature selection

Figure 3.2 shows the data processing flow chart. Once we get the segmented words, we can convert them into a vector matrix for later training. This process is called word embedding, or distributional models. The reason for constructing such a vector matrix is that we can utilize the term-context matrix to represent the short text, which is much simpler for training purposes. There are many ways to construct the vectors, such as sparse vectors and dense vectors. Sparse vectors

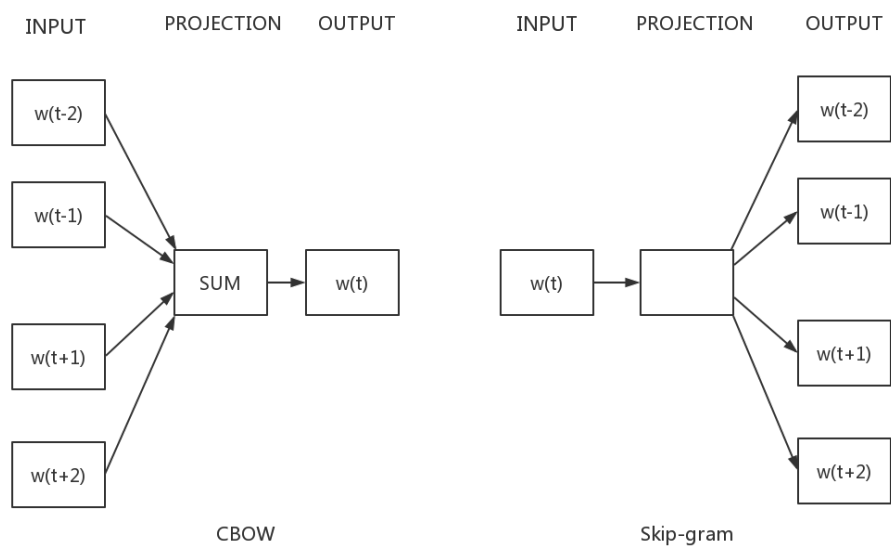


Figure 3.4: Two models of word2vec

have most elements equal to zero and lengths of about 20,000 to 50,000, which will be very time-consuming computationally, while dense vectors are constructed in 100-500 dimensions, so are much faster than sparse vectors when used in training and classifications. Dense vectors may also better capture synonymies than sparse vectors [45]. Moreover, we employed two methods for the sparse vector construction, i.e. counter vectorizer and term frequency-inverse document frequency (tf-idf). Counter vectorizer is also called one-hot coding, which is applied to categorical features.

Categorical features are "attribute-value" pairs where the value is restricted to a list of discrete possibilities without ordering. Counter vectorizer is like a raw vectorizer, while tf-idf is more refined, since it is a numerical statistic that is intended to reflect how important a word is to the short text in our collections. It is often used as a weighted factor in applications. The key characteristic of tf-idf is that it increases proportionally with the frequency of a word appearing in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general [46].

Compared with sparse vectors, dense vectors are more popular because they are shorter and

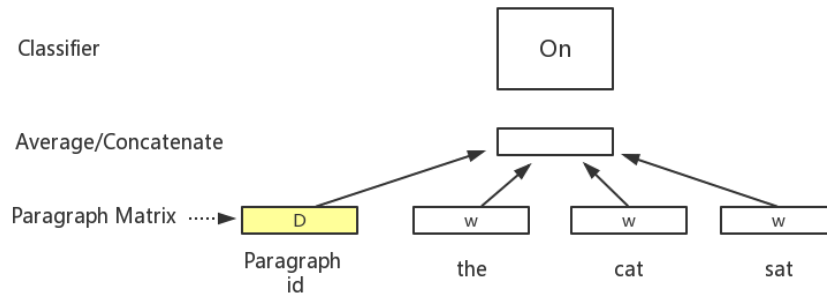


Figure 3.5: Paragraph2vec model

yet more meaningful. There are three popular methods for constructing dense vectors, i.e. singular value decomposition, neural language models and Brown clustering. Here we would like to focus on the neural language model, which is the state-of-the-art method for dense vector construction [45][47]. There are two type of neural language models, Skip-gram and continuous bag of words (CBOW), which are also collectively called word2vec models, as shown in Figure 3.4. In short, the CBOW architecture predicts the current word based on the context while skip-gram predicts surrounding words given the current word. One advantage of dense vectors is that we can get a short and yet meaningful vector to represent each word. Also, it has the superior characteristic that the matrix of similar words also has a closer distance, which is helpful to constructing the thesaurus. Moreover, [48] enhanced his work of word embedding and proposed a novel model called paragraph2vec (or doc2vec) as shown in Figure 3.5. This method trains the entire document as a vector matrix, while for word2vec, the basic idea is to predict the word. Similar to word2vec, in the training of the document vector we need to go through the whole text. We have also implemented doc2vec in this section and will compare it with other feature selection methods in the results and analyses part.

3.2.2 Different types of classifiers

Once the feature is selected, the next step is to train the classifier. Classification is one of the most important steps in all machine learning tasks. Classification is the problem of identifying to

which set or category a new observation belongs, on the basis of a training set of data containing observations whose class is known. Since we already have labeled all the instances, we only need to choose supervised learning classifiers. Among the various learning algorithms, we cannot simply decide the best one before experimenting and comparing some of them. Hence we selected several popular classifiers, including naive Bayes (NB), decision tree (DT), k-nearest neighbor (KNN), logistic regression (LR) and support vector classifier (SVC), and applied them with both the big and small classes of labels. Each classifier has its own advantages and disadvantages.

3.2.2.1 *Naive bayes*

Naive Bayes (NB) methods build upon the famous Bayes' theorem with the "naive" assumption of independence between each pair of features. The NB method contains a very low time complexity, and its assumption usually works quite well in some real-world situations such as spam filtering and document classification. As a consequence of the decoupling of the conditional probability distributions of different features, the probability distribution of each feature can be independently estimated as a one dimensional distribution, which in turn helps solve problems stemming from the curse of dimensionality. In this study we compared both Gaussian Naive Bayes (GNB) and Multinomial Naive Bayes (MNB) classifiers. When dealing with dense vectors, we treat those data as continuous data, and when dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. In a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial (p_1, \dots, p_n) where p_i is the probability that event i occurs (or k such multinomials occur in the multi-class case).

3.2.2.2 *Decision tree*

Decision tree learning is a useful machine learning method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. Actually there are basically two types of decision trees: classification and regression trees. In classification tree analysis the predicted outcome is the class to which the data belongs,

while in regression tree analysis the predicted outcome can be considered a real number. Decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Besides, it is very easy to construct an overfitting tree, thus appropriate pruning is necessary to avoid over-complexity.

3.2.2.3 *Logistic regression*

Multinomial logistic regression is known by a variety of other names, including polytomous LR, multi-class LR, softmax regression, multinomial logit, maximum entropy (MaxEnt) classifier, and conditional maximum entropy model. In fact, multinomial logistic regression is a classification method that generalizes logistic regression to multiple-class problems. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

3.2.2.4 *Support vector machine*

Support Vector Machine (SVM) is widely used among classification, regression and even outlier detection. The advantage of SVM is obvious: First, it is very effective not only in high dimensional spaces, but also in cases if the number of dimensions is greater than the number of samples. Second, it is considerably memory efficient due to its own advantage of kernel mapping to high-dimensional feature spaces. Since SVM is a double-edged sword, the disadvantage of SVM is that if the number of feature is much greater than the number of samples, the method is likely to give poor performance. A linear support vector classifier (SVC) is used in this section.

3.3 **Specific system implementations**

The whole structure of our system is divided into four parts as we illustrated before: Getting raw data, Processing data, training classifier and cross validation. We implemented it by Python2.7 with some open source APIs like Scikit-learn [49] and Gensim [46].

3.4 **Summary**

Several classifiers have been trained to classify Chinese short text files, including GNB, MNB, SVC, LR, KNN and DT. However, we will only present the experiment results of MNB, SVC and

big				small			
Data = 1/2				Data = 1/2			
Classifier	Accuracy, %			Classifier	Accuracy, %		
	word2vec	doc2vec	tf-idf/counter		word2vec	doc2vec	tf-idf/counter
MNB	46.5454798	30.979916	67.9974846	MNB	50.5129724	3.9339202	71.8358194
SVC	56.523723	20.7917638	70.6630796	SVC	77.1284876	2.055327	83.5525026
LR	64.3266498	30.979916	70.7194048	LR	81.4567048	2.4392982	84.218733
Data = 1/5				Data = 1/5			
Classifier	Accuracy, %			Classifier	Accuracy, %		
	word2vec	doc2vec	tf-idf/counter		word2vec	doc2vec	tf-idf/counter
MNB	46.020836	30.9789578	66.8190094	MNB	47.8368036	3.6108782	71.3708568
SVC	58.7445636	22.0106284	69.811158	SVC	70.3710892	2.7923356	83.9617922
LR	64.9365838	30.9789578	70.0025738	LR	81.5665502	2.438548	84.1286016
Data = 1/10				Data = 1/10			
Classifier	Accuracy, %			Classifier	Accuracy, %		
	word2vec	doc2vec	tf-idf/counter		word2vec	doc2vec	tf-idf/counter
MNB	45.997871	30.979801	64.6257898	MNB	47.0733842	3.5133214	71.528666
SVC	56.9952578	25.7708636	69.0387578	SVC	60.1547316	2.1768262	83.9285676
LR	64.455278	30.979801	68.6656358	LR	81.3346074	2.4397412	83.5524258

Figure 3.6: Result

LR in this section, since they are much better than those of the other classifiers.

Figure 3.6 to Figure 3.10 show the 5-fold cross-validation results for each of the three classifiers. In the cross-validation tests We have used four features, i.e. *word2vec*, *doc2vec*, *tf-idf* and *counter vectorizers*. The performances of the last two are similar so we treat them indifferently as a single *tf-idf/counter* feature. In addition, we have filtered stop words in each experiment, which also slightly improves the accuracy.

From the tables we can see that in all cases, the *tf-idf/counter* feature has the highest accuracy, while *word2vec* next, and *doc2vec* the lowest. The feature *doc2vec* produces the worst result in any circumstance, even much worse than plain guessing, which is different from the long text classification results [48]. We also get different results for the big and small classes of labels. The small class generally results in higher accuracy than the big class, which is counterintuitive and needs further investigation.

A comparison of different classifiers would show that with the *tf-idf/counters* feature, LR and SVC are much better than MNB, and the results of the two are comparable with each other. This may not be the case when other features are used. While a high accuracy is expected for the

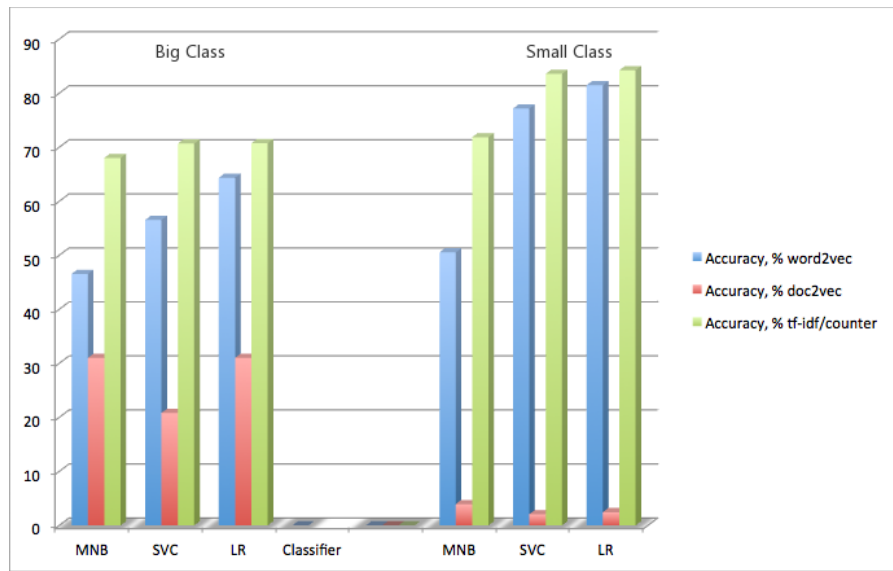


Figure 3.7: 1/2 datasets of overall

SVC because of the use of kernel function, it is a little surprising that the overall highest accuracy 84.22% is associated with LR. Additionally we have tried to change the size of the dataset, and it seems that increasing the size of the dataset can raise the accuracy, but this impact is not significant.



Figure 3.8: 1/2 datasets of word2vec

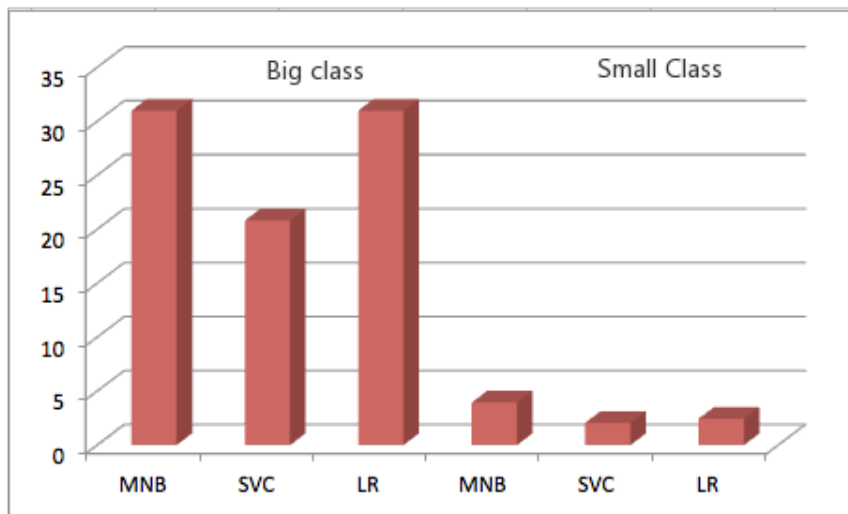


Figure 3.9: 1/2 datasets of doc2vec

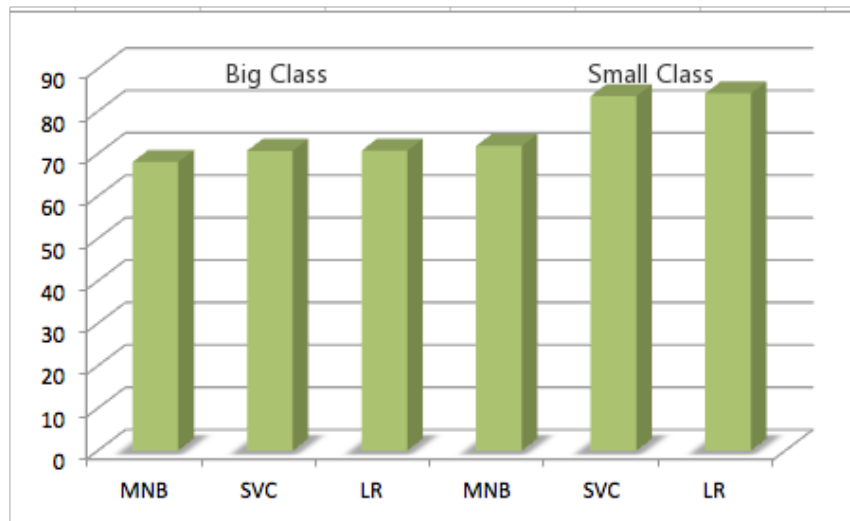


Figure 3.10: 1/2 datasets of tf-idf/counter

4. ATTENTION-AUGMENTATION WITH MULTI-RESIDUAL IN BIDIRECTIONAL LSTM¹

4.1 Background

Unlike image data, all the sequential data contains a time series characteristic. Recurrent neural networks (RNNs) have been proven to be efficient in processing sequential data. However, the traditional RNNs have suffered from the gradient diminishing problem until the advent of Long Short-Term Memory (LSTM). However, LSTM is weak in capturing long-time dependency in sequential data due to the inadequacy of memory capacity in LSTM cells. To address this challenge, we propose an Attention-augmentation Bidirectional Multi-residual Recurrent Neural Network (ABMRNN) to overcome the deficiency. We propose an algorithm which integrates both past and future information at every time step with omniscient attention model. The multi-residual mechanism has also been leveraged in the proposed model targeting the pattern of the relationship between current time step and further distant time steps instead of only one previous time step. The results of experiments show that our model outperforms the traditional statistical classifiers and other existing RNN architectures.

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are the two primary architectures in neural networks. RNNs are often applied to sequential data such as natural language processing and speech processing [11, 12], while CNNs are more employed in image processing areas [13, 7, 14]. Among the existing RNN models, LSTM [15] is one of the most popular approaches since it initially solves gradient vanishing and exploding problems during RNN training by introducing forget gates and memory cells. After the literature review, we found that numerous RNN variations have been proposed to achieve the state-of-the-art performance in different tasks, where LSTM is the cornerstone of those structures. However, due to the limited memory

¹Reprinted with permission from “An Attention-aware Bidirectional Multi-residual Recurrent Neural Network (Abmrnn): A Study about Better Short-term Text Classification” by Ye Wang, Han Wang, Xinxiang Zhang, Theodora Chaspari, Yoonsuck Choe, Mi Lu, 2019. *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Page 3582-3586, ©2019 IEEE.

cell in LSTM, when a time sequence is long, the LSTM performance is heavily influenced.

With the increase of the depth of layers and the length of the sequences, residual networks have proved their advantages in both CNNs [16, 17] and RNNs [18]. Residual networks connect current and distant previous time steps for optimizing the layer information. [16] and [19] propose similar residual ideas to randomly connect one previous distant time step to current time step, where the problem of long time dependency is solved partially. Therefore, the residual networks motivate us to combine the residual network with LSTM, where the information in current time step has been updated dynamically based on the attained correlation between previous time steps and current time step.

To better attain the correlation between current time step and previous time steps, the attention model is widely applied in image processing, speech processing and natural language processing. The objective of the attention model is to ultimately optimize the training procedure when the amount of attention is limited. [20] initially leverages the attention model from image processing to natural language processing. [21] proposes a model as a decoder network between previous states and current state. [22] simplifies the model as an attention-based weighted pooling RNN to acquire utterance representation in speech processing. Since the attention is limited, the way to effectively distribute those attention becomes considerably important. Inspired by the attention-based approaches, we leverage the attention model [23] to strengthen the correlation between the current state and both previous and future distant states. As for [23], they focus on the relationship between current time step with previous information. Since the objective is to allocate the attention properly, we regard the past and the future time steps as the same important time steps, which means we integrate both the previous and the future time steps to refine the information of current time step instead of only relying on previous time steps.

Therefore, to address the aforementioned challenges in long time dependency and to optimize the text correlations, this section develops an Attention-augmentation Bidirectional Multi-residual Recurrent Neural Network (ABMRNN). The proposed ABMRNN achieves the state-of-the-art performance among several existing sequential classification tasks. The main contributions of this

work are summarized as following:

- Our algorithm overcomes the deficiency of LSTM in weak modeling the long-time dependency, so we can handle much longer sequential data and obtain higher accuracy rate in longer sequential tasks. In this algorithm, we design a novel bi-directional layer to dynamically acquire and allocate attention from both previous and future time steps. Bi-directional layers help us focus not only on the past but also on the future, so that we can attain the better correlation between current steps and distant time steps (In STCT, after incorporating the bi-direction layer, the accuracy has improved from 93.01% to 94.10%).
- To better supplement the acquired attention from bi-directional layer, we also leverage the multi-residual mechanism to the recurrent networks. Compared with traditional residual networks, the advantages of ours are more obvious because the proposed multi-residual mechanism is more rational than previous residual networks in sequence learning as the traditional residual networks connect current time step with only one randomly previous time step.
- The proposed model contains fewer parameters than current popular models such as [16] and [24], which indicates that the architecture of the proposed model is less complicated than those popular models. The proposed model also achieves the state-of-the-art performance in sequence learning of STCT from 93.01% to 96.50%.

Three major directions have been studied in recent years towards the structure exploration for sequence learning, including capturing better feature representation, optimizing cell memory usage and improving the capacity of long time dependency. First, an increasing number of layers and numerous embedding techniques are employed for feature extraction [50][47][51]. However, with the increase of layers, the computational complexity becomes critical and unaffordable in current neural networks. Second, a wide range of variations towards the RNN interior cell structure units such as LSTM and GRU [52] are proposed. Nonetheless, those RNN variations are confronted with the same imperfection due to the limited cell memory. Third, the attention-based approaches

[20][53] are proposed to improve the capacity of long time dependency in RNN variations. Nevertheless, partial information has been obtained because the proposed attention models only focus on previous states.

Therefore, the contribution of our work integrates the advantages of residual networks and attention-augmentation mechanism for the tasks of interest. Unlike the popular trend of combining deeper and wider neural networks [54], we propose a novel RNN variation with forward and backward layers. The proposed model attains the text information from both past and future time steps based on the limited memory cell of LSTM and improves the capacity of long time dependency.

4.2 Recurrent neural networks preliminaries

The basic LSTM architecture of solving the problem of gradient diminishing in traditional RNNs is described and the illustrated equations are given in Section 4.3.1. Besides, the fundamental residual mechanism employed in recurrent neural networks is presented in Section 4.3.2.

4.2.1 Long short-term memory (LSTM)

The traditional RNNs have suffered from gradient diminishing problem until the advent of LSTM. The appearance of LSTM is meaningful because the authors introduced the gates' mechanism by adding nonlinear activation functions. Activation functions squash the values of these vectors between 0 and 1. Figure 4.1 shows the structure of LSTM cell, where Equations (4.1-4.6) follows the data flow: The input gate activation vector i_t (forget gate activation vector f_t , output gate activation vector o_t respectively) is obtained by the sigmoid function of the updated current input vector x_t and updated hidden state vector h_{t-1} . The update of x_t is by a weight matrix U converting the input to the current hidden layer in the input gate (forget gate, output gate respectively). The update of h_{t-1} is through matrix W^i , representing a recurrent connection between the previous hidden layer and the current layer in the input gate (forget gate, output gate respectively).

$$i_t = \sigma(x_t U^i + h_{t-1} W^i) \quad (4.1)$$

$$f_t = \sigma(x_t U^f + h_{t-1} W^f) \quad (4.2)$$

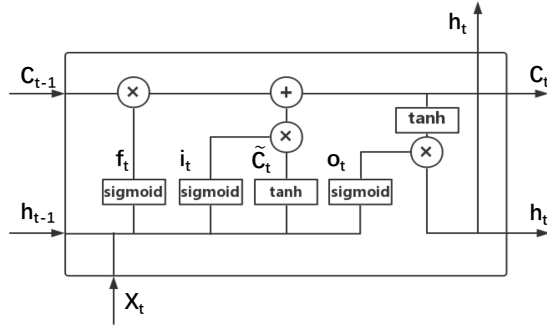


Figure 4.1: LSTM cell

$$o_t = \sigma(x_t U^o + h_{t-1} W^o) \quad (4.3)$$

In the forget gate, by the element-wise multiplication activation function with input and hidden state vectors, we can control the amount of information from the previous state. Similarly, in the output gate, we can control the amount of information between internal state and external network.

$$\tilde{C}_t = \tanh(x_t U^c + h_{t-1} W^c) \quad (4.4)$$

$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \quad (4.5)$$

$$h_t = \tanh(C_t) * o_t \quad (4.6)$$

For cell state vector C_t , \tilde{C}_t is the candidate value to update it. \tilde{C}_t can be obtained by the tanh function of updated x_t and update h_t , the output vector of the LSTM unit in Equation 4.6. C_t is obtained by the sigmoid function of two Hadamard products (element-wise products), the Hadamard product of f_t and C_{t-1} , and that of i_t and \tilde{C}_t .

4.2.2 Recurrent residual network

LSTM solves gradient vanishing and exploding problems. However, the dependency between the past and the current information is neglected because current time step only depends on previous time step if the time sequence is too long. To enhance such a distant relationship, residual

recurrent neural networks have been proposed [18][19].

Figure 4.2 shows the general structure of a residual recurrent network. Residual recurrent network introduces a direct shortcut between different time steps to strengthen the connection.

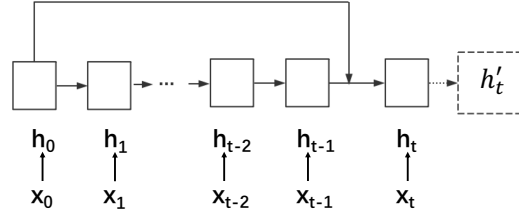


Figure 4.2: Residual recurrent network

Regarding the implementation of the basic residual network, for each current time step t , we consider both the previous time step $t - 1$ and the additional specific previous time step (e.g. we assume the time step to be t_0 in Figure 4.2).

$$C_{t-1}^{new} = C_{t-1}^{old} + \alpha * C_0 \quad (4.7)$$

$$h_{t-1}^{new} = h_{t-1}^{old} + \alpha * h_0 \quad (4.8)$$

where α represents the specific scalar weight of how much information is imported from previous time step to current time step. Recurrent residual networks leverage the convolutional residual network [16] and improve the performance in particular sequential tasks.

4.3 Proposed scheme

In this section, we propose an Attention-augmentation Bidirectional Multi-residual Recurrent Neural Network (ABMRNN). The algorithm will be illustrated and scheme described in details. First, a modified attention model is proposed which can resolve the memory cell limitation in

LSTM, by the sliding window-based implementation. Also, efforts are made to identify the specific previous time step such that linking to it is most effective. Instead of randomly connecting to a previous time step, a multi-residual mechanism is developed to enhance the correlation of current time step and distant previous time steps. Additionally, a bidirectional multi-residual mechanism is proposed that can combine both past and future information comprehensively, rather than partially capturing the previous information only. Moreover, the detailed training procedure of ABMRNN is made available, and its function further explained.

4.3.1 Attention-augmentation mechanism

The objective of the attention model is to ultimately optimize the training procedure when the amount of attention is limited. Therefore, the way to effectively distribute those attention becomes considerably crucial. We illustrate the equations as follows:

$$WS = \sum_{T=t_1}^{tn} (a_T \times h_T) \quad (4.9)$$

$$a_T = \frac{\exp(W \cdot h_T)}{\sum_{T=t_1}^{tn} \exp(W \cdot h_T)} \quad (4.10)$$

In Equation 4.9, we define a Weighted Summation (WS) at current time step T as the whole attention from previous time steps. h_T represents the value in hidden state at time step T . a_T is a scalar value representing the weight at time step T . We compute a_T through a softmax form, and W is a parameter which needs to be learned during training. $\exp(W \times h_T)$ represents the potential energy at time step T . Figure 4.3 illustrates one example of the attention model. To simplify the model, previous six steps are considered. At time step t , we compute the relationship between h_t and previous time steps. The energy height represents the corresponding weight value. Therefore, we put more attention in specific time steps of which the energy is high. In Figure 4.3, we acquire the highest attention at $t - 3$ and the lowest attention at $t - 6$ when current time step is t . For every selected time step T , theoretically, we should review all the previous states to obtain the comprehensive relationship. However, the complexity of such an attention model is too high, being $\mathcal{O}(N^2)$ if we compute all the previous states. Due to the limitation of computational power,

we select the fixed N past states to cover by a sliding window, complexity is hence reduced to $\mathcal{O}(N)$.

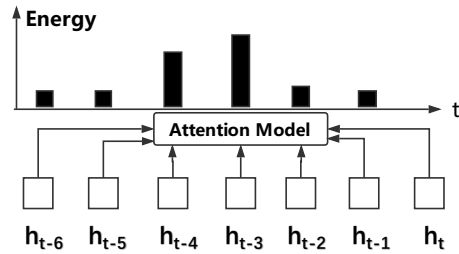


Figure 4.3: Attention model

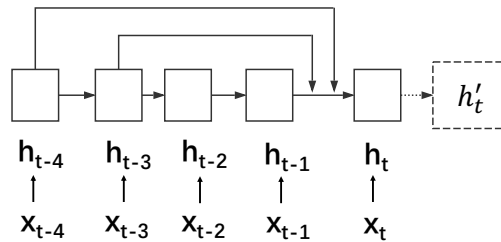


Figure 4.4: Multi-residual LSTM with attention Model, unrolling our model along the time axis. The dashed box indicates the updated state in current time step.

4.3.2 Multi-residual LSTM

Due to the limited memory cell in LSTM, when the time sequence is long, the LSTM performance is heavily impeded. The residual networks introduced in Section 5.3.2 inspire us by combing residual network and LSTM with attention model. In this way, the information in current

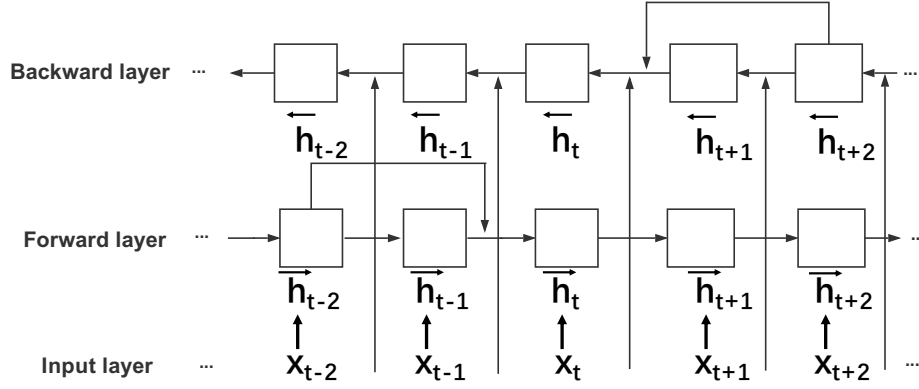


Figure 4.5: Bidirectional multi-residual LSTM with attention model

time step has been updated reasonably because we obtain the correlation between previous time steps and current time step.

[16] initially proposes a residual learning framework. They attempt to build a block as:

$$y = \mathcal{F}(x, \{W_i\}) + x \quad (4.11)$$

where x and y are input and output vectors respectively. The function $\mathcal{F}(x, \{W_i\})$ represents the residual mapping to be learned. The operation $\mathcal{F} + x$ is performed by a shortcut connection and element-wise addition.

Our idea is illustrated in Figure 4.4. With the help of attention model, we are inferred that h_{t-4} and h_{t-3} gained more attentions compared with other states when we only pick top two attentions. Therefore, we import the information from h_{t-4} and h_{t-3} to current time step h_t . The proposed equations below are introduced to explain our model:

$$\tilde{C}_t = \tanh(W_C \cdot [h'_{t-1}, x_t]) \quad (4.12)$$

$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \quad (4.13)$$

$$C' = \Sigma_T(W_{C_T} \times C_T) \quad (4.14)$$

$$W_{C_T} = \frac{\exp(W \cdot C_T)}{\Sigma_T \exp(W \cdot C_T)} \quad (4.15)$$

$$h_t = \tanh(C_t) * o_t \quad (4.16)$$

$$h' = \Sigma_T(W_{h_T} \times h_T) \quad (4.17)$$

$$W_{h_T} = \frac{\exp(W \cdot h_T)}{\Sigma_T \exp(W \cdot h_T)} \quad (4.18)$$

In Equation 4.12, we modify \tilde{C}_t based on the updated previous hidden state h'_{t-1} . In Equation 4.13, we calculate C_t by using the updated \tilde{C}_t . In our model, we update every C_t and h_t not only depending on time $t - 1$, but also involving several past states by weighted summation of multi-residual schemes in terms of memory cells and hidden states. Therefore, in Equation 4.14 and 4.17, the updated C' and h' are introduced by the weighted summation of the previous time steps. T is the candidate set of previous time steps. W_{C_T} and W_{h_T} are both scalar weighted numbers at the corresponding specific time steps. In Equation 4.15 and 4.18, W is a shared candidate vector by the model which needs to be learned from data.

4.3.3 Bidirectional multi-residual network

Compared with the previous models, We add one more layer as shown in Figure 4.5. The forward sequence and backward sequence are fed into training processing simultaneously. The advantage is to enable updating the weights combining both previous and future time steps. The equations are shown as follows:

$$h'_t = [\vec{h}_t, \overleftarrow{h}_t] \quad (4.19)$$

$$\vec{h}'_{t-1} = \vec{h}_{t-1} + \Sigma_{\vec{T}} \vec{a}_{\vec{T}} (\tanh(C_T) \otimes \sigma(x_T)) \quad (4.20)$$

$$\overleftarrow{h}'_{t-1} = \overleftarrow{h}_{t-1} + \Sigma_{\overleftarrow{T}} \overleftarrow{a}_{\overleftarrow{T}} (\tanh(C_T) \otimes \sigma(x_T)) \quad (4.21)$$

where $\vec{T}, \overleftarrow{T} \in \mathbb{N}, t - n \leq \vec{T} \leq t - 1, t + 1 \leq \overleftarrow{T} \leq t + n$ and \otimes represents tensor product.

In Equation 4.20 and Equation 4.21, \vec{h}_{t-1} and \overleftarrow{h}_{t-1} are the original forward and backward hidden states at time step $t - 1$ of bidirectional LSTM respectively. \vec{h}'_{t-1} and \overleftarrow{h}'_{t-1} are updated forward and backward hidden states at time step $t - 1$ of the proposed ABMRNN. The concept of residual is the weighted summation of the hidden states from selected time steps T based on the attention scalar a_T obtained in Equation 4.10. The updated hidden states at time step $t - 1$ are the input to compute the output at time step t .

Since the forward sequence and backward sequence are fed into training processing simultaneously, both of the past and future information have been considered together with the current time step. Besides, our model allows more time flexibilities which enable recalling past pieces of information, predicting the future time steps and evaluating the influences between each states and current state.

4.3.4 Training procedure

Algorithm 2: ABMRNN training procedure

```

 $x_{bi} \leftarrow \{\vec{x}, \overleftarrow{x}\}$  where  $\vec{x}$  is the forward input,  $t$  is the target value and  $\overleftarrow{x}$  is the backward
(reversed) input.
 $\epsilon$  : number of epochs
 $e \leftarrow 0$ 
for  $e < \epsilon$  do
  for  $x_T \in x_{bi}$  do
     $y \leftarrow \mathcal{F}(x_{bi}, \{W\})$ 
     $W_{tmp} \leftarrow \mathcal{A}(\{W\}), h^* \leftarrow \mathcal{H}(h, \{W_{tmp}\})$ 
     $y^* \leftarrow \mathcal{F}(y, h^*, \{W_{tmp}\})$ 
    error  $E \leftarrow \|t - y^*\|$ 
    update  $W \leftarrow \text{backpropagate}(W, E)$ 
  end for
   $e \leftarrow e + 1$ 
end for

```

We present the training procedure of ABMRNN as pseudo-code in Algorithm 1. The input sequence x_{bi} is composed of a forward order sequence and a backward order sequence. The ob-

jective is to minimize the loss function by updating hidden states and the corresponding attention-augmentation dynamics.

\mathcal{F} is denoted as the function of ABMRNN to obtain output states when the input is x_{bi} and matrix weights is W . \mathcal{A} represents the function of updated attention, where we can obtain the attention distribution. \mathcal{H} stands for the function of updated hidden states which means we have imported the important information from previous and future to current time step. W is the matrix weights while W_{tmp} is the temporary updated weights from attention model \mathcal{A} . h is defined as the initial hidden states while h^* is treated as the updated hidden states. y is the initial output while y^* is the updated output.

4.4 Summary

Dataset	Ave. Len	Max Len	#Classes	#Train : #Test
STCT	20	30	8	28,000 : 12,000
IMDB	300	3000	2	25,000 : 25,000
AG_NEWS	30	200	4	8,000 : 1,000
MNIST	784	784	10	60,000 : 10,000

Table 4.1: Classification datasets

In this section, we will introduce the classification tasks and illustrate our experimental results.

Table 4.1 shows the detailed statistics of each dataset.

1. Short-term text classification task (STCT) is still a challenging task. Unlike traditional long text documents, short-term texts including headings and news titles are usually concise, which somehow impact the performance. [55] introduces STCT which is constructed by average 20 Chinese words in coarse and refined categories. There are eight labels. The total number of texts is 400,000. Besides, the baseline performance has been provided by traditional statistical methods such as support vector machine, decision tree, logistic regression and so on.

2. AG_NEWS is a collection of more than 1 million news articles. All the titles have been labeled in four categories. We randomly select 8,000 samples for training and 1000 samples for testing.
3. IMDB movie review dataset is a binary classification task containing movie reviews with positive and negative labels. We select 5000 samples in total, half for training and the other half for testing. The maximum length in the review is up to 3000 and the average length is about 300.
4. Originally, MNIST is an image classification task (10 categories). However, we flat the pixels as the sequential data and feed into the training to predict the image label. Therefore, MNIST is assumed as a solid task for long time dependencies modeling (up to 784). There are 60000 training samples and 10000 testing samples.

Model	IMDB	AG_NEWS	MNIST	STCT
Plain LSTM	88.77%	82.33%	97.01%	93.01%
Bi-LSTM	89.91%	83.13%	98.31%	94.10%
2-layer LSTM	88.42%	82.27%	98.03%	93.16%
1-layer IndRNN	80.60%	84.98%	97.58%	93.02%
5-layer IndRNN	76.39%	84.74%	97.71%	88.89%
Plain RNN	77.12%	80.33%	97.66%	78.89%
5-layer RNN	50.00%	77.76%	97.45%	87.23%
1-D CNN	88.70%	84.61%	98.01%	94.50%
Attention-LSTM	89.50%	82.17%	98.31%	95.88%
Residual-LSTM	90.80%	84.71%	98.03%	93.55%
Our-model	90.91%	86.31%	98.53%	96.50%

Table 4.2: Accuracy in classification results

Regarding the preprocessing, we leverage the mechanism [44] for word segmentation. Since English is delimited by whitespace, nevertheless, Chinese, Korean and any other Asian language are not. Therefore, segmentation is a fundamental step in the first step. For our study, we adopt the Viterbi algorithm [56] in segmentation.

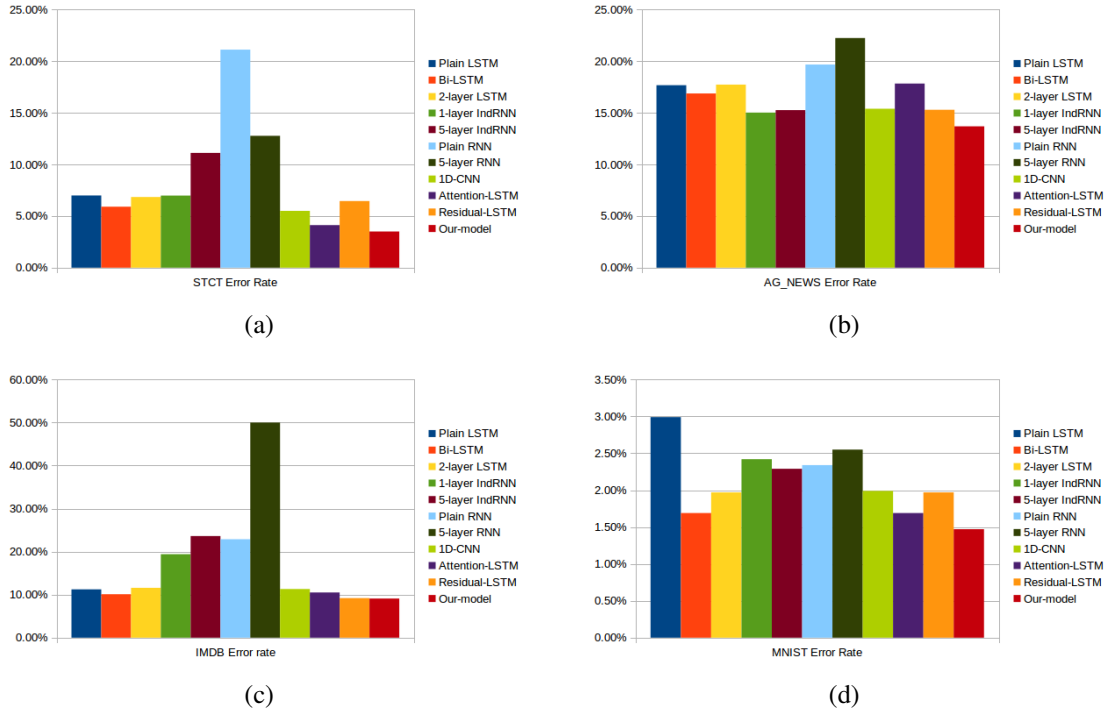


Figure 4.6: Error rate

The next step is word embedding. There are numerous methods for word embedding such as dense vector and sparse vector. We apply word2vec [47] for word embedding. The dense vector is more superior than sparse vector in terms of training speed and capacity of synonym capturing. Dense vectors are much shorter than sparse vectors, which means the computation power will be low. Besides, dense vectors obtained by word2vec reveal an attractive property that similar words in the vector matrix have a closer distance than others.

After selecting the features, the next step is to build the neural network. For better illustrating the improvement, various models are evaluated and compared such as plain RNNs, LSTMs, Bidirectional LSTM, 1-D CNNs, single residual and multi-residual networks. Two layers with 128 forward and 128 backward LSTM units are employed in our model. We also utilized gradient clipping [57]. All the weights are randomly initialized by the isotropic Gaussian distribution of variance 0.1. The dropout rate is 0.2 for each layer [58] and the batch size is 64. Regarding the other models, we keep the consistent setting with 128 hidden units in hidden layers.

Overall results of the experiment are shown in Table 4.2. Our model achieves the state-of-the-art performance in STCT. The highest accuracy rate in STCT is 96.5%, while the baseline performance provided is 69.03% [55]. We improve the ground truth about 39.7%, even the plain RNN model outperforms the statistical classification models (SVM 69.03% vs. Plain RNN 78.89%). Besides, with the model becoming more advanced, the corresponding accuracy rate is increasingly improved (Plain RNN 78.89% - LSTM 93.01% - Bi-LSTM 94.10%). We leverage the attention mechanism for optimizing the relationship between distant time steps, which improves the performance as well (plain LSTM 93.01% vs. attention LSTM 95.88%). We also attempt other architectures in STCT such as IndRNN [54], multi-layer RNNs. Our model outperforms all the existing methods.

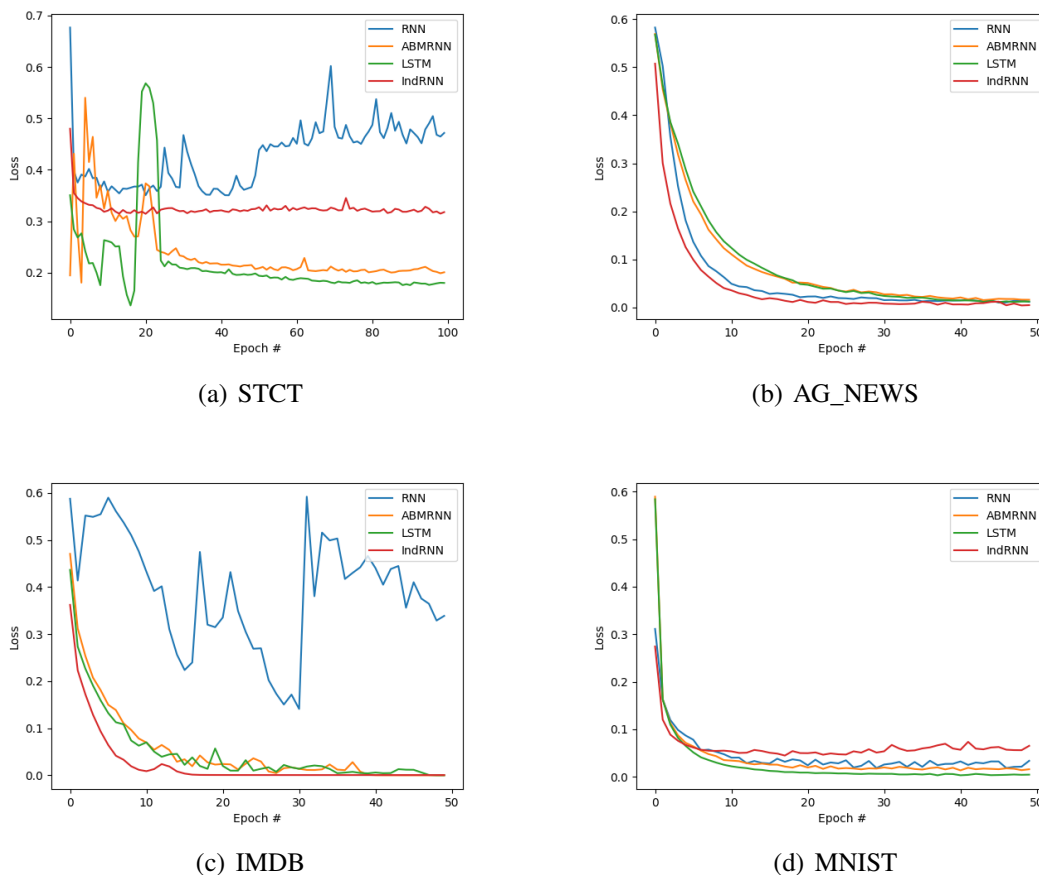


Figure 4.7: Training loss

We also concern about the training loss, because training loss can determine whether the model is converged or not, as well as provide the lower bound to estimate the performance. Those four models are selected because they represent the most typical structures. In Figure 4.7(a) and 4.7(c), plain RNNs keep oscillating, which means it is hard to converge. This is because compared with task AG_NEWS and MNIST, STCT and IMDB are much more complicated, which indicates RNNs cannot handle those hard tasks. However, ABMRNN, LSTM and IndRNN converge after only a few epochs in those four tasks, but LSTM converges slower than IndRNN. Although the training loss of ABMRNN is a little higher than that of LSTM, training loss can only guarantee the lower bound instead of the upper bound with regarding the accuracy rate.

AG's news corpus is a short-term text task which is similar to STCT. The accuracy rate in our model is 86.31%, where outperforms the rest RNN models. However, the 5-layer RNN obtains the lowest accuracy rate (77.76%). The accuracy rate in plain LSTM, 2-layer LSTM and bi-LSTM are 82.33%, 82.27% and 82.13% respectively. With the residual mechanism adopted into LSTM, the performance improves correspondingly (from 82.33% to 84.71%).

In IMDB datasets, with the text length increasing up to 3000, compared with short-term text, the performance is impacted because more redundancies and noises are introduced. We achieve the highest accuracy (90.91%) while the lowest accuracy is only 50% in the 5-layer RNN. Since IMDB is a binary classification task, 50% accuracy rate means blind guess.

In MNIST, RNNs generally cannot perform as good as CNNs. We still apply models in MNIST because this is also a good task to test the robustness and reliability if we regard the images as sequential data. In sequential MNIST, the accuracy rate in 1-D CNN is only 98.01%. However, our model obtains the highest accuracy rate (98.53%) than other models because our model inherits the advantages from both residual network and LSTM.

For better illustrating the improvement of our model, we define Error Rate(ER) = 1 - accuracy and show the results with bar chart. Figure 4.6 shows the ER with four datasets. In STCT and AG's NEWS, ERs are decreased about 503% (from 21.11% to 3.5%) and 62.5% (from 22.24% to 13.69%) respectively. In IMDB and MNIST, we finally improve the performance by 450% (from

50% to 9.09%) and 103% (from 2.99% to 1.47%) respectively.

Regarding recent neural networks towards AG’s NEWS, IMDB and MNIST, all the models become increasingly complex with deeper layers. We illustrate the number of parameters of each model in Table 4.3. The number of parameters in [50] and [51] are 7.8M and 50M respectively. However, our model only contains 0.5M parameters. In AG’s NEWS, [51] and [50] claim that their ER are 13.39% and 10.17% respectively. However, the ER in our model is 13.69%; our model demonstrates high efficiency in training (0.5M vs. 7.8M vs. 50M) and comparatively top performance (13.69% vs. 13.39% vs. 10.17%). Besides, in IMDB datasets, [59] and [60] declare that the ER are 11.52% and 11% respectively while the ER in our model is 9.19%. Furthermore, in [60], the depth of their model is 200 nevertheless ours is only two. In MNIST, although there are numerous famous CNN architectures such as ResNet (25.5M) AlexNet (60M) and VGGNet (138M), the depth of their models are considerably high (up to 1000). They perform from 95% to 97% variously because they mainly focus on the size of 224*224 while the size of the image in MNIST is 28*28. Therefore, [24] introduces a smaller architecture towards MNIST and achieves the state-of-the-art performance (ER = 0.23%). The number of parameters in [24] is 12M whereas ours is 0.5M. As we mentioned before, unlike all the 2D-CNN methods, we consider MNIST as a sequential classification task to test the model regarding robustness and reliability, and we outperform other RNN models.

Model	#parameter
VDCNN[50]	7.8M
CharCNN[51]	50M
ResNet[16]	25M
AlexNet [9]	60M
APNN[24]	12M
VGGNet [61]	138M
ABMRNN	0.5M

Table 4.3: The number of parameters

5. CONCLUSION

Regarding the data fetching, an optimized system has been proposed for defeating the CAPTCHAs including an adaptive length system and an optimal classifier to achieve state-of-the-art performance, which is faster and more accurate. The adaptive length system can perform up to 10.67 times faster than the sequential segmentation, and the successful single character recognition rate is improved variously from 75% to 99% depending on the type of CAPTCHA.

Defeating the CAPTCHAs is also beneficial to improving the safety when we expose the CAPTCHAs' deficiency. Regarding the recognition performance of the classifiers, although TM/OCR is fast and cost-effective, we can only use OCR under ideal conditions, such as twisted-free, noise-free and rotation-free. This is because with the situation getting more sophisticated, the performance in TM/OCR drops dramatically. CNN acts as a more reliable classifier than TM/OCR during the whole test. CNN cannot perform well without sufficient high quality and quantity training data and reliable neural network. The cost to collect the training data in the beginning is huge. We also proposed our method to reduce the collecting cost massively. Another contribution of our work is completely solving the problem of CAPTCHAs in terms of manually collecting sufficient training data, evaluating different existing methods in practice, and combining the optimal method with our proposed algorithm to defeat the CAPTCHAs with state-of-the-art performance.

We have demonstrated the classification task of Chinese short-term text, in the context of public financial news titles. Different features and classifiers are applied and compared, and the baseline performance has been provided. Cross-validation results show that logistic regression and support vector classifier with the *tf-idf* or *CounterVectorizer* feature attain the highest accuracy and are most stable in all circumstances.

An Attention-augmentation Bidirectional Multi-residual Recurrent Neural Network (ABM-RNN) has been proposed to achieve state-of-the-art performance in the proposed classification task. Furthermore, we also test our model in the public tasks. The memory cell limitation in LSTM has been eliminated through a modified attention model. The specific previous time steps

can be identified so that linking to those specific time steps is considerably effective. Moreover, unlike randomly connecting to a previous time step, a multi-residue mechanism has been leveraged to enhance the correlation of current time step and distant previous time steps. In addition, a bidirectional multi-residual mechanism has been proposed which can combine past and future information comprehensively, instead of partially capturing the previous information solely. Last but not the least, the detailed training procedure of ABMRNN has been made available. The results have shown that our model outperforms other RNN models such as plain RNN, the variations of LSTM and IndRNN, and has achieved state-of-the-art performance in the proposed task. In addition to solve the proposed task, we also apply our model in other public tasks and have obtained the competitive performance. Compared with existing models, our model has demonstrated high efficiency in training and has achieved top performance.

Our future work includes applying ABMRNN in other tasks, and further optimizing our model to handle longer sequences.

REFERENCES

- [1] R. Sun, “Introduction to sequence learning,” in *Sequence Learning*, pp. 1–10, Springer, 2000.
- [2] S. Perreault and P. Hébert, “Median filtering in constant time,” *IEEE Transactions on Image Processing*, pp. 2389–2394, IEEE, 2007.
- [3] E. Bursztein, M. Martin, and J. Mitchell, “Text-based CAPTCHA strengths and weaknesses,” in *Proceedings of the 2011 ACM conference on Computer and Communications Security*, pp. 125–138, ACM, 2011.
- [4] Y. Wang and M. Lu, “A self-adaptive algorithm to defeat text-based CAPTCHA,” in *Proceedings of the 2016 IEEE International Conference on Industrial Technology (ICIT)*, pp. 720–725, IEEE, 2016.
- [5] S.-Y. Huang, Y.-K. Lee, G. Bell, and Z.-h. Ou, “An efficient segmentation algorithm for CAPTCHAs with line cluttering and character warping,” *Multimedia Tools and Applications*, vol. 48, no. 2, pp. 267–289, Springer, 2010.
- [6] R. Smith, “An overview of the tesseract OCR engine,” in *Proceedings of 2007 IEEE International Conference on Document Analysis and Recognition*, pp. 629–633, IEEE, 2007.
- [7] Y. Wang, Y. Huang, W. Zheng, Z. Zhou, D. Liu, and M. Lu, “Combining convolutional neural network and self-adaptive algorithm to defeat synthetic multi-digit text-based CAPTCHA,” in *Proceedings of the 2017 IEEE International Conference on Industrial Technology (ICIT)*, pp. 980–985, IEEE, 2017.
- [8] J. P. Lewis, “Fast template matching,” in *Vision Interface*, vol. 95, pp. 15–19, Canadian Image Processing and Pattern Recognition Society, 1995.
- [9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the 2012 Advances in Neural Information Processing Systems (NIPS)*, pp. 1097–1105, Curran Associates, Inc., 2012.

- [10] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” *IEEE transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, IEEE, 1997.
- [11] F. Tao and C. Busso, “Gating neural network for large vocabulary audiovisual speech recognition,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 7, pp. 1286–1298, IEEE, 2018.
- [12] H. Zhang, J. Li, Y. Ji, and H. Yue, “Understanding subtitles by character-level sequence-to-sequence learning,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 616–624, IEEE, 2017.
- [13] Y. Ji, H. Zhang, and Q. J. Wu, “Salient object detection via multi-scale attention CNN,” *Neurocomputing*, vol. 322, pp. 130–140, Elsevier, 2018.
- [14] Y. Zhang and X. Zhang, “Effective real-scenario video copy detection,” in *Proceedings of the 2016 IEEE International Conference on Pattern Recognition (ICPR)*, pp. 3951–3956, IEEE, 2016.
- [15] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, MIT Press, 1997.
- [16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 770–778, IEEE, 2016.
- [17] H. Wu, X. Zhang, B. Story, and D. Rajan, “Accurate vehicle detection using multi-camera data fusion and machine learning,” in *Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3767–3771, IEEE, 2019.
- [18] Y. Wang and F. Tian, “Recurrent residual learning for sequence classification,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 938–943, Association for Computational Linguistics, 2016.

- [19] Y. Zhang, G. Chen, D. Yu, K. Yaco, S. Khudanpur, and J. Glass, “Highway long short-term memory rnns for distant speech recognition,” in *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5755–5759, IEEE, 2016.
- [20] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [21] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, “End-to-end attention-based large vocabulary speech recognition,” in *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4945–4949, IEEE, IEEE, 2016.
- [22] F. Tao, G. Liu, and Q. Zhao, “An ensemble framework of voice-based emotion recognition system for films and tv programs,” *arXiv preprint arXiv:1803.01122*, 2018.
- [23] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 577–585, Curran Associates, Inc., 2015.
- [24] I. Sato, H. Nishimura, and K. Yokoi, “Apac: Augmented pattern classification with neural networks,” *arXiv preprint arXiv:1505.03229*, 2015.
- [25] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, “Building segmentation based human-friendly Human Interaction Proofs (HIPs),” in *Human Interactive Proofs*, pp. 1–26, Springer, 2005.
- [26] X. Ling-Zi and Z. Yi-Chun, “A case study of text-based CAPTCHA attacks,” in *Proceedings of the 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 121–124, IEEE, 2012.
- [27] N. Roshanbin and J. Miller, “A survey and analysis of current CAPTCHA approaches,” *Journal of Web Engineering*, vol. 12, no. 1-2, pp. 1–40, Rinton Press, 2013.

- [28] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using hard ai problems for security," in *Proceedings of the 2003 Advances in Cryptology—EUROCRYPT*, pp. 294–311, Springer, 2003.
- [29] J. Yan and A. S. El Ahmad, "A low-cost attack on a microsoft CAPTCHA," in *Proceedings of the 2008 ACM conference on Computer and Communications Security*, pp. 543–554, ACM, 2008.
- [30] S.-Y. Huang, Y.-K. Lee, G. Bell, and Z.-h. Ou, "A projection-based segmentation algorithm for breaking msn and yahoo CAPTCHAs," in *Proceedings of the 2008 International Conference of Signal and Image Engineering*, Citeseer, IAENG (International Association of Engineers), 2008.
- [31] P. Liu, J. Shi, L. Wang, and L. Guo, "An efficient ellipse-shaped blobs detection algorithm for breaking facebook CAPTCHA," in *Trustworthy Computing and Services*, pp. 420–428, Springer, 2013.
- [32] C. C. Stearns and K. Kannappan, "Method for 2-d affine transformation of images," Dec. 12 1995. US Patent 5,475,803.
- [33] B. Rashidi and B. Rashidi, "Implementation of a high speed technique for character segmentation of license plate based on thresholding algorithm," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 4, no. 12, pp. 9–18, MECS Press, 2012.
- [34] F. Kurugollu, B. Sankur, and A. E. Harmanci, "Color image segmentation using histogram multithresholding and fusion," *Image and Vision Computing*, pp. 915–928, Elsevier, 2001.
- [35] C. A. Glasbey, "An analysis of histogram-based thresholding algorithms," *CVGIP: Graphical Models and Image Processing*, pp. 532–537, Elsevier, 1993.
- [36] R. Smith, D. Antonova, and D.-S. Lee, "Adapting the tesseract open source OCR engine for multilingual OCR," in *Proceedings of the 2009 International Workshop on Multilingual OCR*, p. 1, ACM, 2009.

- [37] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, “Multi-digit number recognition from street view imagery using deep convolutional neural networks,” *arXiv preprint arXiv:1312.6082*, 2013.
- [38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, IEEE, 1998.
- [39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Cognitive Modeling*, vol. 5, no. 3, p. 1, MIT Press, 1988.
- [40] P. Y. Simard, D. Steinkraus, J. C. Platt, *et al.*, “Best practices for convolutional neural networks applied to visual document analysis.,” in *Proceedings of the 2003 IEEE International Conference on Document Analysis and Recognition*, vol. 3, pp. 958–962, IEEE, 2003.
- [41] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Synthetic data and artificial neural networks for natural scene text recognition,” *arXiv preprint arXiv:1406.2227*, 2014.
- [42] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.
- [43] F. Peng, F. Feng, and A. McCallum, “Chinese segmentation and new word detection using conditional random fields,” in *Proceedings of the 2004 International conference on Computational Linguistics*, p. 562, Association for Computational Linguistics, 2004.
- [44] C. Huang and H. Zhao, “Chinese word segmentation: A decade review,” *Journal of Chinese Information Processing*, vol. 21, no. 3, pp. 8–20, Oripobe Inc., 2007.
- [45] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [46] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of Machine Learning Research*, vol. volume 3, pp. pages 993–1022, Jan 2003.

- [47] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proceedings of the 2013 Advances in Neural Information Processing Systems (NIPS)*, pp. 3111–3119, Curran Associates, Inc., 2013.
- [48] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *Proceedings of the 2014 IEEE International Conference on Machine Learning*, pp. 1188–1196, IEEE, 2014.
- [49] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, *et al.*, “API design for machine learning software: experiences from the scikit-learn project,” *arXiv preprint arXiv:1309.0238*, 2013.
- [50] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, “Very deep convolutional networks for text classification,” *arXiv preprint arXiv:1606.01781*, 2016.
- [51] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *Proceedings of the 2015 Advances in Neural Information Processing Systems (NIPS)*, pp. 649–657, Curran Associates, Inc., 2015.
- [52] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.
- [53] F. Tao and G. Liu, “Advanced lstm: A study about better time dependency modeling in emotion recognition,” in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2906–2910, IEEE, 2018.
- [54] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, “Independently recurrent neural network (indrnn): Building a longer and deeper rnn,” in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5457–5466, IEEE, 2018.
- [55] Y. Wang, Z. Zhou, S. Jin, D. Liu, and M. Lu, “Comparisons and selections of features and classifiers for short text classification,” in *Materials Science and Engineering Conference Series*, vol. 261, p. 012018, IOP Publishing, 2017.

- [56] G. D. Forney, “The viterbi algorithm,” *In Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, IEEE, 1973.
- [57] G. Hinton, N. Srivastava, and K. Swersky, “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent,” *Lecture Notes Distributed in CSC321 of University of Toronto*, p. 14, 2012. [online] Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
- [58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, JMLR.org, 2014.
- [59] A. Tripathy, A. Agrawal, and S. K. Rath, “Classification of sentiment reviews using n-gram machine learning approach,” *Expert Systems with Applications*, vol. 57, pp. 117–126, Elsevier, 2016.
- [60] S. Karimi, X. Dai, H. Hassanzadeh, and A. Nguyen, “Automatic diagnosis coding of radiology reports: a comparison of deep learning and conventional classification methods,” *BioNLP 2017*, pp. 328–332, Association for Computational Linguistics, 2017.
- [61] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.