University of Sheffield

# Classifying News Stories by Topic


Ziling Li

*Supervisor:* Mark Stevenson

A report submitted in fulfilment of the requirements
for the degree of data analytics in Computer Science

*in the*

Department of Computer Science

September 4, 2019

## Declaration

All sentences or passages quoted in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

Name: _____

Signature: _____

Date: _____

# Acknowledgments

A huge thank you to my supervisor, Dr. Mark Stevenson, who provided project, experiment data and helpful insight throughout the dissertation project. Thank you also to my family and friends, they provided help when I was doing experiments. Finally, I would like to thank the University of Sheffield and Diamond for providing computer equipment and learning environment and I can use them to run the program and write papers.

# Abstract

The objective in multi-label learning is to train a classifier that can automatically label a data point with the most relevant subset of labels from an extremely large label set (Bhatia et al. 2015). News text categorization is a typical example of multi label classification, each news can be related with many different labels and the number of labels is hard to confirm. The problem of multi label classification has not been solved and it is difficult to directly determine which classifier is real effective, so this paper focus on comparing the performance between traditional machine learning models such as Decision Tree and deep learning models such as Convolutional Neural Network (CNN).

By comparing the result of macro averaging evaluation and micro averaging evaluation, findings show that the overall performance of deep learning classifiers is better than of traditional machine learning models, but some traditional models can still be helpful in this field and their performance is similar to the deep learning models, such as Support Vector Machine (SVM). In addition, the performance of classifiers will decrease when the labels complexity increase, so how to handle massive labels classification would be the future work of this project.

# Contents

# Chapter 1

# Introduction

News plays an important role in transferring current information and keeps readers link with the world, with the improvement of the Internet, electronic newspapers have started as online-only publications. For example, the Southport Reporter published from UK is the only web-based newspaper (Panda & Swain 2011). News and journalism are in a boom period of expansion, the amount of news material produced each day is booming (Russell 2009). The increasing number of electronic documents make text processing more important, in this case, document classification is an important area for research, in order to handle this problem, Natural Language Processing (NLP) and Machine Learning techniques works together to automatically classify (Manikandan & Sivakumar 2018).

Machine learning (ML) is the study of algorithms and statistical models which can perform a specific task by computer systems, it is seen as a subset of artificial intelligence (AI), Machine learning algorithms build a mathematical model based on sample data, which is known as "training data", in order to make predictions or decisions (Bishop 2006). Machine learning algorithms are used in a wide variety of applications, such as email filtering, and computer vision, NLP is also an important application of Machine Learning.

The field of NLP aims to convert human language into a formal representation which is easy for computers to manipulate (Collobert & Weston 2008). Technologies based on NLP are becoming increasingly widespread (Bird et al. 2009). For example, semantic Parsing, question answering, machine translation, text classification and categorization.

For text automatic classification, NLP is the main way to handle it because machine can process text data and automatic classify documents after training. Text classification is the activity of labelling natural language texts with thematic categories from a predefined set, Sebastiani (2002) concludes several traditional methods to handle this problem, including probabilistic classifiers such as Naive Bayes, decision tree classifiers, support vector machine (SVM), and neural networks.

News automatic categorisation is really important, and there are many different type of classifiers can be used, in this case, comparing the performance of different classifiers for news categorisation and find out the most suitable one is also crucial.

## 1.1 Aims

The project will focus on news multi-label automatic categorisation, implement existing traditional algorithms such as decision tree and implement deep learning algorithms such as Recurrent Neural Network (RNN), then apply these algorithms to establish classifiers on different label classes.

In the second stage, evaluate the algorithms and compare their performance such as micro F1 score, precision score and recall score, then analyze their error and try to find some approaches to optimize them.

## 1.2 Objectives

1. Understanding news data, finish data exploratory
2. Pre-processing dataset, divide it into training data and test data, divide labels into three main classes and train classifiers for them separately
3. Train traditional classifiers by tuning the best parameters, train Neural Networks by adjusting super parameters
4. Evaluation of classifiers using methods such as micro F1 and compare different classifiers by their performance
5. Analyze error of models and try to adjust them, then optimize algorithms if possible

## 1.3 Overview

The remaining report is divided into 4 parts. The first part is literature review survey, this part will mainly generalize the process of multi label classification, then introduce some traditional algorithms and deep learning methods; The second part is methodology, this part is going to analyze problems of the project and explore data information, then do some model designation and implementation ; The third part is talking about final results and discussion, it is split into results of traditional models and deep learning models, it mainly focus on comparison of their performance on different classes; The final part is conclusion, which will explain what have been found in the experiments and the plan for future work.

# Chapter 2

# Literature Review

This chapter can be divided into four parts. The first part introduce the multi label classification, including the purpose of it and the main methods to handle it. The second part focus on how traditional machine learning can be used to handle multi label classification problem, including data preprocessing, feature selection, imbalanced data handling, traditional classifiers and evaluation. The last part is going to talk about neural network and deep learning, including what is deep learning and how deep learning can be used for text classification, including multilayer perceptron, fasttext convolutional neural network and recurrent neural network.

## 2.1 Multi label Classification

The aim of multi-label learning is to train classifiers that can automatically allocate the most relevant subset of labels to the relevant data points from an extremely large label set (Bhatia et al. 2015). The number of predicted labels is not unstable, it means classifiers should predict any tags that might match the text, instead of several fixed number of labels.

The existing methods for multi-label classification can be divided into two main categories: problem transformation methods and algorithm adaptation methods. Problem transformation methods are methods that transform multi-label classification problem into several single-label classification problems, the number of subclassification problems depend on the number of labels in the original data set. Algorithm adaptation methods extends specific machine learning algorithms in order to handle multi-label data directly, instead of transferring problems into several classifiers (Tsoumakas et al. 2006). The difference of these two approaches is that problem transformation methods are actually combination of many binary classifiers when algorithm adaptation methods are single classifiers which can handle multi labels directly.

Multi label problem means each point in the dataset is associated to a set of relevant labels, the main difficulty of multi-label classification is that the learning task would be to output a set of labels whose size is unknown in advance, for example, one news document may be

talking about food, meat and finance, although another news document would concern only on food and meat. The multi label problem could be solved by using some machine learning algorithms, for example, a ranking based SVM is proposed to deal with multi-label problem, the key idea of their system is minimizing the ranking loss between prediction and true labels (Elisseeff & Weston 2002), their system is actually an algorithm adaptation method, which calculate the probabilities of label matches the specific document and return the most related several labels.

## 2.2 Traditional Machine Learning Classification

Traditional text classification works mainly focus on three topics: feature engineering, feature selection and using different types of machine learning algorithms. It can be divided into several stages. Firstly, preprocessing documents and indexing them; Secondly, select proper features; Then build classifier by features in training set; Finally, evaluate the classifier by performance measurement(Korde & Mahender 2012). In the next part, detailed approaches will be introduced in these stages.

### 2.2.1 Preprocessing

Text normalization and text removal are useful approaches for text preprocessing(Carvalho et al. 2007). Normalization can be considered as using one specific representation to represent all occurrences of words with similar meaning, lowercase conversion and punctuation removal are two main usage of text normalization, they are helpful because the normalized format is easier for machine to read instead of messy data, for example, chequebook and cheque book can all be normalized as cheque book. Text removal is the removal of elements from the original text, in order to remove noise data, which is non-meaningful for the whole text, such as prepositions and particles, they can not help machine recognize the meaning of text. Moreover, stemming is also important in text categorization system, it is the process of extracting stem by removing the affixes from the word and extracting the word, which can guaranteed text consistency.

### 2.2.2 Feature Selection

The basic idea of TFIDF (term frequency inverse document frequency) is an important statistics method, which can assess the importance of a word in a collection of documents or corpora by determining whether a word is distinguishing enough for classification. It represents each document $d$ as a vector $\vec{d} = \left(d^{(1)}, \ldots, d^{(|F|)}\right)$ in a vector space so that documents with similar content will have similar vectors, each dimension of the vector space represents a word, the dimension should be the number of unique words in the corpus if it does not go through dimension reduction. The value of $d^{(i)}$ is calculated by the product of term frequency $TF(w, d)$ and inverse document frequency $IDF(w, |D|)$, $TF(w, d)$ is the times word $w$ occurs

in document $d$, and , $IDF$ is calculated by equation 2.1.

$$IDF(w, |D|) = \log\left(\frac{|D|}{DF(w)}\right) \tag{2.1}$$

Where the document frequency $DF(w)$ is the documents in which the word $w$ occurs, and $|D|$ is the total number of documents, the inverse document frequency of a word is low if it occurs in many documents so it can help to filter out key words in documents. The TFIDF value $d^{(i)}$ is calculated by equation 3.6.

$$d^{(i)} = TF\left(w_i, d\right) \cdot IDF\left(w_i\right) \tag{2.2}$$

Where $d^{(i)}$ is called the weight of word $w_i$ in document $d$. The higher the TFIDF is, the more important this word is in the document (Joachims 1996).

Besides TFIDF, there are many feature selection methods, Rogati & Yang (2002) compare several major feature selection methods for text classification, including information gain, document frequency and $\chi^2$ statistic ,by putting them into four different classification algorithms and comparing the performance among different methods, they obtain that the method based on $\chi^2$ statistic is better than others.

### 2.2.3 Imbalanced data classification

Unbalanced data sets often appear in real life, it is difficult for machine learning algorithms to handle this problem because rare data is more valuable and it is difficult for machines to learn from a small amount of data. The most obvious characteristic of imbalanced data is the skewed data distribution, Sun et al. (2009) propose some solutions to handle this problem, including data-level approaches, algorithm-level approaches and cost-sensitive learning.

At the data level, resampling is the most popular methods, such as oversampling the small class or undersampling the prevalent class. It is helpful for imbalanced data classification but there are also some drawbacks, oversampling simply repeats the positive cases and it would cause overfitting the positive example, undersampling discards many negative data and it would cause large deviations in the model.

At the algorithm level, a common strategy is to choose an appropriate inductive bias, such as to adjust the probabilistic estimate. For example, modify the threshold of the model and make models more sensitive to fewer labels.

Cost sensitive classification considers the varying costs of different misclassification types. For example, adjusting the parameter weight of the model, when using the cost function, increasing the weight of the rare samples and reduce the weight of the frequently occurring sample.

### 2.2.4 Classifier Algorithms

This part will introduce several popular machine learning classification models, including Decision Tree, Naive Bayes, K nearest neighbor (KNN), Support Vector Machine (SVM), logistic regression and random forest.

#### Decision Tree

Decision tree is constructed by leaf nodes and internal node, leaf nodes indicate classes or labels, internal nodes specifies some specific features. At each node, the outcome is determined and attention shifts to the root of the subtree corresponding to this outcome, the class of the case is predicted to be that recorded at the leaf when the attention shifting process finally leads to a leaf (Quinlan 2014).

Decision tree creates a hierarchical partitioning of the data, the partitioning at each level is created with the split criterion, the overall approach is to recursively split the training data and to maximize the discrimination among the different classes over different leaf nodes(Aggarwal 2014). This discrimination is maximized when the level of skew among different classes in a given node is maximized. In order to quantify the skew, measurement such as gini index and entropy is used.

If there are $k$ different classes in a node $N$, $p_1, ..., p_k$, then the gini index $G(N)$ of the node $N$ is defined in 2.3, the skew becomes greater with the decrease of the value of $G(N)$. Another measure approach is the entropy $E(N)$, it is calculated in equation 2.4. The smaller the entropy, the greater the skew. Gini index and entropy evaluate the quality of a node in terms of its level of discrimination between the different classes.

$$G(N) = 1 - \sum_{i=1}^{k} p_i^2 \tag{2.3}$$

$$E(N) = -\sum_{i=1}^{k} p_i \cdot \log(p_i) \tag{2.4}$$

Decision tree classifies a document by recursively testing the weights that the attributed labeling the internal nodes have in document vector, until a leaf reached. Harrag et al. (2009) use decision tree to classify Arabic text documents and achieve a high precision, over 0.9. Previous research shows that decision tree can be used for text classification.

#### Naive Bayes

Dai et al. (2007) propose a transfer learning algorithm for text classification, an EM-based Naive Bayes classifiers, the Bayes Classifier they used is a simple Bayesian classification algorithm, it estimates the conditional probability $P(c|d)$ which represents the probability of document $d$ belongs to class $c$, according to Bayes rule, it can be calculated as 2.21.

$$P(c|d) \propto P(c) \cdot P(d|c) \tag{2.5}$$

Naive Bayes classifier assumes that words in documents are conditionally independent by given the class value, so 2.21 could transfer to 2.22.

$$P(c|d) \propto P(c) \prod_{w \in d} P(w|c) \tag{2.6}$$

### K nearest neighbor

Trstenjak et al. (2014) present the possibility of using KNN algorithm with TFIDF method for text classification, firstly, represent documents by using TFIDF representation, then select a specific K value to indicate the number of documents from the collections which is closest to the selected document, the classifier determinate the vector distance between the documents by using equation 2.7.

$$d(x, y) = \sqrt{\sum_{r=1}^{N} (a_{rx} - a_{ry})^2} \tag{2.7}$$

Where $d(x, y)$ is the distance between two documents, N is the number of unique words in the collection, $a_{rx}$ is a weight of the term r in document x, $a_{ry}$ is a weight of the term $r$ in document $y$. If the Euclidean distance between documents $x$ and $y$ becomes smaller, the similarity of $x$ and $y$ becomes higher.

KNN classifiers can be improved in many ways. The calculation complexity of traditional KNN is huge because it uses all training samples for classification, in this background, Yong et al. (2009) improve the KNN algorithm by combining it with k-means clustering. Firstly, compress the training sets by deleting samples near the border in order to eliminate the multi-peak effect of the training samples; Secondly, cluster the training sets by k-means algorithm and make all cluster centers as the new training samples; Thirdly, use a weight value to indicate the importance of each training sample; Finally, accomplish KNN classifier by using the modifies samples.

Besides, Han et al. (2001) argue that considering each word to be either present or absent in a document is questionable, so they proposed Weight Adjusted k-Nearest Neighbor (WAKNN) classification algorithm based on the traditional KNN classifier, when classifying a test document, firstly find out the k nearest neighbors of the test document by using the weighted cosine similarity with the weight learned from WAKNN, then sum up these similarities to the k neighbors according to their class labels, finally classify the test documents according to the class with the most similarity sum value.

### Support Vector Machine

(Suykens & Vandewalle 1999) demonstrate the procedure of SVM, he proposes that the SVM classifier is constructed of the form shown in 2.8,

$$y(x) = sign \left[ \sum_{k=1}^{N} \alpha_k y_k \psi(x, x_k) + b \right] \tag{2.8}$$

where $\alpha_k$ are positive real constants and $b$ is a real constant, and $\psi(x, x_k) = x_k^T x$ for linear SVM, $\psi(x, x_k) = \exp\left\{-\|x - x_k\|_2^2 / \sigma^2\right\}$ for radial basis function (RBF) SVM, where $\sigma$ is a constant, the classifier is constructed as shown in 2.9.

$$y_k \left[w^T \varphi(x_k) + b\right] \geq 1 - \xi_k \tag{2.9}$$

where $\xi_k \geq 0, \quad k = 1, \ldots, N$ and $\varphi(\cdot)$ is a nonlinear function which can map the input space into higher dimensional space.

SVM can be helpful in classification field, many researchers train SVM models for text classification tasks. For example, Dilrukshi et al. (2013) use SVM to automatically classify short Twitter news which are manually labeled into 12 groups, the SVM model supports high dimensional data and is able to find the global minimum instead of local minimum, its performance is good for most groups, but his system is not perfect to handle all categories.

### Logistic regression

(Kleinbaum et al. 2002) explain the logistic function, $f(z) = \frac{1}{1+e^{-z}}$, which describes the mathematical form on which the logistic model is based. They plotted the values of this function $f(z)$ in figure 2.1.

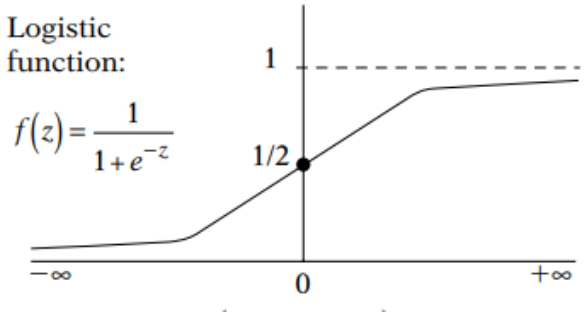



**Figure 2.1:** *logistic function*

We can see that $z$ varies from $-\infty$ to $\infty$ in x-axis, on the right side, if $z$ equals $\infty$, then $f(z)$ is almost near 1. In this way, the range of $f(z)$ is from 0 to 1, regardless of the value of $z$. The logistic model, therefore, is set up to ensure that whatever estimate of value we get, it will always be threshold between 0 and 1.

Hosmer Jr et al. (2013) demonstrate the mathematical points about logistic regression. Based on the standard linear regression, take $f(x) = \mathbf{w}^\top \phi(\mathbf{x})$, a logistic regression should be shown in 2.10.

$$\log \frac{\pi}{(1-\pi)} = \mathbf{w}^\top \phi(\mathbf{x}) \tag{2.10}$$

By setting a inverse link function $g^{-1}(\pi) = \mathbf{w}^\top \phi(\mathbf{x})$, then use $\pi$ to represent it as shown in 2.11.

$$\pi(\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp\left(-\mathbf{w}^\top \phi(\mathbf{x})\right)} \tag{2.11}$$

Next, apply this to objective function which is calculated in 2.12.

$$E(\mathbf{w}) = -\sum_{i=1}^{n} y_i \log g\left(\mathbf{w}^\top \phi\left(\mathbf{x}_i\right)\right) - \sum_{i=1}^{n} \left(1 - y_i\right) \log\left(1 - g\left(\mathbf{w}^\top \phi\left(\mathbf{x}_i\right)\right)\right) \tag{2.12}$$

Finally, minimize this objective by differentiating with respect to the parameters of $\pi(\mathbf{x}, \mathbf{w})$. According to these equations, logistic regression is therefore a generalized linear regression analysis model, it can be used for binary classification, it can also be used for multi-classification.

#### Random forest

Random forest is a classifier which is consisting of a collection of tree-structured classifiers $\{h\left(\mathbf{x}, \Theta_k\right), k = 1, \ldots\}$ where $\{\Theta_k\}$ are independent distributed random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$ (Breiman 2001).

There is a $K$ classifiers $h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_K(\mathbf{x})$, with the training set drawn at random from the distribution of the random vector $X$ and $Y$, define the margin function as,

$$mg(\mathbf{X}, Y) = av_k I\left(h_k(\mathbf{X}) = Y\right) - \max_{j \neq Y} av_k I\left(h_k(\mathbf{X}) = j\right) \tag{2.13}$$

Where $I(\cdot)$ is the indicator function. The margin measures the extent to which the average number of votes at $\mathbf{X}, Y$ for the right class exceeds the average vote for any other class. The larger the margin, the more confidence in the classification. The generalization error is given by,

$$PE^* = P_{\mathbf{X},Y}(mg(\mathbf{X}, Y) < 0) \tag{2.14}$$

Random forest is a good algorithm for classification. Pal (2005) compare the performance between random forest and SVM, they find that random forest classifier performs equally well to SVM in terms of classification accuracy and training time, but the number of userdefined parameters required by random forest classifiers is less than the number required for SVM, so it is easier to define a random forest classifier than defining a SVM classifier.

### 2.2.5 Evaluation

Evaluation is important in estimating the performance of Machine Learning algorithms(Goutte & Gaussier 2005), considering binary label $l$ as the true label of object, and assignment $z$ as the prediction, table 2.1 shows the confusion table.

| | | assignment $z$ | |
|---|---|---|---|
| | | + | - |
| label $\ell$ | + | TP | FN |
| | - | FP | TN |

**Table 2.1:** *precision and recall*

where "+" means the label is positive or the classifiers predict this label as positive, "-" means the label is negative or the classifiers predict this label as negative, as for the values in the table, TP stands for true positive, TN stands for true nagetive, FP stands for false positive, and FN stands for false nagetive, they can be helpful to calculate precision score $p$ and recall score $r$, which is shown in 2.15:

$$p = \frac{TP}{TP + FP} \qquad r = \frac{TP}{TP + FN} \tag{2.15}$$

It is not objective to judge the performance of the model based solely on these two values. In actual applications, Simply pursuing the promotion of precision and score is not very useful for model promotion, instead of them, F1 score is calculated by combining these two scores, F1 can comprehensively evaluate the performance of a model and the detailed calculation of F1 is shown in 2.16:

$$F_1 = \frac{2pr}{p + r} \tag{2.16}$$

Receiver operation characteristic (ROC) curve is also a good way of visualising performance of classifiers in order to select a suitable model (Bradley 1997). It is drawn based on the real category and prediction probability of the sample, there are some evaluation equation in 2.17, 2.18 and 2.19.

$$Accuracy(1 - Error) = \frac{(T_p + T_n)}{(C_p + C_n)} = P(C) \tag{2.17}$$

$$Sensitivity(1 - \beta) = \frac{T_p}{C_p} = P\left(T_p\right) \tag{2.18}$$

$$Specificity(1 - \alpha) = \frac{T_n}{C_n} = P\left(T_n\right) \tag{2.19}$$

Then based on these figures, the value of Area Under ROC (AUC) can be calculated as the final evaluation, it is actually the area under the ROC curve, the derived formula is shown in 2.20

$$\text{AUC} = \sum_i \left( (1 - \beta_i \cdot \Delta\alpha) + \frac{1}{2}(\Delta(1 - \beta) \cdot \Delta\alpha) \right) \tag{2.20}$$

where $\Delta(1 - \beta) = (1 - \beta_i) - (1 - \beta_{i-1})$ and $\Delta\alpha = \alpha_i - \alpha_{i-1}$. When compared to overall accuracy, in some cases, AUC can perform better as a classification performance measure, for example, it is invariant to prior class probabilities, but there is still no best way to evaluate a classifier's performance.

With compared to single label classification, in multi label classification, the evaluation is more complicated, because a result can be fully correct, partly correct, or fully incorrect (Boutell et al. 2004).

Take an example belonging to classes $c_2$ and $c_2$, the prediction result could be one of the following answer:

1. $c_1, c_2$ (correct)
2. $c_1$ (partly correct)

3. $c_1, c_3$ (partly correct)
4. $c_1, c_3, c_4$ (partly correct)
5. $c_3, c_4$ (incorrect)

The above five results are different from each other in the degree of correctness. So they propose the special Base-class evaluation for multi label classification. Let $Y_x$ be the set of true labels for example $x$ and $P_x$ be the set of predicted labels from classifier $h$, let $H_x^c = 1$ if $c \subset Y_x$ and $c \subset P_x$, 0 otherwise. Likewise, let $Y_x^c = 1$ if $c \subset Y_x$, 0 otherwise, and let $P_x^c = 1$ if $c \subset P_x$, 0 otherwise, let $C$ be the set of base classes. Then base-class recall and precision on data set, D, are defined as following functions 2.21, 2.22 and 2.23:

$$Recall(\mathrm{c}) = \sum_{\mathrm{x}\in\mathrm{D}} \mathrm{H}_{\mathrm{x}}^{\mathrm{c}} \sum_{\mathrm{x}\in\mathrm{D}} \mathrm{Y}_{\mathrm{x}}^{\mathrm{c}} \tag{2.21}$$

$$Precision(\mathrm{c}) = \sum_{\mathrm{x}\in\mathrm{D}} \mathrm{H}_{\mathrm{x}}^{\mathrm{c}} \sum_{\mathrm{x}\in\mathrm{D}} \mathrm{P}_{\mathrm{x}} \tag{2.22}$$

$$Accuracy_{\mathrm{D}} = \sum_{\mathrm{x}\in\mathrm{D}} \sum_{c\in\mathrm{C}} \mathrm{H}_{\mathrm{x}}^{\mathrm{c}} \max \sum_{\mathrm{x}\in\mathrm{D}} \sum_{\mathrm{c}\in\mathrm{C}} \mathrm{Y}_{\mathrm{x}}^{\mathrm{c}}, \sum_{\mathrm{x}\in\mathrm{D}} \sum_{\mathrm{x}\in\mathrm{C}} \mathrm{P}_{\mathrm{x}}^{\mathrm{c}} \tag{2.23}$$

Hamming loss can also be used to evaluate classification tasks (Norouzi et al. 2012). Hamming distance is a natural similarity measure on binary codes. In classification tasks, the smaller the hamming loss, the better the model. The calculation of hamming loss is shown in 2.24.

$$HammingLoss = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathrm{XOR}\left(Y_{i,j}, P_{i,j}\right)}{L} \tag{2.24}$$

Where $N$ is the amount of samples, $L$ is the amount of labels, $Y_{i,j}$ is the $j$th true value of $i$th prediction result, $P_{i,j}$ is the $j$th prediction value of $i$th prediction result,$XOR$ is the special calculation, $\mathrm{XOR}(0,1) = \mathrm{XOR}(1,0) = 1, \mathrm{XOR}(0,0) = \mathrm{XOR}(1,1) = 0$.

## 2.3 Neural Network and Deep Learning

Deep learning dramatically improved the state-of-the-art in many machine learning fields, it discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer(LeCun et al. 2015).

The deep learning methods introduced here including multilayer perceptron, fasttext, CNN, RNN and RCNN.

### 2.3.1 Multilayer perceptron

Deep feedforward network, which is also called multi layer perceptron, is combined by many different functions. For example, there are three functions $f^{(1)}, f^{(2)}$ and $f^{(3)}$, they are combined in a chain $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, this chain structure is the most popular structure in neural network. In this situation, $f^{(1)}$ is called the first layer, $f^{(2)}$ is called second layer, the length of this chain is called depth of neural network model. The last layer of deep feed forward network is called output layer. In the training process, we match $f(x)$ with the label of each instance $x$. This network is called neural network because it is innovated based on neural science(Goodfellow et al. 2016).

Single layer networks are based on a linear combination of the input variables which is transformed by a non-linear activation function. More general functions could be constructed by considering networks having successive layers of processing units, with connections running from every unit in one layer to every unit in the next layer. Assuming there are $d$ inputs, $M$ hidden units and $c$ output units in a layered network. The output of the $j$th hidden unit is obtained by first forming a weighted linear combination of the $d$ input values, and adding a bias, see in function 2.25,

$$a_j = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.25}$$

Here $w_{ji}^{(1)}$ denote a weight in the first layer, going from input $i$ to hidden unit $j$, and $w_{j0}^{(1)}$ denotes the bias for hidden unit $j$. Set an extra input variable $x_0$ whose value is set as 1, then can rewrite function 2.25 to function 2.26

$$a_j = \sum_{i=0}^{d} w_{ji}^{(1)} x_i \tag{2.26}$$

The activation of hidden unit $j$ is then obtained by transforming the linear sum in 2.26 using an activation function $g(\cdot)$ to give equation 2.27.

$$z_j = g\left(a_j\right) \tag{2.27}$$

The output of the network are obtained by transforming the activation of the hidden units using a second layer of processing elements. For each output unit $k$, we construct a linear combination of the outputs of the hidden units of the form, see in function 2.28.

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.28}$$

The bias would be absorbed into the weights again, see in function 2.29.

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j \tag{2.29}$$

The activation of the $k$th output unit is then obtained by transforming this linear combination using a non-linear activation function, as shown in equation 2.30.

$$y_k = \widetilde{g}\left(a_k\right) \tag{2.30}$$

Here using notation $\widetilde{g}(\cdot)$ for the activation function of the output units to emphasize that this need not be the same function as used for the hidden units. Combine 2.26, 2.27, 2.29 and 2.30 as the final function 2.31.

$$y_k = \tilde{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right)\right) \tag{2.31}$$

If the activation functions for the output units are taken to be linear, then $\tilde{g}(a) = a$, this functional form is a special case of the generalized linear discriminant function, in which the basis functions are given by the particular function $z_j$ defined by equations 2.26 and 2.27 (Bishop et al. 1995).

### 2.3.2 Fasttext

Neural network achieve a good performance in NLP, but it costs too much time. Fasttext, which is proposed by Facebook, can finish projects quickly in many large corpus, it can be used for sentiment analysis, tag prediction and some other text projects, it is actually a simple baseline method for deep learning classification. (Joulin et al. 2016).

The most special process of Fasttext is that the word features can be averaged to represent sentences. For the entire process, The first step of fasttext is to convert texts into vectors, then use them as input for a linear classifier and then feed to the hidden layer, this hidden is the probability distribution of the predefined class, and finally use softmax to calculate the probability. Besides, in order to improve efficiency, Fasttext use stochastic gradient descent to train the classifier and use hierarchical softmax and n-gram to optimize in order to get better performance.

There is the model architecture of fasttext with N gram features $x_1, \ldots, x_n$, the features are embedded and averaged to form the hidden variables, see in figure 2.2.

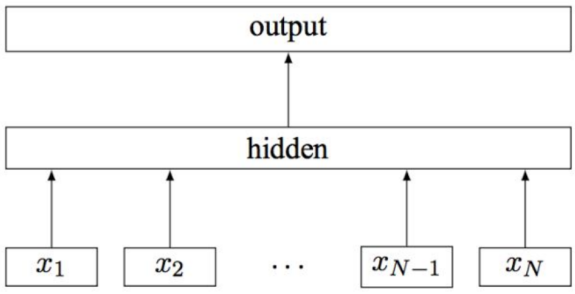



**Figure 2.2:** *fasttext architecture*

Fasttext can be wisely used as a baseline method in deep learning field. Yang et al. (2018) compare the performance of classification task between CNN and Fasttext, they argue that the classification performance of fasttext is very similar to the CNN classifier because they find that the performance difference between CNN and fasttext is not statistically significant after statistic analysis. Although Fasttext shows similar performance with CNN classifiers, it uses bag of words that is invariant to the words ordering. Therefore, to take the order information into network, a bag of N-grams are used as additional features by Fasttext. On the contrary, the CNN classifiers capture the local word ordering using existing word embedding models and therefore does not need to learn N-gram features. In particular, the CNN hyperparameters, such as the filter sizes and stride, provide more flexibility on how to capture the word content information and words order information. So in their research, there is no obvious winner among these deep learning methods.

### 2.3.3 Convolutional neural network

Convolution is a mathematical operation of two real functions,the first parameter of convolution is input, which is also called feature mapping, the second is called kernel function. In machine learning, multi dimensional array is common input, kernels are the parameters after learning algorithm, they are usually multi dimensional array. We can do convolutional operation in many dimensions at one time, for example, take a 2 dimensional picture $I$ as input, put 2 dimensional kernel $K$ on it, we can get equation 2.32.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.32}$$

convolutions are commutative, it can derived in to equation 2.33,

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.33}$$

Convolutional operation help improve machine learning system by sparse interactions, parameter sharing, equivariant representations. For sparse interactions, the size of convolutional kernel is much less than the size of input. For example, networks use small kernel which only contains several pixels to check some small and meaningful features, instead of using thousands of pixels. Compared to full connected network, in deep convolutional network, the units in deep network may interact with most of input indirectly. For parameter sharing, this network use same parameters in many functions in one model, each element of kernel can work on each situation in the input layer. The special function of parameter sharing make neural network owns equivariance, if the input of functions change, the output will change in the same way.

In the convolutional networks, there are three important layer, firstly, generate linear activation response by computing multi convolutions parallelly, then this response will pass a non linear activation function, and finally, pool function will adjust the final output. Pooling refers to the aggregation statistics of features at different locations. For example, max pooling functions calculate the maximum value in the related matrix area. When the input shifts, most of the output after the pooling function will not change a lot(Goodfellow et al. 2016).

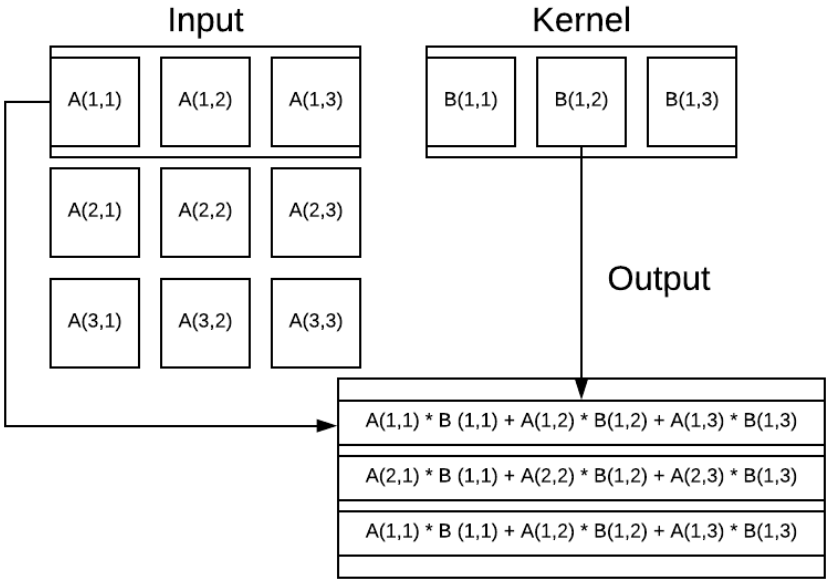



**Figure 2.3:** *example of 2-d convolutional operation, no flipped kernel*

### 2.3.4 Recurrent neural network

RNNs are powerful for sequential data such as text, their hidden state is a function of all previous hidden states. Given an input sequence $x = (x_1, \ldots, x_T)$, a standard RNN computes the hidden vector sequence $h = (h_1, \ldots, h_T)$ and output vector sequence $y = (y_1, \ldots, y_T)$ by iterating the following equations from $t = 1$ to $T$, see in equations 2.34 and 2.35:

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right) \tag{2.34}$$

$$y_t = W_{hy}h_t + b_y \tag{2.35}$$

where the $W$ terms denote weight metrices, the $b$ terms denote bias vectors, $\mathcal{H}$ is the hidden layer function, it is usually a sigmoid function.

Long Short Term Memory (LSTM) used memory cells to store information, it is better to find and exploit long range context. The internel structure is shown in figure 2.4

$\mathcal{H}$ is implemented by 5 functions, see in 2.36, 2.37, 2.38, 2.39 and 2.40.

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \tag{2.36}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \tag{2.37}$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \tag{2.38}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \tag{2.39}$$

$$h_t = o_t \tanh\left(c_t\right) \tag{2.40}$$

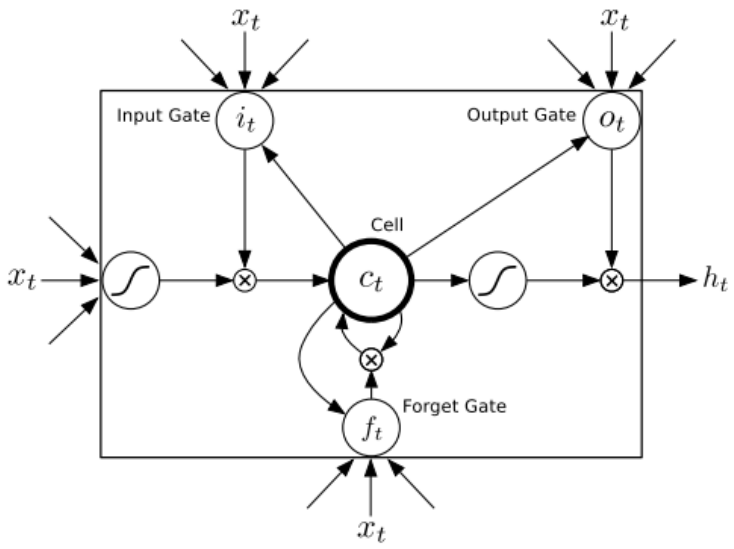



**Figure 2.4:** *LSTM structure*

where $\sigma$ is the logistic sigmoid function, and $i, f, o$ and $c$ are respectively the input gate, forget gate, output gate and cell activation vectors, all of which are the same size as the hidden vector $h$ (Graves et al. 2013).

Lee & Dernoncourt (2016) introduce a model based on RNNs and CNNs for sequential single label short-text classification, they compare the performance between RNNs, CNNs and some other traditional models such as SVM, the conclusion is that the CNNs outperform than other models in two dataset and RNN outperform than other models in one dataset. They demonstrate that adding sequential information improves the quality of the predictions, and the performance depends on what sequential information is used in the model.

### 2.3.5 Recurrent convolutional neural network

Lai et al. (2015) apply a recurrent structure to gain contextual information when machines learn word representations, which may introduce less noise compared to single RNN or single CNN. They also employ a max-pooling layer that automatically judges which words play key roles in text classification to capture the key components in texts. The RCNN model they proposed outperforms CNN and RNN using four different text classification datasets, where the number of labels in these four datasets are less than 20, and they actually compare algorithms for multi class with single label classification problem.

In their RCNN model, a bidirectional recurrent neural network is used to capture the contexts. The calculation of the context including the left side context and the right side context of $w_i$, by defining $e(w_{i-1})$ as the word embedding of $w_{i-1}$, $c_l(w_{i-1})$ as the left context of $w_{i-1}$, $\mathbf{W^{(l)}}$ as a matrix that transforms the context into the next hidden layer, and $\mathbf{W^{(sl)}}$ as a matrix that is used to combine the semantic of the current word with the left context of next

word, in addition, set $f$ as a nonlinear activation function, then the result of $c_l(w_i)$ could be calculated in equation 2.41:

$$c_l\left(w_i\right) = f\left(W^{(l)}c_l\left(w_{i-1}\right) + W^{(sl)}e\left(w_{i-1}\right)\right) \tag{2.41}$$

Calculation of $c_r(w_i)$ is in the similar way, it is the left context of $w_{i-1}$, see in equation 2.42:

$$c_r\left(w_i\right) = f\left(W^{(r)}c_r\left(w_{i-1}\right) + W^{(sr)}e\left(w_{i-1}\right)\right) \tag{2.42}$$

The representation of $x_i$ for word $w_i$ is shown in equation 2.43, it is actually the combination of $c_l\left(w_i\right)$, $e\left(w_i\right)$ and $c_r\left(w_i\right)$.

$$x_i = \left[c_l\left(w_i\right); e\left(w_i\right); c_r\left(w_i\right)\right] \tag{2.43}$$

Then apply the linear function tanh, which is the activation function, this function can transfer value of $x_i$ into between $-1$ and 1, this result can be used in the next layer by equation 2.44:

$$y_i^{(1)} = \tanh\left(W^{(1)}x_i + b^{(1)}\right) \tag{2.44}$$

In the convolutional layer, after all representation of words are calculated, apply a max-pooling layer, see in equation 2.45:

$$y^{(2)} = \max_{i=1}^{n} y_i^{(1)} \tag{2.45}$$

After that, there is an output layer, which is called fully connected layer, for prediction, see in equation 2.46:

$$y^{(3)} = W^{(2)}y^{(2)} + b^{(2)} \tag{2.46}$$

Finally, apply softmax layer in order to transfer numbers into probabilities, the probability $p_i$ is calculated in equation 2.47:

$$p_i = \frac{\exp\left(y_i^{(3)}\right)}{\sum_{k=1}^{n}\exp\left(y_k^{(3)}\right)} \tag{2.47}$$

The combination of RNN and CNN can be widely used in classification projects. For example, in order to handle the problem of encoding the intrinsic relations between sentences in the semantic meaning of a document, Tang et al. (2015) firstly use CNN or LSTM to learn sentence representation, then the semantics of sentences and their intrinsic relations are encoded in document representation with gated recurrent neural network. They argue that gated recurrent neural network dramatically outperforms standard recurrent neural network in document modeling for sentiment classification.

### 2.3.6 Pretrained models

The use of language model pretrained methods has achieved a good improvement in many NLP tasks. By using pretrained ideas, the model would be initialized with this pretrained set of parameters such as word embedding, instead of random initialization.

Bidirectional Encoder Representations from Transformers (BERT) is a new language representation model which is proposed by Google, it is the pretrained deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. This language representation model is trained with oversized data and huge computational resources, it is state of the art for many natural language processing tasks. Pretrained BERT representation can be fine-tuned according to just an extra output layer, and it can be widely used in question answering, or some other NLP projects (Devlin et al. 2018).

Word representation technology can transform natural language words into dense vectors, and words with similar semantically have similar vector representations. Global Vectors for Word Representation (GloVe) is a word representation tool based on count based overall statistics, it can transfer a word into a vector consisted of float numbers, these vectors represent some semantic features of relationship between words, such as word similarity and analogy. According to the calculation of vectors such as cosine similarity, the semantic similarity between two words could be used in actually NLP tasks. GloVe is a model for the unsupervised learning of word representations that outperforms other models on word analogy, word similarity, and named entity recognition tasks (Pennington et al. 2014).

## 2.4 Summary

This part mentions what is multi label classification and the methods for solving multi label classification tasks. First of all, many traditional machine learning methods are proposed, from preprocessing and feature selection to specific classifier algorithms, the algorithms includes Decision Tree, KNN, SVM and some other methods. Then, with the development of deep learning, many algorithms based on deep learning are proposed, for example, Fasttext, CNN and some other algorithms, they can be used as classifiers.

From traditional machine learning methods to deep learning methods, until now, many classifiers can be used in text classification field and multi label classification field. The next chapter will mainly talk about the actual problem in this project and how to solve it by different models.

# Chapter 3

# Methodology

The previous section investigated the methods for text automatic classification, the finding from literature is that both traditional machine learning algorithms and deep learning methods can be used to classify text. Comparing the performance of these models is important because choosing what classifier could be better for multi label classification is a great beginning.

This part will be divided into 4 parts. The specific problem of this project will be discussed in the first part. The second part will describe the news dataset, including information of news text, news labels and some other features. The third part introduce the model designation of traditional machine learning methods and deep learning methods, and it also mention the final evaluation methods. The final part is talking about implementation of models.

## 3.1 Problem

In Reuters Corpus, news stories cover a range of topic such as politics and business. The aim of this project is to develop a system that can analyze news stories and determine the topic it discusses. The project will make use of the news stories from Reuters Corpus, which have been manually labelled with topics. Performance of the implemented approaches will then be evaluated by comparing the predicted topics against the manually labeled ones.

This task is a multi label classification problem. Given a single news without labels, the classifier should predict all possible labels of this news. The training data is shown in equation 3.1:

$$D_{train} = \{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^M, \mathbf{y}^M)\} \tag{3.1}$$

Where $M$ is the amount of news, $\mathbf{x}^m$ contains the text information of $m$th news ($1 <= m <= M$), $\mathbf{y}^m$ contains the labels of the related news. For example, take news in figure 3.1 as the $m$th news, it is clear that the different infomation are stored in different part in this XML file, so crucial content such as news text should be extracted from original files firstly. After extraction, the content of text and its related labels could be seen in equation equation 3.2 and equation 3.3.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<newsitem itemid="2533" id="root" date="1996-08-20" xml:lang="en">
<title>USA: U.S. June trade gap narrows sharply as imports drop.</title>
<headline>U.S. June trade gap narrows sharply as imports drop.</headline>
<dateline>WASHINGTON 1996-08-20</dateline>
<text>
<p>The U.S. trade gap narrowed dramatically in June as imports of merchandise and petroleum plunged from May levels, the Commerce Department said Tuesday.</p>
<p>The monthly deficit dropped 23.1 percent to $8.11 billion from a revised $10.55 billion in May -- much lower than the $9.4 billion shortfall that Wall
Street economists had forecast for June.</p>
<p>June exports eased a slight 0.3 percent to $69.71 billion while imports dropped 3.3 percent to $77.82 billion.</p>
</text>
<copyright>(c) Reuters Limited 1996</copyright>
<metadata>
<codes class="bip:countries:1.0">
  <code code="USA">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1996-08-20"/>
  </code>
</codes>
<codes class="bip:topics:1.0">
  <code code="E512">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1996-08-20"/>
  </code>
</codes>
<dc element="dc.date.created" value="1996-08-20"/>
<dc element="dc.publisher" value="Reuters Holdings Plc"/>
<dc element="dc.date.published" value="1996-08-20"/>
<dc element="dc.source" value="Reuters"/>
<dc element="dc.creator.location" value="WASHINGTON"/>
<dc element="dc.creator.location.country.name" value="USA"/>
<dc element="dc.source" value="Reuters"/>
</metadata>
</newsitem>
```

**Figure 3.1:** *Data example*

$$\mathbf{x}^m = [\prime the\prime, \prime us\prime, \prime trade\prime, \prime gap\prime, \prime narrowed\prime, \prime dramatically\prime, \prime in\prime, \prime june\prime, \ldots] \tag{3.2}$$

$$\mathbf{y}^m = [\prime USA\prime, \prime E512\prime] \tag{3.3}$$

In this example, we can see that $\mathbf{y}^m$ contains 2 labels and they are subordinate to two main classes, "USA" is a region label, "E512" belongs to topics labels. The task is to learn models that can predict the true labels which should belong to the news, the output of the classifiers contains label elements, see in equation 3.4. The output is not necessarily two labels like the input labels, it may be one or three, which depends on the actual performance of the classifiers. Finally, the prediction result $\hat{\mathbf{y}}$ would be compared to the real label set $\mathbf{y}$ and calculate the final evaluation result to evaluate the quality of classifiers.

$$\hat{\mathbf{y}} = classifier(\mathbf{x}) \tag{3.4}$$

This task can be divided into three subtasks because there are three main classes in the labels: industry class, region class and topics class, the reason will be illustrated in 3.2.1. They are just the categories to which the actual labels belong instead of the actual labels in the dataset. For each classifiers, three different classifiers are required for these three main

classes, for example, there would be three different Decision Tree classifiers to train and test in the dataset for three different class.

## 3.2 Data exploratory

### 3.2.1 Data description

There are over 800,000 news stories in the Reuters Corpus, they are all stored in XML format. News information mainly includes newsitem (basic information includes news id and date), title, headline, dateline, text and codes.

The news id in newsitem column is the unique identifies of news, there is a one-to-one correspond relationship between id and news, we can index to news by their id. The headline column contains the title of news, the title column contains not only title information but also the publishing country information. The dateline column shows publishing date and the name of writer. The text columns contains the content of news.

For codes, they represent the labels of news, there are over 1300 labels appear in the description files, but only 752 types of codes appear in this dataset, they are divided into 3 classes: industry, region, topics. Industry class contains 350 labels, region class contains 296 labels, and topics class contains 103 labels. In these three classes, the structure of topics are hierarchical, for example, in topics class, 'ECAT' means the economic catagory and all labels with prefix of 'E' are its sublabels. In industry class, there are some dependency relationship between labels. In region class, labels are independent.

There are three main features of the data information: text length, labels for each news and code frequency. For text length, around 40% news contain less than 100 words, most news contain less than 500 words, the length of news are random distributed. For amount of codes for each news, most news contain less than 10 labels, so it is actually a multi-class and multi-output problem, but the distributions of them are complex after they are split into three classes, it will be explained in next paragraph. For code frequency, which means how many times has a label appeared, 80 labels appear less than 20 times in the whole dataset, 250 labels appear less than 200 times, 250 labels appear between 200 and 1600 times, and around 10 labels appear over 50,000 times, it is clear that the data distribution is skew.

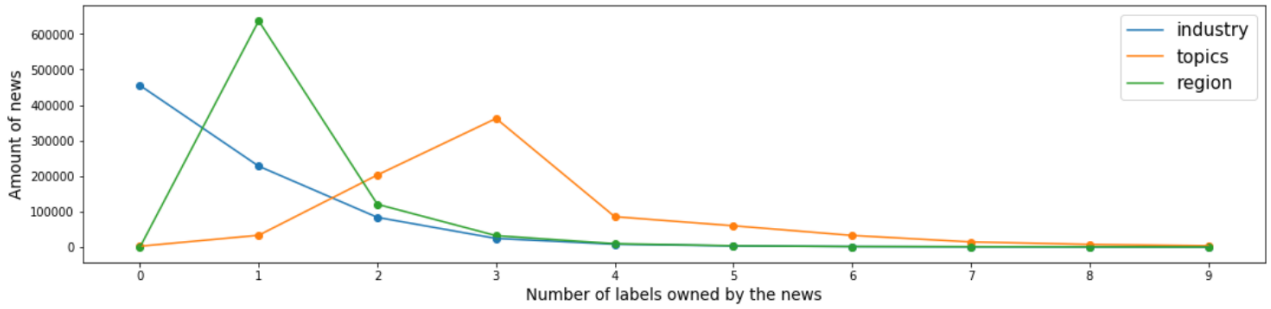
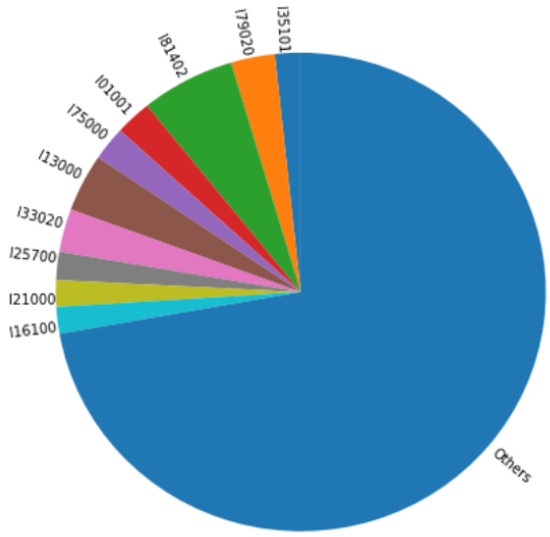
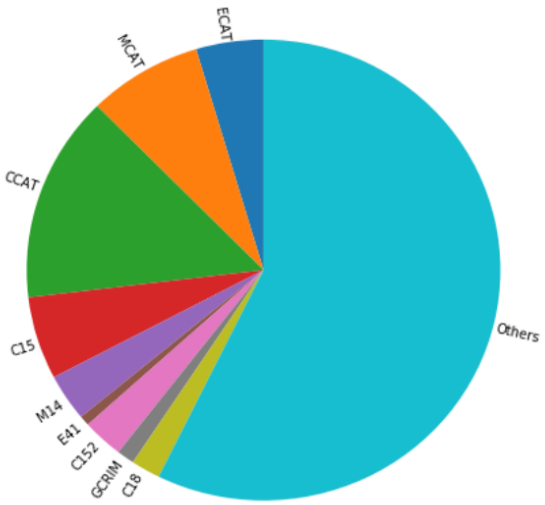



**Figure 3.2:** *The distribution of data classes*

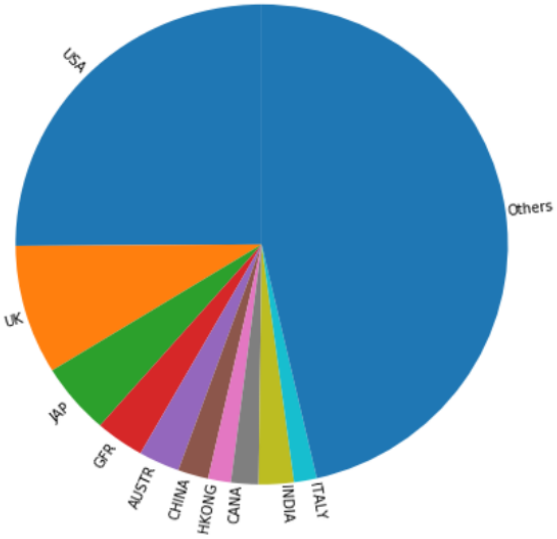
The distribution of data classes is a little unbalanced, the amount of news with industry labels is fewer than the amount of news with other labels, see in figure 3.2. It is clear that over half news do not own industry labels, and most news only own less than 4 industry labels; For region class, most news, over 75%, own only 1 region label; For topics class, the distribution is different from other two classes, there are around half news with 3 topics labels, and most news do not own over 6 topics labels. So, the distributions of data classes are different from each other. That is why the main task of this project would be divided into three subtasks.




(a) label distribution of industry class




(b) label distribution of topics class




(c) label distribution of region class

**Figure 3.3:** *the proportion of most frequency labels and others*

Figure 3.3 shows some pies which are also talking about distribution of labels, these pies

compare the proportion of top 10 most frequency labels and other labels in each class. It is clear that this dataset is imbalanced because the top 10 labels account for a large proportion in the data set, especially in region class, the number of "USA" is almost half of the number of all the other labels and the proportion of top 10 labels in this class is near 50%. For industry class, top 10 labels account for over 25% of all industry labels. For topics class, top 10 labels account for near 40% of all topics labels. It is clear that a small number of labels appear too many times, and it may affect the classification task.

### 3.2.2 Data selection

Due to the large data set and limited hardware configuration, there are only 100,000 news selected as training set, 20,000 news selected as validation set and 50,000 news selected as testing set, these 170,000 news are extracted by hand because of the data imbalanced distribution, the amount of news with industrial labels is less than the amount of news with other types of labels, and news with industrial labels often contains other types of labels, so the training and testing dataset are selected based on whether news contain industrial labels, in order to keep the balance of classes.

The datasets that are used in this project are taken from Reuters Corpus. There are no ethical nor legal issues since the datasets are publicly available. All sources are referenced correctly, including books and journal articles. Software used such as Python, scikit learn and Pytorch are free for use.

## 3.3 Models design

This part is mainly focus on models designation. Firstly, selecting several models as candidate models in this task, and some specific tools will be introduced. The second part is talking about traditional classifier designation, including the architecture of models and preprocessing. The third part is talking about deep learning classifier designation, including the network structures of 4 different classifiers. And the final part is focusing on evaluation metric which will be used as final evaluation tools.

### 3.3.1 Model and tool selection

The aim of this article is to compare the performance between traditional machine learning algorithms and deep learning methods, so I need to choose several models as candidates for comparing. By investigating different methods in multi label classification field, I decide to use KNN, SVM, Logistic Regression and Decision Tree as traditional machine learning group, and fasttext, CNN, RNN and RCNN as deep learning group.

Python is widely used in the field of NLP because it has a mature package repository for machine learning and neural network. In addition, a large number of open source frameworks work for it. The project environment is chosen as Windows-64 bit operating System with Python.

To implement the classification approaches, the following public Python packages have been used:

- Sci-Kit Learn: For machine learning algorithms, including Decision Tree, SVM, KNN and logistic regression.
- Natural Language Toolkit (NLTK): For natural language processing tasks such as tokenisation and stemming.
- MatPlotLib: For data visualisation.
- NumPy: For general mathematical operations.
- Pandas: For presentation of datasets and results in dataframe layout.
- csv: This module implements classes to read and write data in CSV format.
- PyTorch: A rich ecosystem of tools and libraries which supports development in computer vision, NLP and more.
- itertools: The module standardizes a core set of tools that are useful by themselves or in combination.
- Glove(Stanford): It is an unsupervised learning algorithm for obtaining vector representations for words.
- re: This module provides regular expression matching operations.
- bs4(BeautifulSoup): It is a Python library for pulling data out of HTML and XML files.
- random: This module implements pseudo-random number generators for various distributions.

### 3.3.2 Traditional classifier design

This part is talking about traditional methods for text classification, the architecture of all classifier models and the features for these classifiers will also be introduced.

#### Traditional machine learning architecture

The implementation and evaluation of traditional classifiers will be divided into six steps, which is described in figure 3.4.

For the whole dataset, in the first stage, preprocessing news and then extract features such as TFIDF(term frequency inverse document frequency) as the preparation for training classifiers. In the second stage, training classifiers such as decision tree by the extracted features. In the third stage, predict the labels of test data and evaluate the performance of trained classifier, such as micro F1 score.

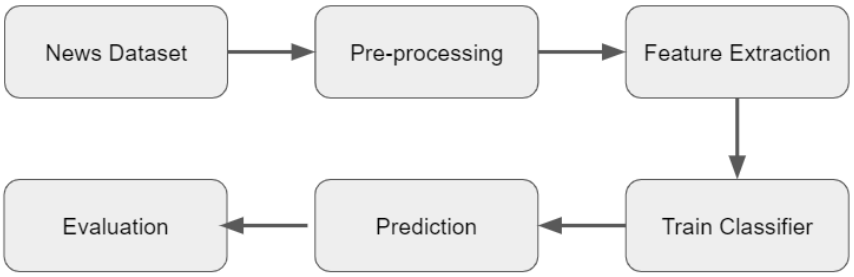



**Figure 3.4:** *The text classification process*

#### Traditional machine learning preprocessing

Firstly, load news text and extract words by regular expression in order to remove some meaningless symbols such as punctuation and numbers, this step is also called tokenisation. Then stemming and remove stop words, stemming is an important way to normalize the form of the word, and it can achieve the purpose of effectively merging the form of the word, stopwords are words which are used too frequently in corpus, such as "is", including adverbs, prepositions and conjunctions, removing them can help to remove noise and effectively help to improve keyword density.

Secondly, transfer remaining words to TFIDF vectors. TFIDF is used to assess the importance of a word for a file set or one of the files in a corpus. The importance of a word increases proportionally with the number of times it appears in the file, but it also decreases inversely with the frequency it appears in the corpus. So it can be used to filter out common words and retain important words, because the more times a word appears in an article and the fewer times it appears in all documents, the more it can represent the article.

Finally, processing labels. In practical machine learning tasks, features are sometimes not always continuous values, and may be some categorical values. Especially in the dataset of this project, there are totally 752 different labels of news. For these labels, we usually need to convert them into a form that is easy to exploit by machine learning algorithms. A good method is one hot encoding, it uses a bit status register to encode a state, each state is independent of its register bits, and only one bit is active at any time. So for labels, I choose to use one hot representation. One result of this is that the labels representation becomes very sparse, but it can help machine to understand labels.

### 3.3.3 Deep learning design

This part is talking about deep learning methods for text classification, the architecture of all classifiers will also be introduced.

### Fasttext

The architecture of fasttext neural network is described in figure 3.5. The output layer and hidden layer are listed separately because of their complex structures.

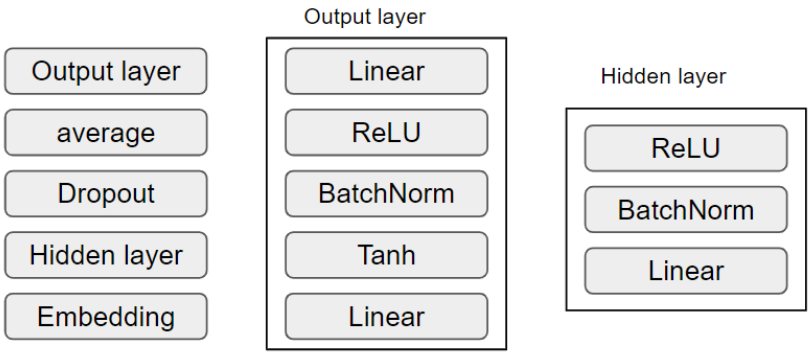



**Figure 3.5:** *fasttext architecture*

Features are embedded from the output of first hidden layer, the hidden layer is consisted of linear layer, BatchNorm layer and ReLU layer, the BatchNorm layer in here is a good method to avoid overfitting, add a dropout layer between hidden layer and average layer is also trying to avoid overfitting. The averaging layer is actually simple summation average calculation, in here, the word sequence will not be used, because according to the keyword, machine can judge whether the news belong a certain field, word sequence information is not so helpful for long text classification. Finally, there is a output layer which is consisted of linear layers, Tanh layer, BatchNorm layer and ReLU layer.

### CNN architecture

The architecture of CNN neural network is described in figure 3.6. The convolution layers are listed separately because they are used many times in the neural network.

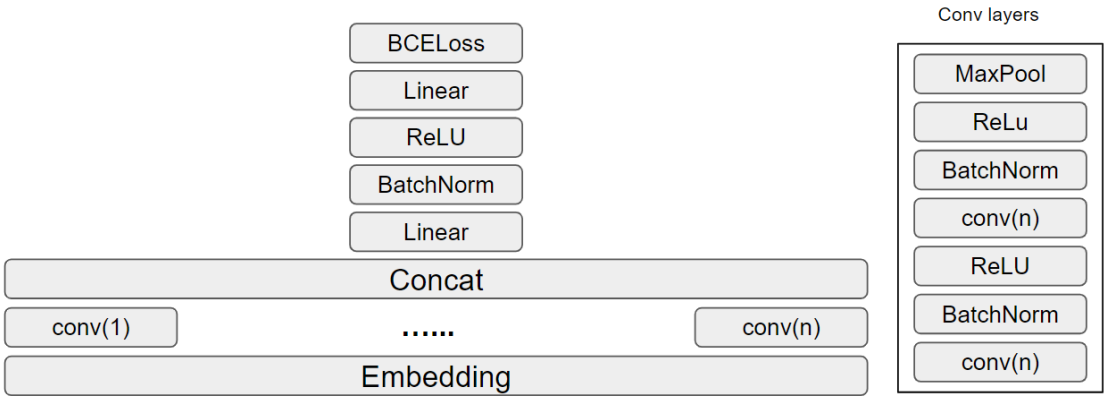



**Figure 3.6:** *CNN architecture*

After embedding, the input word index will be convoluted by 1-d convolution layers with different kernel size to extract local feature, I use 2 convolutional layers as a convolution unit, in the unit, there are always a BatchNorm layer and a ReLU layer after convolution, and finally, maxpooling the output of the second convolutional layer and concatenate all outputs of different convolution units. The number $n$ in figure 3.6 means the convolution layer with a specific kernel size, which can be increased or decreased at any time in training.

After concatenating, fully connected network is used for prediction, there are also batch normalization and ReLU activation between two linear layers. Finally, use loss function to evaluate the difference between the output and real labels.

### RNN architecture

The architecture of RNN neural network is described in figure 3.7. The output layer is listed separately because of its complex structures.

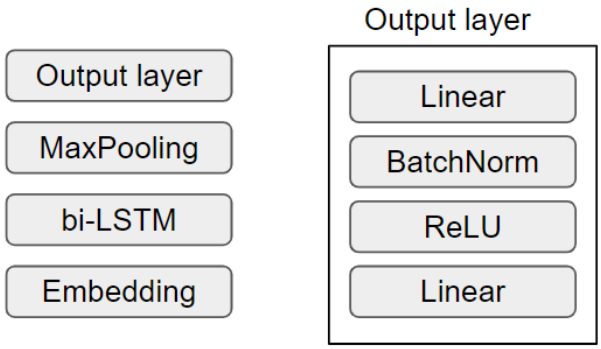



**Figure 3.7:** *RNN architecture*

After embedding, the input word index will be sent to a bidirectional LSTM layer in order to store more context information. After that, a max pooling layer is used for reducing the spatial size of the representation by reducing the amount of parameters and computation in the network. In the next layer, 2 fully connected linear layers are used for prediction, and a BatchNorm layer and a ReLU layer are set between them.

### RCNN architecture

In this paper, the RCNN model is actually the combination of RNN and CNN, the architecture of RCNN is described in figure 3.8, the output layer and convolution layer are listed separately.

In the first stage, the process is similar to RNN, embedding and then send the output to bidirectional LSTM, but then I will concatenate the output of embedding and output of LSTM. In the second stage, the process is similar to CNN, set 2 convolutional layers as a convolution unit, and use several convolution units with different kernel size as the convolutional layer.

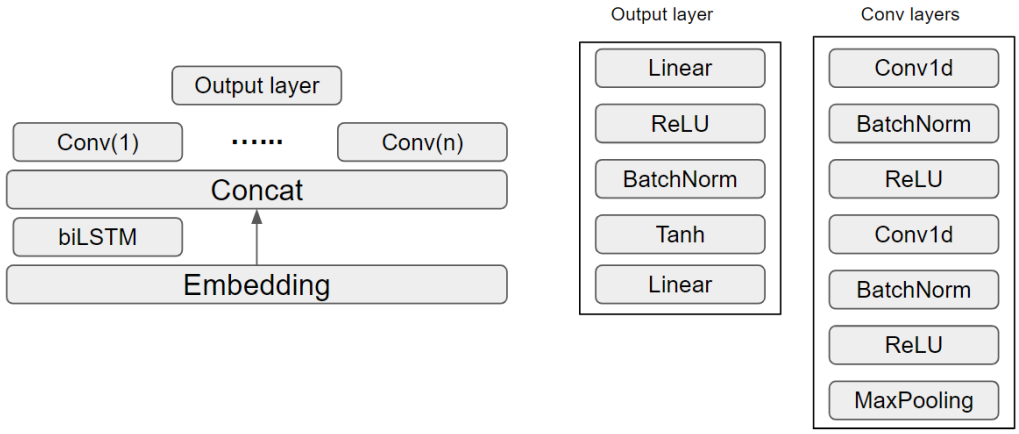



**Figure 3.8:** *RCNN architecture*

In the final stage, predict the matching labels by using fully connected linear layers which is consisted with 2 linear layers, 2 activation layers and one BatchNorm layer.

#### Features for deep learning methods

The features preprocessing for deep learning is a little different from for traditional machine learning. The methods for preprocessing labels are similar, I will still use one hot representationfor labels. But for word representation and document representation, I will use a new method, word embedding. It refers to embed a high-dimensional space whose dimension is the number of all words into a continuous vector space with a much lower dimension, each word or phrase being mapped to a vector on the real number field. By using it ,we can convert a word into a fixed-length vector representation for mathematical processing.

### 3.3.4 Evaluation design

F1 score will be used as the main evaluation standard, and the precision score and recall score will also be considered as importance evaluation figures, because they can provide detailed performance of classifiers. The evaluation metric which has been used in this project see in figure 4.4.

Where True Positive means if the prediction of a the specific label is true, False Negative means the labels of prediction does not contain this labels, which is actually the real labels, False Positive means the prediction is wrong, True negative means both true labels and predicted labels do not contain this labels.

Precision means ratio between the amount of data which is predicted as the correct positive and the amount of data which is predicted as positive. Recall means ratio between the amount of data which is predicted as the positive and the amount of data which is actual

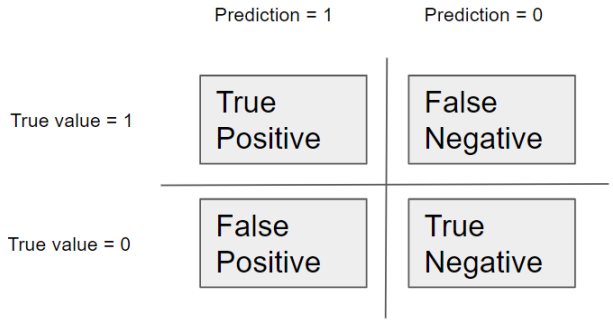



**Figure 3.9:** *confusion table for this project*

positive. There value can be calculated by the equations which have been discussed in 2.2.5. The F1 score can be calculated according to the value of precision and recall by the equation which has also been showned in 2.2.5.

In the final evaluation step, both micro averaging and macro averaging will be considered when evaluate a classifier. The macro average is the method to calculate the evaluation figures for each class firstly, and then to calculate the arithmetic mean for all classes. The micro-average is to establish a global confusion matrix for each instance of the data set without category, and then calculate the total evaluation figures.

## 3.4 Model Implementation

The above part has discussed the designation of models, this part will focus on how to implement these classifiers. The traditional machine learning part mainly focus on data preprocessing and parameters of different models, the deep learning part mainly focus on data preprocessing, activation functions and hyperparameters.

### 3.4.1 Traditional machine learning implementation

For traditional models, there are two important implementations in building classifiers, the first one is data preprocessing, the second one is models implementation. In the second stage, the best parameters for traditional classifiers from sklearn library can be calculated by comparison the performance of models after adjusting parameters.

#### Data Preprocessing

For the plain data, the extracted information including id, title, news text and codes, then extract words from context by regular expression tokenizer, after tokenisation, stemming words by using NLTK library. After these works, the clean text is , each news contains different amount of words in the vector type, see in equation 3.5:

$$\mathbf{x}^m = [w_1, w_2, \ldots, w_i, \ldots] \tag{3.5}$$

| | parameter | industry | topics | region |
|---|---|---|---|---|
| **KNN** | algorithm | balltree | balltree | balltree |
| | leafsize | 20 | 20 | 20 |
| | nneighbors | 5 | 3 | 3 |
| | weights | uniform | uniform | distance |
| | p | 2 | 2 | 2 |
| **Decision Tree** | criterion | gini | gini | gini |
| | max depth | 20 | 20 | 20 |
| | min sample leaf | 2 | 1 | 2 |
| **Logistic Regression** | solver | lbfgs | lbfgs | lbfgs |
| | multi class | multinomial | multinomial | multinomial |
| **SVM** | class weight | balanced | balanced | balanced |

**Table 3.1:** *best parameters for traditional models*

Where $w_i$ represent a word and this vector contains strings. In the next stage of preprocessing, transfer the value of this vector into numbers by using bag of words, the TfidfVectorizer from sklearn library is used in this step. After transferring words vector to their related TFIDF vector, the news representation become a $N$-d vector, where $N$ is the amount of unique words in the whole dataset, for example, $tfidf_i$ represent the tfidf value of $i$th feature for news $m$ , see in equation3.6.

$$\mathbf{x}^m = [tfidf_1, tfidf_2, \ldots, tfidf_i, \ldots, tfidf_N] \tag{3.6}$$

Labels are transferred into one hot representation, because many labels are mutually exclusive, with only one activation at a time. For example, there are 3 labels in the $m$th news, if the index of this news are $i, j, k$ in the 752 labels, where $i = 1, j = 3, k = 5$, the representation of the label is shown in equation 3.7.

$$\mathbf{y}^m = [0, 1, 0, 1, 0, 1, \ldots] \tag{3.7}$$

### Models implementation

There are some supervised machine learning classifiers which can be used for text classification tasks. In this project, four classifiers (KNN, SVM, Logistic Regression, Decision Tree) are implemented with the aim of comparing their performance in multi-label and multi-output task.

These four classifiers are imported from Scikit Learn Library and have been tested in this project, including KNN classifier (KNeighborsClassifier),SVM classifier (SVC, SVM with a non- linear kernel), Logistic Regression classifier (LogisticRegression) and Decision Tree classifier (DecisionTreeClassifier). Training these classifiers with the training data and find out the best parameters by using GridsearchCV function from Scikit Learn, then build models with these best parameters and test with the testing data.

There are the best parameters for these four classifiers see in table 3.1 the parameters which are not mentioned in below are setting by default.

### 3.4.2 Deep learning implementation

In deep learning part, preprocessing of text data and implement the algorithm code is the process of first stage, the subsequent work includes shuffle data and hyperparameter tuning, including learning rate, number of iterations, hidden layer dimension and batch size.

#### Data Preprocessing

In deep learning approaches, removing stopwords, tokenising words, and stemming them are still helpful, but for word representation, word embedding will be used instead of TFIDF. For labels preprocessing, the practice here is the same as in traditional methods, one hot representation will be used to represent labels.

Glove from Stanford University will be used for word embedding, the pretrained embedding file which is use here is "glove.840B.300d.zip". In order to ensure that the consistent format of input for embedding layer, the length of the news content is limited to 120 words. If the original length exceeds 120 words, intercept a fragment with 120 words. If the original length is less than 120 words, then added to 0 until the length of 120 words is reached.

#### Activation functions

When there are only linear layers, no matter how they are combined, the output is a linear transformation of the input, and they are not helpful for nonlinear classification, so the activation functions are introduced into the neural networks in order to add nonlinear factors. Some non linear activation functions are considered to be used in the output layer of models, including ReLU, Sigmoid and ELU, see in equation 3.8, 3.9 and 3.10.

$$ReLU(x) = \max(0, x) \tag{3.8}$$

$$Sigmoid(x) = \frac{1}{1 + \exp(-x)} \tag{3.9}$$

$$ELU(x) = \max(0, x) + \min(0, (\exp(x) - 1)) \tag{3.10}$$

Where $x$ is the input of these activation functions. These functions are used to add nonlinear factors and solve the problems that the linear functions cannot solve.

Different activation functions have their own advantages and disadvantages. When using sigmoid and other functions, the calculation of activation function belongs to the exponential operation, and the derivation of back propagation involves division, this requires huge computing resources. For deep networks, when the sigmoid function is backpropagated, gradients will disappear, because the transformation is too slow and the derivative tends to 0 when the sigmoid function is close to the saturation region, which will cause information loss, therefore, it is hard to complete the training of the deep network.

ReLU will set the output of some neurons as 0, which will keep the sparseness of the network, and reduce the interdependence of parameters, in this case, ReLU can help to avoid overfitting

problem. ELU is an improvement on ReLU, it can take a negative value, which makes the unit activation mean closer to 0, similar to the effect of Batch Normalization but it requires lower computational complexity.

In this project, ReLU function is used in most of deep learning methods because of its ability to avoid overfitting.

### Hyperparameters

For learning rate, in order to prevent overfitting, the learning rate is setting as decay mode. Firstly initial the learning rate as 0.01, then in the iteration step, change its value for each epoch by following the equation 3.11.

$$lr = \frac{lr}{1 + 0.01 * epoch} \tag{3.11}$$

In this way, after 10 epoch, the learning rate will decrease to 0.0058, after 20 epoch, the learning rate will decrease to 0.0014, it is helpful for fitting into global minimum.

For epoch, try to test the converge speed of models in the same learning rate, set up multiple experiments in different epoch, including 10 and 20.

The choice of the batch size determines the direction of the gradient descent. There are three cases for setting the batch size, online learning, full batch learning and mini batch learning. If the batch size is equal to 1, it is also called online learning, the direction of each correction is corrected by the gradient direction of the respective samples, it decreases in every direction, which makes it difficult to converge. For full batch learning, the batch size is equal to the sample size, although the direction determined by the full sample set is more representative of the population, the correction of each gradient cancels out and the gradient cannot be degraded because of the sampling difference between all the samples. Instead of these two extreme cases, mini batch learning can be used, if the data set is sufficient, the gradient trained with half or even much smaller sample data is almost same to the gradient trained with full data samples.

It is important to decide the number of batch size, increasing the batch size within a reasonable range can effectively improve the memory utilization rate, it will make the direction of the gradient drop more accurate, and reduce the training shock. In this project, in order to make sure the training quality and training speed, the batch size will be set as 250 or 500.

A single-layer neural network can only be used to represent linearly separable functions. However, the problem in this project needs to solve non linearly separable, a multi-layer neural network can be used to represent convex regions, in fact, they can learn to plot shapes around instances in some high-dimensional spaces to classify them to overcome the limitations of linear separability.

The choice of the number of hidden layer hidden layer nodes is also very important. It has a great impact on the performance of the established neural network model, and also is the direct incentive of overfitting during training. For the number of hidden layers, as the model

changes, the number of hidden layers will also changes, we need tot discover the most effective method for the particular data set by using experiments.

## 3.5 Summary

This chapter presents the practical problems of the project and analyzes the specific news dataset of Reuters Corpus, it explain the actual data which would be used in the training ,validating and testing. Based on the problems raised before, the designation of both traditional machine learning methods and deep learning methods are proposed. Finally, this chapter illustrates how to implement these classifiers.

The next chapter will show the result of prediction in three main classes, and then it is going to compare and discuss the different performance of classifiers which are mentioned and designed in this chapter, especially the difference between 4 traditional classifiers and 4 deep learning classifiers.

# Chapter 4

# Results and Discussion

This chapter is divided into 5 parts, the first three parts are talking about the evaluation result of classifiers for the prediction on three main class, it compares the macro performance and micro performance of all classifiers. The fourth part analyzes the training loss of 4 different deep learning classifiers. The final part is the error analysis, it focuses on error visualization by using confusion matrix which is built by the top 10 frequent labels from three different class.

## 4.1 Industry class

There is the summary of results for industry class, see in table 4.1. The columns represent different evaluation figures, including macro F1 score, macro precision score, macro recall score, binary F1 score, binary precision score and binary recall score. The rows represent different algorithms, including KNN, Decision Tree, SVM, Logistic Regression, RNN, CNN, RCNN and Fasttext.

| | Macro | | | Micro | | |
|---|---|---|---|---|---|---|
| industry class | F1 | Precision | Recall | F1 | Precision | Recall |
| KNN | 0.465 | 0.823 | 0.369 | 0.632 | 0.803 | 0.521 |
| Decision Tree | 0.424 | 0.573 | 0.238 | 0.386 | 0.767 | 0.258 |
| SVM | 0.629 | 0.754 | 0.576 | 0.743 | 0.741 | 0.744 |
| Logistic Regression | 0.316 | 0.899 | 0.230 | 0.599 | 0.888 | 0.452 |
| RNN | 0.389 | 0.780 | 0.296 | 0.628 | 0.798 | 0.517 |
| CNN | 0.686 | 0.850 | 0.613 | 0.719 | 0.802 | 0.652 |
| RCNN | 0.539 | 0.745 | 0.445 | 0.726 | 0.802 | 0.663 |
| fasttext | 0.501 | 0.690 | 0.396 | 0.666 | 0.766 | 0.589 |

**Table 4.1:** *result for industry class*

The performance under the micro averaging evaluation is better than the performance under the macro evaluation, because the macro evaluation considers the average performance of

different labels, while the micro evaluation considers the overall performance of all labels, the dataset which is used in this project is actually an unbalanced dataset, this fact results in poor classification performance on labels which are rare in whole dataset. In this case, micro evaluation is more susceptible to the labels which appear frequently and it is easier for machines to learn how to classify these labels, so the result of micro evaluation is better than the result of macro evaluation.

For macro F1 score, half of classifiers can get F1 score higher than 0.5, the performance of CNN is the best, the F1 score of CNN is around 0.686, the SVM classifier also performs well and the related F1 score is 0.689. In contrast to these two classifiers, Logistic Regression can only get F1 score less than 0.35.

For macro precision score, most classifiers except Decision Tree and Fasttext can achieve the precision over 0.70. For macro recall, most classifiers except SVM and CNN can only get recall score less than 0.5. According to these two figures, SVM and CNN is better than others.

For micro F1 score, the SVM, CNN and RCNN can get F1 score over 0.7 and the Decision Tree can only get F1 score less than 0.4. It is clear that the performance of SVM, CNN and RCNN for industry labels is good, by comparing the difference between macro F1 score and micro F1 score of these three models, we can found that the growth from macro F1 to micro F1 of RNN is higher than SVM and RCNN, so RNN is more likely to handle labels which appear frequently, instead of handling imbalanced data.

For micro precision, all classifiers can get precision over 0.7, especially KNN, CNN and RCNN, the precision scores of these three classifiers are higher than 0.8. For micro recall, the recall score of SVM is the highest one, which is 0.744, and the recall scores of CNN and RCNN are also better than the rest classifiers, which are 0.652 and 0.663 respectively, the recall score of Decision Tree is the lowest one among all classifiers, which is only 0.258.

## 4.2 Topics class

There is the summary of results for topics class, see in table 4.2. The table format is similar to table 4.1.

| | | Macro | | | Micro | |
|---|---|---|---|---|---|---|
| topcis class | F1 | Precision | Recall | F1 | Precision | Recall |
| KNN | 0.494 | 0.578 | 0.372 | 0.716 | 0.779 | 0.662 |
| Decision Tree | 0.424 | 0.573 | 0.238 | 0.717 | 0.875 | 0.607 |
| SVM | 0.582 | 0.611 | 0.604 | 0.820 | 0.780 | 0.864 |
| Logistic Regression | 0.464 | 0.862 | 0.383 | 0.831 | 0.921 | 0.756 |
| RNN | 0.513 | 0.695 | 0.419 | 0.828 | 0.875 | 0.785 |
| CNN | 0.542 | 0.606 | 0.443 | 0.823 | 0.842 | 0.805 |
| RCNN | 0.533 | 0.649 | 0.457 | 0.829 | 0.850 | 0.809 |
| fasttext | 0.541 | 0.582 | 0.409 | 0.791 | 0.817 | 0.767 |

**Table 4.2:** *result for topics class*

For macro F1 score, most classifiers can get F1 score higher than 0.5, the performance of SVM is the best, the F1 score of it is 0.582, RNN, CNN and RCNN also perform well and the F1 scores of them are 0.513, 0.542 and 0.533 respectively. In contrast to these four classifiers, Decision Tree can only get F1 score less than 0.45, which is similar to its performance of industry labels.

For macro precision, most classifiers except KNN, Decision Tree and Fasttext can get the precision over 0.60. For macro recall, most classifiers except SVM can only gain recall score less than 0.5. According to macro averaging evaluation, SVM is the best model to handle imbalanced labels of data.

For micro F1 score, the F1 scores of most classifier are over 0.7, half of them can even achieve F1 score over 0.8. So the performance under the micro evaluation is also better than the performance under the macro evaluation, the reason of this result has been discussed above in part 4.1.

For micro precision, all classifiers can get precision score over 0.75, especially Logistic Regression, the precision score of it is higher than 0.9. For micro recall, the recall score of SVM is the highest one, which is 0.864, and the recall scores of CNN and RCNN are also better than the rest classifiers, which are 0.805 and 0.809 respectively, the recall score of Decision Tree is the lowest one among all classifiers, which is only 0.607.

For topics class, the performances of traditional machine learning classifiers are different, SVM and Logistic Regression are better than KNN and Decision Tree, but the performances of deep learning classifiers are similar, all of them performs as well as SVM and Logistic Regression.

## 4.3 Region class

There is the summary of results for region class, see in table 4.3. The table format is similar to table 4.1.

| | Macro | | | Micro | | |
|---|---|---|---|---|---|---|
| region class | F1 | Precision | Recall | F1 | Precision | Recall |
| KNN | 0.692 | 0.870 | 0.635 | 0.821 | 0.870 | 0.776 |
| Decision Tree | 0.448 | 0.876 | 0.378 | 0.824 | 0.946 | 0.730 |
| SVM | 0.764 | 0.883 | 0.724 | 0.916 | 0.922 | 0.910 |
| Logistic Regression | 0.547 | 0.989 | 0.499 | 0.889 | 0.973 | 0.818 |
| RNN | 0.657 | 0.895 | 0.599 | 0.898 | 0.932 | 0.866 |
| CNN | 0.686 | 0.850 | 0.613 | 0.892 | 0.912 | 0.872 |
| RCNN | 0.688 | 0.860 | 0.639 | 0.899 | 0.920 | 0.878 |
| fasttext | 0.680 | 0.863 | 0.600 | 0.878 | 0.916 | 0.842 |

**Table 4.3:** *result for region class*

For macro F1 score, the performance of SVM is the best, the related F1 score is 0.764, in contrast to SVM, Decision Tree and Logistic Regression can only get F1 score less than 0.55,

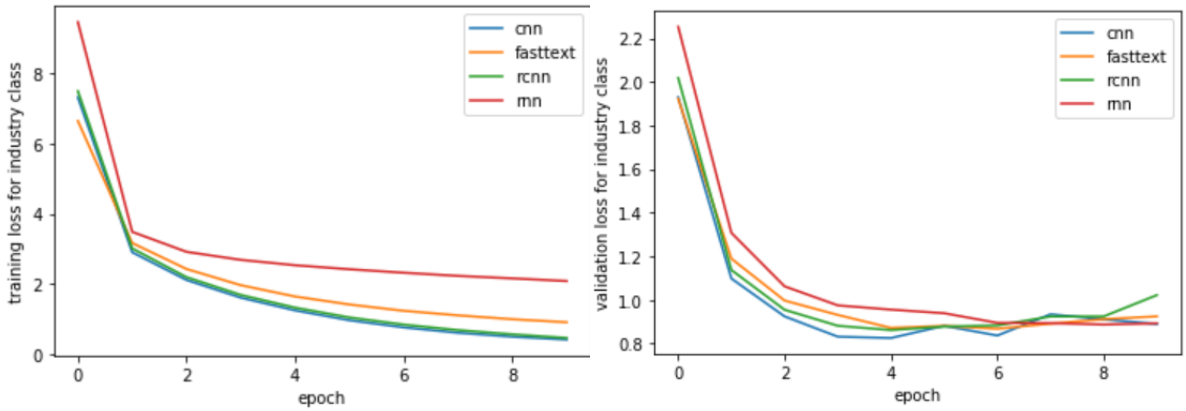
the F1 scores of other deep learning classifiers are between 0.6 and 0.7. For macro precision, most classifiers can get precision score over 0.8. For macro recall, most classifiers except Decision Tree and Logistic Regression can get recall score higher than 0.6.

For micro F1 score, the F1 scores of most classifiers are over 0.8, and SVM can even achieve F1 score over 0.9. There is no significant difference in their performance. For micro precision, most classifiers except KNN can get precision over 0.9, especially Logistic Regression, the precision score of it is 0.973. For micro recall, the recall score of SVM is the highest one, which is 0.910, and the recall scores of all deep learning classifiers are also better than the rest classifiers.

For these three classes, it is clear that the performance for industry class is the worst one, the performance for region class is the highest one, the F1 score difference between them is around 0.15. The reason is that there are 350 labels in industry class and they are difficult to be distinguish, it is also hard for classifiers to identify topics labels, but there are only around 100 labels in topics class. In contrast to these two classes, the region information is obvious and it is easy for classifiers to identify it, if a region is mentioned in news, this region will appear in the labels. The difficulty of predicting is increasing with the increase of complexity of label categories.

## 4.4 Training loss

There is the training loss of different deep learning classifiers, which are divided into three main classes. In the experiments, the batch size is set as 250, the epoch number is set as 10, and learning rate initialize as 0.01, then decrease with the increase of epoch with the equation $lr = lr/(1 + 0.005 * lr)$.

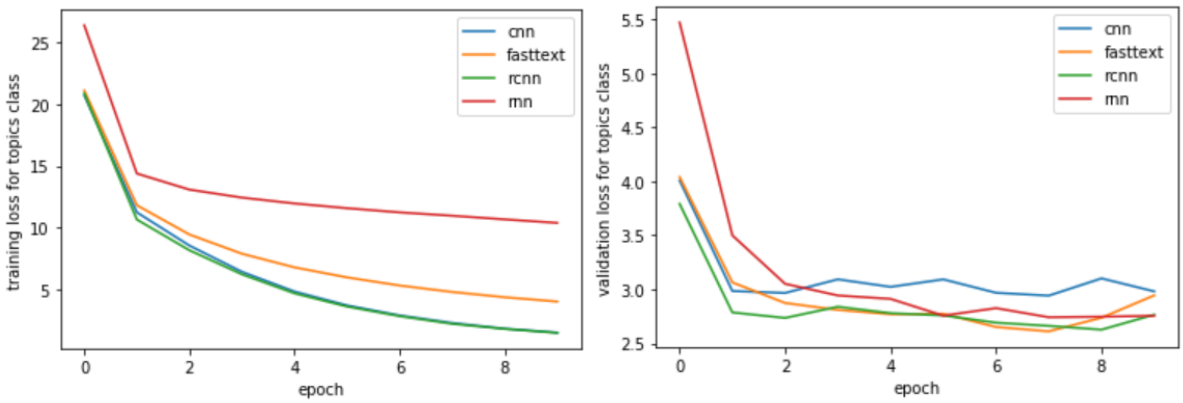



(a) loss of training set (b) loss of validation set

**Figure 4.1:** *training loss for industry class*

In industry class, see in figure 4.1, for training set, all models converge quickly in 5 epoch, but then they converge slowly, especially RNN, it seems like that RNN can not totally converge in only 10 epoch. But when it comes to validation set, the loss is not always falling and the curve

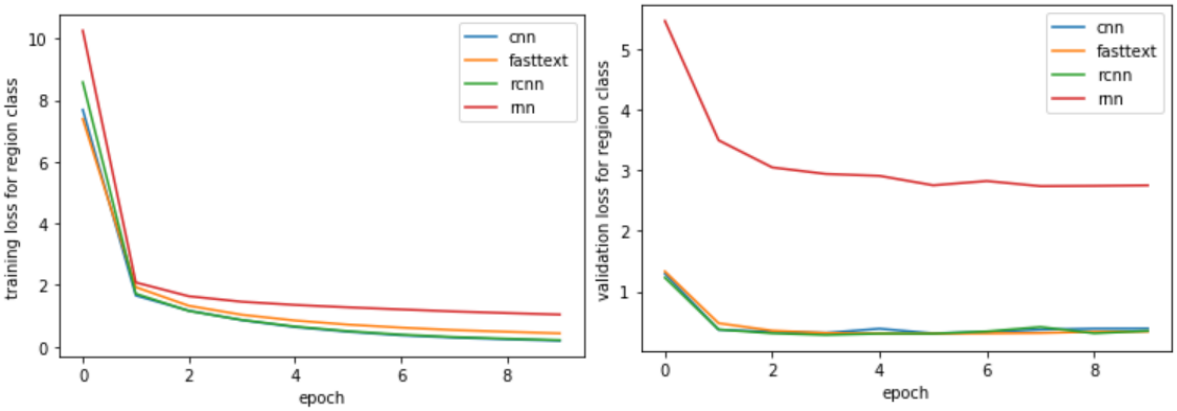
is not smoothing, after 5 iterations, the loss value begins to fluctuate. In the experiment, learning rate is not large and the batch size is enough, so the reason of fluctuation is likely to be uneven sample distribution.




(a) loss of training set (b) loss of validation set

**Figure 4.2:** *training loss for topics class*

In topics class, see in figure 4.2, for training set, CNN and RCNN converge quickly, but fasttext and RNN can not converge in only 20 epoch, especially RNN. It is clear that the converge speed of RNN is lowest in these four networks, and the converge speed of CNN and RCNN is always highest in these four networks. For validation set, the loss curve of all classifiers are also not smoothing. In addition, the training loss of CNN is the lowest one but the validation loss of it is the highest one among all deep learning classifiers, so it is a little overfitting. The training loss and validation loss of RNN are always decreasing, so it is underfitting and need more training iterations.




(a) loss of training set (b) loss of validation set

**Figure 4.3:** *training loss for region class*

In region class, see in figure 4.3, for training set all models converge with a fast speed at first 3 epoch and then converge slowly to the optimum point. For validation set, the trend of loss

curve of all classifiers are similar, but the validation loss of RNN is much higher than others, the fact is verified on the F1 evaluation value of classifiers, the macro F1 score of RNN is also the worst one among all deep learning classifiers. The reason is that the RNN mainly solves the problem of text sequence, and the news classification focuses on keywords recognition.

Overall, the training cost of Fasttext classifier is the lowest one because it only considers linear operations and it is easy for GPU to accelerate it, so it is the best choice as a deep learning benchmark model. If a deep learning model does not overperform Fasttext, it means the neural network of this deep learning model is not well designed.

## 4.5 Error analysis

If the performance of algorithms is not as good as humans, the next step can be guided by manual checking of the errors in algorithms. This process is called error analysis, and the correct direction of optimization system is selected by analyzing the error factors.

Confusion matrix is really helpful for error analysis, it is clear which issues need to be optimized based on visualization. But for multi label classification, especially when the amount of labels is over 700, it is impossible for confusion matrix to plot all labels in one single figure, and it is also hard to match the number of predicted labels and the number of actual labels for each news. In here, the confusion matrix of all labels is represented by selecting the most frequent 10 labels in dataset, and then analyze the specific error by comparing the relationship of these 10 labels.

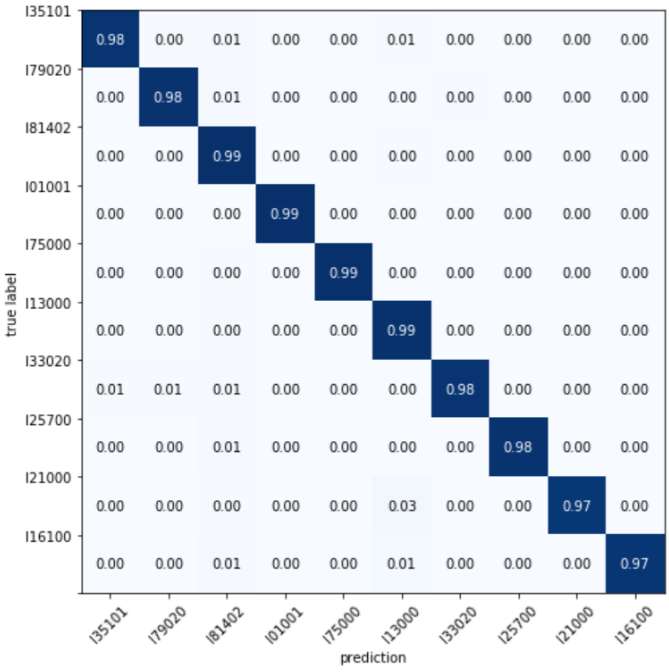



**Figure 4.4:** *The confusion matrix of top 10 frequent labels in industry class*

Take Fasttext as the example to figure out different confusion matrices, because the training of Fasttext is fast and its performance can usually be set as the baseline of all deep learning experiments. For each class, extract top 10 frequent labels which can represent the whole class as the research object. There are 20,000 news are used to plot confusion matrices, these news only consider the top 10 labels and the figures did not consider the news without giving predictions.

All formats of subplots of figure 4.4, 4.5 and 4.6 are the same, each label on the x-axis of the confusion matrices represents the prediction of classifiers for the testing dataset, and each label on the y-axis of the confusion matrices represents the real labels of the dataset. For example, the number of lower right point of 4.4 is 1306, which means there are 1306 "I16100" labels are predicted as "I16100", the number on the left side of it is 2, which means there are 2 "I16100" labels are predicted as "I21000".

Figure 4.4 is the confusion matrix of top 10 frequent labels in industry class, it is clear that most labels are predicted correctly. It is clear that the deeper the color corresponding to the point, the more other labels will be misclassified into this label. For example, there are 4189 "I81402" labels are correctly predicted, and the corresponding number of labels which are wrongly predicted as this label is 97; there are 2725 "I13000" labels are correctly predicted and the corresponding number of labels which are wrongly predicted as this label is 90.

The fact which is found above means the more samples that contain this specific label, the classifier is more likely to predict other irrelevant news as this label because of its higher weight value.

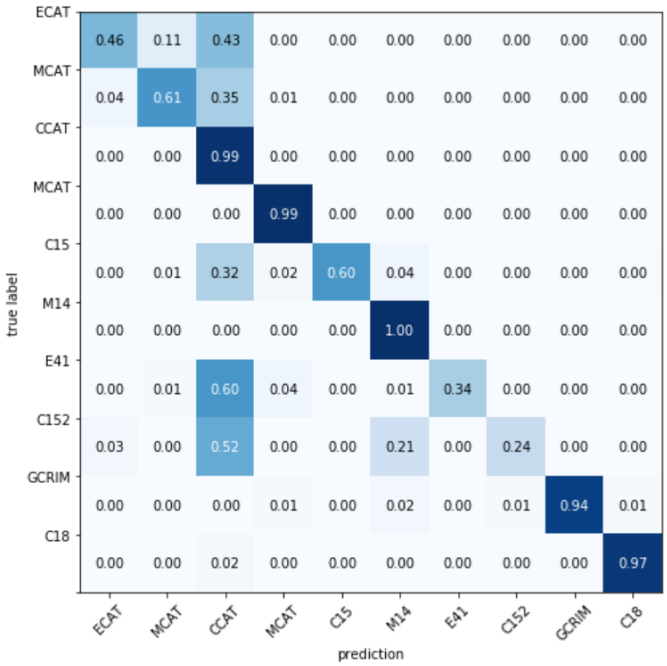



**Figure 4.5:** *The confusion matrix of top 10 frequent labels in topics class*

Figure 4.5 is the confusion matrix of top 10 frequent labels in topics class, there are 4 category

labels, "ECAT", "MACT", "CCAT", "MACT", which have been explained in 3.2.1. It is clear that many other labels are predicted as "CCAT", and the real "CCAT" labels are rarely predicted as other labels, almost 43% of "ECAT" labels and 34% of "MCAT" labels are wrongly predicted as "CCAT". The reason is that the frequency of this labels is much higher than others and the classifiers are more likely to identify news as this label. Conversely, the "MCAT" labels are hardly to be predicted as wrong labels, which means the characteristics are very obvious in this field. So it is a little hard for classifiers to identify the specific category of labels.

In addition, only a very small part of category labels are predicted as other tags, for example, only 18 "MCAT" labels are predicted as "M14" and 15 "CCAT" labels are wrongly predicted as "GCRIM". The reason is that the classifiers have been able to accurately identify the category labels based on training on large number of samples with category labels.

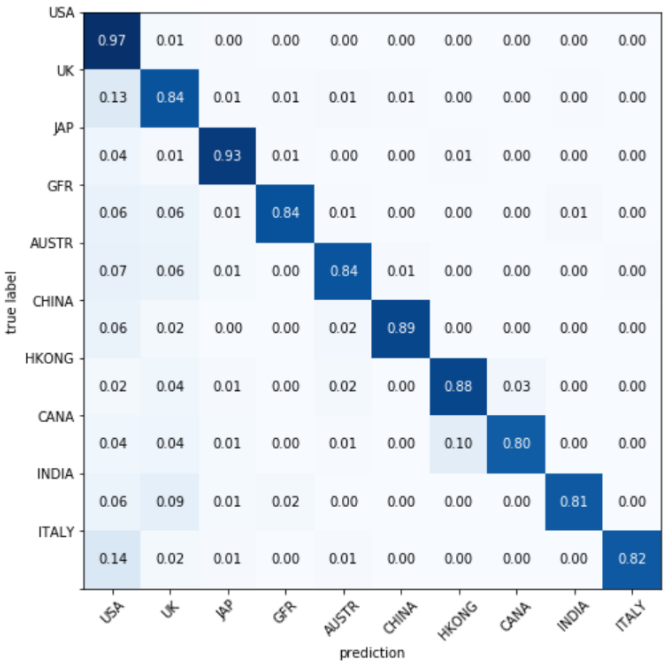



**Figure 4.6:** *The confusion matrix of top 10 frequent labels in region class*

Figure 4.6 is the confusion matrix of top 10 frequent labels in region class, it is clear that most of labels are also predicted correctly. The intersting thing is that 427 "UK" labels are predicted as "USA" and 134 "USA" labels are predicted as "UK", There are two reasons for this fact, the first one is that the frequency of "USA" and "UK" is much higher than other labels, so it is easy for classifiers to identify them; the second one is that the features of "UK" and "USA" are similar.

In addition, there are also many others labels are predicted as "USA" because the number of sample with "USA" is much higher than the number of sample with other labels.

Overall, the precision scores are higher than recall scores at most of the time. High precision score means that the classifiers returned substantially more relevant results than irrelevant

ones, and the lower recall means that the classifiers returned only part of the relevant results.

The reason of the higher precision score is that when the classifiers need to handle multi label classification tasks with huge amount of labels, the positive labels are much less than the negative labels, and the corresponding weight rewards of positive labels will be large.

## 4.6 Summary

This chapter show the macro averaging evaluation and micro averaging evaluation of all traditional machine learning classifiers and deep learning classifiers, and the discussion of these evaluation results. Then the training loss of deep learning methods are shown to figure out the convergence speed of different classifiers on both training set and validation set. Finally, by using confusion matrices as visualization tools, the detailed classification results are shown for error analysis.

The next chapter is the conclusion, it will summarize all the implementations and results, and figure out the direction of future work.

# Chapter 5

# Conclusions

## 5.1 Conclusion

Multi label text classification is an important challenging task in NLP field, developing multi label text classification systems entails developing systems that can determine the labels towards text. These systems would have a great effect on real life applications through providing automatic text classification to improve services or products in media industry.

There are two main difficulties when machine face to multi label text classification problems, the first one is that the number of labels related to the specific text is uncertain so it is hard to predict labels simply by choosing top labels after calculating probabilities, the other one is that there may be some dependencies between some labels so the classifier would lost some information if it ignore the dependencies.

In order to solve the main difficulties and recommend some useful algorithms for multi class classification problem in real life, this project has designed, implemented 4 traditional machine learning classifiers and 4 deep learning classifiers for multi label news classification. Finally consider both macro evaluation and micro evaluation as the evaluation indicators, and recommend the most reliable multi label classifiers by comparing the performance of different type of classifiers.

By computing the F1 score, precision score, recall score of both macro averaging and micro averaging after building models, findings show that the overall performance of deep learning classifiers are better than the overall performance of traditional machine learning classifiers. One of the biggest advantages of deep learning is the ability to be accelerated by using GPU, which greatly increases the training speed.

According to the result, all evaluation indicators show that CNN is a good classifier, in contrast to CNN, almost all evaluation indicators show that the Decision Tree is not very suitable for multi label classification. Although most traditional machine learning classifiers underperform than deep learning classifiers, SVM, which performs best in the traditional machine learning classifiers, is still a reliable traditional model can be helpful to solve multi label classification problem because of its high level macro and micro scores, sometimes it

even gain higher scores than deep learning classifiers.

In conclusion, in actual application in real life, some deep learning classifiers such as CNN and SVM could be used for the text classification task, these classifiers are able to handle multi label classification, which is the most common classification task in the industry field.

## 5.2 Future work

Although the project has been completed and the performance of some classifiers are reliable, the ability of classifiers to handle the weak dependencies between labels still need to be improved. The reason is that it is hard to judge and calculate the specific dependencies.Using hierarchical structure of neural networks or label encoding could have been done more efficiently in this field, because it consider more information into the classification task, it would be the future work of this project.

Another difficulty is the imbalance of dataset, selecting data manually is efficient in this project because the sufficient dataset, but if the data is insufficient, more methods are needed to deal with the problem of unbalanced dataset, for example, data enforcement is a good idea to increase information of training set and it can help to decrease the influence of imbalanced dataset.

# Bibliography


Aggarwal, C. C. (2014), *Data classification: algorithms and applications*, CRC press.

Bhatia, K., Jain, H., Kar, P., Varma, M. & Jain, P. (2015), Sparse local embeddings for extreme multi-label classification, *in* 'Advances in neural information processing systems', pp. 730–738.

Bird, S., Klein, E. & Loper, E. (2009), *Natural language processing with Python: analyzing text with the natural language toolkit*, " O'Reilly Media, Inc.".

Bishop, C. M. (2006), *Pattern recognition and machine learning*, springer.

Bishop, C. M. et al. (1995), *Neural networks for pattern recognition*, Oxford university press.

Boutell, M. R., Luo, J., Shen, X. & Brown, C. M. (2004), 'Learning multi-label scene classification', *Pattern recognition* **37**(9), 1757–1771.

Bradley, A. P. (1997), 'The use of the area under the roc curve in the evaluation of machine learning algorithms', *Pattern recognition* **30**(7), 1145–1159.

Breiman, L. (2001), 'Random forests', *Machine learning* **45**(1), 5–32.

Carvalho, G., de Matos, D. M. & Rocio, V. (2007), Document retrieval for question answering: a quantitative evaluation of text preprocessing, *in* 'Proceedings of the ACM first Ph. D. workshop in CIKM', ACM, pp. 125–130.

Collobert, R. & Weston, J. (2008), A unified architecture for natural language processing: Deep neural networks with multitask learning, *in* 'Proceedings of the 25th international conference on Machine learning', ACM, pp. 160–167.

Dai, W., Xue, G.-R., Yang, Q. & Yu, Y. (2007), Transferring naive bayes classifiers for text classification, *in* 'AAAI', Vol. 7, pp. 540–545.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018), 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805* .

Dilrukshi, I., De Zoysa, K. & Caldera, A. (2013), Twitter news classification using svm, *in* '2013 8th International Conference on Computer Science & Education', IEEE, pp. 287–291.

Elisseeff, A. & Weston, J. (2002), A kernel method for multi-labelled classification, *in* 'Advances in neural information processing systems', pp. 681–687.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep learning*, MIT press.

Goutte, C. & Gaussier, E. (2005), A probabilistic interpretation of precision, recall and f-score, with implication for evaluation, *in* 'European Conference on Information Retrieval', Springer, pp. 345–359.

Graves, A., Mohamed, A.-r. & Hinton, G. (2013), Speech recognition with deep recurrent neural networks, *in* '2013 IEEE international conference on acoustics, speech and signal processing', IEEE, pp. 6645–6649.

Han, E.-H. S., Karypis, G. & Kumar, V. (2001), Text categorization using weight adjusted k-nearest neighbor classification, *in* 'Pacific-asia conference on knowledge discovery and data mining', Springer, pp. 53–65.

Harrag, F., El-Qawasmeh, E. & Pichappan, P. (2009), Improving arabic text categorization using decision trees, *in* '2009 First International Conference on Networked Digital Technologies', IEEE, pp. 110–115.

Hosmer Jr, D. W., Lemeshow, S. & Sturdivant, R. X. (2013), *Applied logistic regression*, Vol. 398, John Wiley & Sons.

Joachims, T. (1996), A probabilistic analysis of the rocchio algorithm with tfidf for text categorization., Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science.

Joulin, A., Grave, E., Bojanowski, P. & Mikolov, T. (2016), 'Bag of tricks for efficient text classification', *arXiv preprint arXiv:1607.01759* .

Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M. & Klein, M. (2002), *Logistic regression*, Springer.

Korde, V. & Mahender, C. N. (2012), 'Text classification and classifiers: A survey', *International Journal of Artificial Intelligence & Applications* **3**(2), 85.

Lai, S., Xu, L., Liu, K. & Zhao, J. (2015), Recurrent convolutional neural networks for text classification, *in* 'Twenty-ninth AAAI conference on artificial intelligence'.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), 436.

Lee, J. Y. & Dernoncourt, F. (2016), 'Sequential short-text classification with recurrent and convolutional neural networks', *arXiv preprint arXiv:1603.03827* .

Manikandan, R. & Sivakumar, R. (2018), 'Machine learning algorithms for text-documents classification: A review', *Machine learning* **3**(2).

Norouzi, M., Fleet, D. J. & Salakhutdinov, R. R. (2012), Hamming distance metric learning, *in* 'Advances in neural information processing systems', pp. 1061–1069.

Pal, M. (2005), 'Random forest classifier for remote sensing classification', *International Journal of Remote Sensing* **26**(1), 217–222.

Panda, K. & Swain, D. K. (2011), 'E-newspapers and e-news services in the electronic age: an appraisal'.

Pennington, J., Socher, R. & Manning, C. (2014), Glove: Global vectors for word representation, *in* 'Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)', pp. 1532–1543.

Quinlan, J. R. (2014), *C4. 5: programs for machine learning*, Elsevier.

Rogati, M. & Yang, Y. (2002), High-performing feature selection for text classification, *in* 'Proceedings of the eleventh international conference on Information and knowledge management', ACM, pp. 659–661.

Russell, A. (2009), 'News bust; news boom', *Journalism* **10**(3), 365–367.

Sebastiani, F. (2002), 'Machine learning in automated text categorization', *ACM computing surveys (CSUR)* **34**(1), 1–47.

Sun, Y., Wong, A. K. & Kamel, M. S. (2009), 'Classification of imbalanced data: A review', *International Journal of Pattern Recognition and Artificial Intelligence* **23**(04), 687–719.

Suykens, J. A. & Vandewalle, J. (1999), 'Least squares support vector machine classifiers', *Neural processing letters* **9**(3), 293–300.

Tang, D., Qin, B. & Liu, T. (2015), Document modeling with gated recurrent neural network for sentiment classification, *in* 'Proceedings of the 2015 conference on empirical methods in natural language processing', pp. 1422–1432.

Trstenjak, B., Mikac, S. & Donko, D. (2014), 'Knn with tf-idf based framework for text categorization', *Procedia Engineering* **69**, 1356–1364.

Tsoumakas, G., Katakis, I. & Vlahavas, I. (2006), A review of multi-label classification methods, *in* 'Proceedings of the 2nd ADBIS workshop on data mining and knowledge discovery (ADMKD 2006)', Citeseer, pp. 99–109.

Yang, X., Macdonald, C. & Ounis, I. (2018), 'Using word embeddings in twitter election classification', *Information Retrieval Journal* **21**(2-3), 183–207.

Yong, Z., Youwen, L. & Shixiong, X. (2009), 'An improved knn text classification algorithm based on clustering', *Journal of computers* **4**(3), 230–237.