University of Padova

Department of Mathematics

Master Thesis in Data Science

Classifying community text and community groups using machine learning

Supervisor
Alberto Testolin
University of Padova

Master Candidate
Zesen Huang

Academic Year
2021-2022

Abstract

With the development of internet and social media, text mining and text classification are becoming key applications of natural language processing technology. Lots of texts, such as news, tweets, articles and blogs, are created on different internet platforms every day. For social media, like Reddit and Tweeter, it is thus very important to effectively label different groups or tweets by category to guarantee that users can easily find what they are interested in. Several methods and techniques have been developed to perform text mining and classification, most of which can be readily used with Chinese language. In this dissertation, I use $TF \times IDF$ (Term Frequency-Inverse Document Frequency) and bag-of-word approaches to extract keywords and represent texts using numerical vectors, whose dimensionality is then reduced using compression and feature selection methods. The resulting representations are given as input to Light Gradient Boosting Machine (*LightGBM*) and Extreme Gradient Boost (*XGBoost*) classifiers to implement an efficient Chinese text categorization. I also show that the performance of these classifiers can be inspected to provide interpretable explanations of their operations, which can allow to further improve efficiency by adjusting specific features. The data I used in this dissertation comes from Zhishixingqiu, which is a platform that provides community management tools for those who want to build a paid or free community with their fans or readers. This platform contains many groups and a huge amount of posts / articles, and each group is marked with corresponding labels (e.g., Finance and Economics, Science and Technology, Education and Reading, etc.) by expert staff.

Contents

Listing of figures

Listing of tables

Listing of acronyms

NLP . . . . . Natural Language Processing

BoW model . . . . Bag-of-words model

TF-IDF . . . . . Term frequency-inverse document frequency

PCA . . . . . Principal Components Analysis

GBDT . . . . . Gradient Boosted Decision Trees

LightGBM . . . . . Light Gradient Boosting Machine

XGBoost . . . . . Extreme Gradient Boost

SMOTE . . . . . Synthetic Minority Oversampling Technique

ADASYN . . . . . Adaptive Synthetic

CBOW . . . . . Continuous bag-of-words

DSG . . . . . Directional Skip-Gram

MSE . . . . . Mean Square Error

GOSS . . . . . Gradient-based One-Side Sampling

DAG . . . . . Directed Acyclic Graph

1 Introduction

For social networking service, text is one of the important part. Therefore, how to apply some specific techniques of natural language processing (NLP) to understand, manage, and classify a large number of texts is also an important task for Zhishixingqiu. In this work, I collect many text data that are labeled from Zhishixingqiu to construct two datasets and then build a text classification system on these two datasets. The goal is to use existing group labels and supervised learning to classify posts / articles and groups automatically. By using this text classification system, the company can assign different posts / articles and groups with different labels automatically and better manage content and groups on its community platform.

In general, a text classification system can be deconstructed into the following four parts.

- Features (text) representation and extraction.
- Features selection and dimension reduction.
- Classifier selection.
- Evaluation.

Text is one of the most common types of unstructured data. Unstructured data is information in many different ways that does not follow standard data models or schema, and is difficult to store and manage in a traditional relational database. For example, they are unlike binary values that can be represented as True or False, and also unlike human height or weight that have a certain range and unit.

Processing unstructured text data, we need to transform the text into numerical vectors because the text is unstructured and cannot be directly inputted into the model, such as Support Vector Machine(SVM) [2] and Decision Tree [3]. We should use some numerical values to represent text to make the computer understand. Therefore, the first step is to represent the text data and extract features / keywords. There are two kinds of method for representing text data: one is the bag-of-words model and the other is the Word2Vec model. The bag-of-words model is commonly used in document classification, where the occurrence (frequency) of each word is used as a feature to train a classifier. Compared to the bag-of-words model, which treats words as atomic units without considering the sequence of words in a sentence, Word2Vec uses a neural network model to learn word associations from a large corpus of text and creates a list of numbers for each word. Word2Vec measures the similarity between different words by calculating cosine similarity using vectors of words. For the extraction of Chinese text in this work, I use *Jieba*, which is a popular Chinese word segmentation module, to extract keywords based on $TF \times IDF$ (term frequency-inverse document frequency) from each document, and then those keywords and the bag-of-word model are used to build the vector matrix. Moreover, I also employ the Tencent pre-training Word2Vec model to compute similarity and to replace similar keywords. This processing is also known as feature representation and extraction and will be described in Section 2 of Chapter 3.

After vectorizing the text, we will have a vector matrix with thousands of dimensions, which also means that thousands of keywords are extracted from the corpus. Too many dimensions are not always beneficial to model learning. On the one hand, it may result in overfitting of the validation data set. On the other hand, the bag-of-word model is prone to generate a sparse matrix with many zeros because most keywords are low in frequency, and running a large sparse matrix will require many computation resources. Therefore, feature selection / dimension reduction is very necessary in text classification. In this work, I try to perform different dimension reduction techniques, including manual selection according to word frequency, Principal Component Analysis (PCA) [4], and Chi-square test [5], and the comparison will be made among them. I will show that, depending on the company's business and data, selecting or removing certain keywords is meaningful in a commercial environment. This part is described in Section 2 of Chapter 3.

The selection of classifiers plays an important role in the classification task. There are many algorithms that are developed to complete the classification task. Specifically, supervised learning models can be used, considering that the data which come from Zhishixingqiu's database is manually labeled by certain staff. In Section 4 of Chapter 3, I present a discussion on achieving

a text classification based on *XGBoost* [6] and *LightGBM* [7] and make a comparison among them. Similar to each other, *LightGBM* and *XGBoost* are based on the gradient booster algorithm, but they make some effective promotion and improvement, respectively. As machine learning algorithms, they are commonly used in many tasks, such as classification and regression. However, *LightGBM* can also be applied to perform ranking tasks.

The last step of this work is evaluation. Since this task is text classification and the labeled data is unbalanced, balanced accuracy will be applied to estimate the output of different classifiers. Moreover, precision, recall and F1 score are also useful to estimate the predictions in different ways. According to the evaluation, we are able to compare different classifiers' performance and make a trade-off among those metrics, and then select the best one.

2

Related Work

In the first chapter, I intend to provide some basic concepts of natural language processing used in this dissertation. Since in this work I process and classify Chinese text, some characteristic of Chinese processing methods and tools will also be described. Furthermore, key ideas of machine learning and imbalanced learning will also be discussed.

2.1 Natural Language Processing

As Figure 2.1 shows, NLP is an interdisciplinary discipline that includes linguistics, computer science, and artificial intelligence. The purpose of NLP is to use computers to process large-scale human language texts and try to make computers understand human language. In this way, we can find valuable information from it and make the computer capable of "communicating". The early NLP was based on many rules, and this approach was limited. However, with the development and application of machine learning, NLP has made great progress.

According to Figure 2.2, the research tasks of NLP include, but are not limited to, word segmentation, sentiment analysis, text classification, machine translation, and question orientation. In this work, I will discuss the following aspects of NLP in word segmentation, weighted words, text tokenization, and the final goal of text classification.

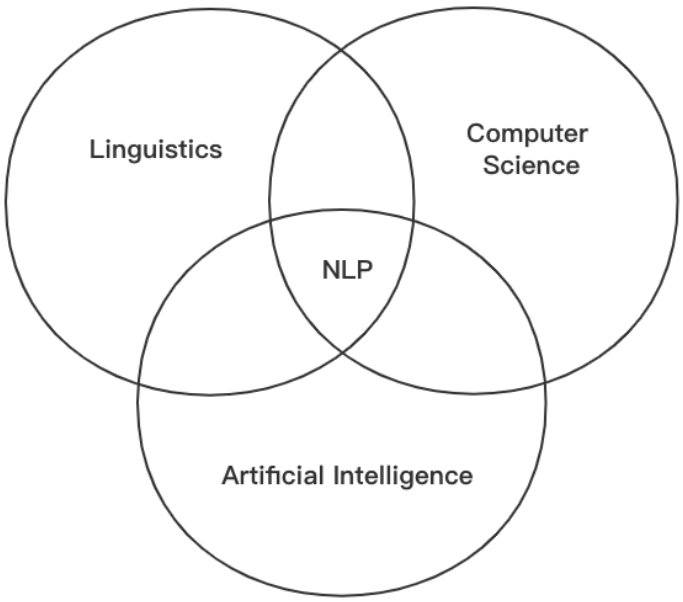

Figure 2.1: NLP is the intersection of linguistics, computer science and artificial intelligence

2.1.1 Stop words

There are many different words in different languages, but not every word has a specific meaning and is not important. Therefore, these words are usually filtered out. For example, '*a*', '*the*', '*and*', '*after*', '*yes*', and '*no*' are considered stop words in English. In Chinese, there are also stop words like '一个'(a), '之后'(after) and '是的'(yes). Note that there are no general stop words list for all NLP projects; it depends on the particular situation and context.

2.1.2 Tokenization

In English, a word is made up of multiple letters, a phrase can be made up of multiple words, and finally a sentence is made up of words, phrases, delimiters, and other meaningful elements. In this case, words, phrases, and delimiters are named tokens [8] which are identified based on the specific rules of the lexer. For example:

"Many English words are derived from Latin."

In this case, this sentence can be split into 7 tokens by blank. Here, the tokens are as follows:

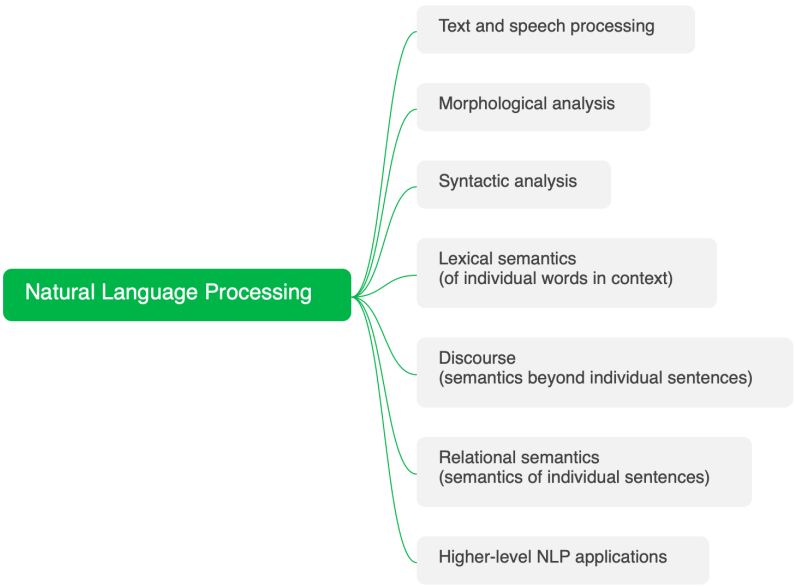

Figure 2.2: NLP common tasks.

["Many", "English", "words", "are", "derived", "from", "Latin"]

Identifying Chinese tokens is more difficult because Chinese words do not have a remarkable boundary like western languages. Here is an example of a Chinese sentence:

"我来到北京清华大学。" ("I come to Tsinghua University in Beijing.")

There are no blanks to separate the sentence. Therefore, previous knowledge about Chinese or a Chinese phrase dictionary is required. Depending on the context, in this case, the tokens can be extracted as follows:

["我", "来到", "北京", "清华", "大学"]

In general, tokenization [8] is a method of text pre-processing, which divides the text stream

into words, phrases, or other elements with specific meanings through tokens.

2.1.3 TF-IDF

First of all, some common concepts are listed before the discussion.

- Document: some text, such as articles.
- Corpus: a large collection of documents which is used for statistical analysis and hypothesis testing. For instance, we can calculate the frequency of a specific word on the basis of a corpus.

TF-IDF(term frequency-inverse document frequency) [9] is a common weighting technology used in information retrieval and data mining. It is often used to extract keywords from articles, and the algorithm is simple and efficient, which is often applied in the industry to clean text data from the beginning. It is used to evaluate the importance of a word for a document set or a document in a corpus. The importance of a word increases proportionally with the number of times it appears in the document but decreases inversely with the frequency of its appearance in the corpus. If a word appears frequently in one article and rarely in other articles, it is considered that the word or phrase has good categorization ability and is suitable for classification.

TF-IDF is expressed as :

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

Here, *TF* stands for word frequency in a document, which can be written as follows:

$$\text{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

where $f_{t,d}$ is the raw count of a term in a document, that is, the number of times that term t occurs in document d. Note that the denominator is simply the total number of terms in all documents d, which is the corpus.

Inverse document frequency, *IDF*, is a measure of how much information the word provides, that is, if it is common or rare in all documents. This can be obtained by dividing the total number of documents in the corpus N by the number of documents containing the term

and taking the logarithm of the resulting quotient:

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

For avoiding division-by-zero, it is common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

2.1.4 Jieba

Chinese NLP is different from English due to its challenge of word segmentation and representation. Also, a dictionary is usually necessary. In this dissertation, *Jieba* is used to perform word segmentation and extract keywords. Jieba is a popular and efficient Python-based Chinese word segmentation module, and its features are listed as follows.

- It achieves efficient word graph scanning based on a prefix dictionary structure. A directed acyclic graph (DAG) is constructed for all possible word combinations.
- *Jieba* uses dynamic programming to find the most probable combination based on word frequency.
- For unknown words, an HMM-based model [10] [11] is used with the Viterbi algorithm [12].

In addition, *Jieba* supports custom dictionary of stop words and new words for developers; thus I make my own dictionary of stop words according to the features of Zhishixingqiu's platform in this work. Words in the stop-word dictionary will be removed. Furthermore, *Jieba* can extract keywords from documents through *TF-IDF*, which plays a very important role in tokenization and feature representation.
Here is an example of extracting keywords by *Jieba* using *TF-IDF*:

Input:

text = "自然语言处理是人工智能和语言学领域的分支学科。此领域探讨如何处理及运用自然语言；自然语言处理包括多方面和步骤，基本有认知、理解、生成等部分。"(Natural language processing is a branch of artificial intelligence and linguistics. This field deals with the processing and use of natural language; Natural language processing includes many aspects and steps, including cognition, understanding, generation, and so on.)

Output:

keywords = [(' 自然语言', 1.3610793585913044),
(' 处理', 0.7057637813947826),
(' 领域', 0.4706343949921739),
(' 人工智能', 0.41121853893434784),
(' 语言学', 0.3895232708421739),
(' 认知', 0.37859655656913044),
(' 步骤', 0.3500588205230435),
(' 多方面', 0.3371455823613044),
(' 探讨', 0.30896048135217 39),
(' 分支', 0.303736815 62)]

For English:
keywords = [('natural language', 1.3610793585913044),
('processing', 0.7057637813947826),
('field', 0.4706343949921739),
('artificial intelligence', 0.41121853893434784),
('linguistics', 0.3895232708421739),
('cognition', 0.37859655656913044),
('steps', 0.3500588205230435),
('many aspects', 0.3371455823613044),
('deals with', 0.30896048135217 39),
('branch', 0.303736815 62)]

As we can see from the output, *Jieba* returns a list of keywords and their *TF-IDF*, sorted by *TF-IDF* value in descending order. That is, the importance of these keywords in the sentence decreases as *TF-IDF* decreases. *TF-IDF* is calculated by this sentence and *Jieba* default corpus. However, "自然语言处理"(natural language processing) are divided into "自然语言"(natural language) and "处理" (processing), but actually we want to acquire the first phrase, "自然语言处理"(natural language processing). This is because both of phrases have their own meaning, and also have another meaning when they are put together in Chinese. In order to solve this problem, *Jieba* allows developers to specify their own custom dictionary in order to include the *Jieba* words not available in its dictionary. Although *Jieba* has the ability to identify new words, adding new words itself can guarantee a higher accuracy rate.

2.2 Machine learning

Machine learning is a multidomain interdisciplinary subject involving probability theory, statistics, approximation theory, convex analysis, algorithm complexity theory, and other disciplines. It specializes in studying how computers simulate or realize human learning behavior to acquire new knowledge or skills and reorganizing existing knowledge structure to continuously improve its own performance. Machine learning can also be called statistical learning and is considered part of artificial intelligence. Machine learning reveals the underlying relationships in the data, which is not a fixed function defined in advance but is derived from historical data. When different data are input, the output of the machine learning algorithm changes, that is, the machine learning model changes. Therefore, machine learning can be understood as a mapping function between data inputs and outputs.

Machine learning approaches can be categorized into supervised learning, unsupervised learning, and reinforcement learning. In this work, supervised learning and unsupervised learning are briefly discussed in the following.

Supervised learning has corresponding outputs and labels on its input data. It is a machine learning task that infers a function from labeled training data. The training data includes a set of training examples. In supervised learning, each instance consists of an input object (usually a vector) and a desired output value. Supervised learning algorithms analyze these training data and produce an inference function that can be used to map out new instances. In these approaches, the label acts as a teacher, guiding the output of the model algorithm toward the target value. Common supervised learning algorithms include Logistic Regression [13], Support Vector Machine [2], K-Nearest Neighbor [14], Random Forest [15], etc. Supervised learning includes regression problems and classification problems. If the output is a finite discrete variable, the task belongs to classification, and if the output is a continuous variable, it is a regression task. In this dissertation, since I have made a discussion on a classification task about text, the data I collected are labeled. The label system will be described in Chapter 3, Section 1.

In general, supervised learning can be roughly divided into five steps.

1. Divide the data randomly into training and test sets.
2. Calculate the predicted value of the objective function using training data.
3. Construct a loss function such as the mean square error for regression problems. Here is $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$, where Y_i is the vector of true values, and $\hat{Y}_i$ is the vector

of prediction.

4. Employ an optimization algorithm like Gradient Descent [16] to update the parameters of the objective function until loss function converge.

5. Evaluate the result of the machine learning model by testing data.

The input data for unsupervised learning do not have a corresponding label, and the algorithm will explore the underlying structure and association of the data by itself. Since unsupervised learning does not require labeling, it does not require manual labeling compared to supervised learning. An important application of unsupervised learning is the Probability Density Function in the field of statistical learning, so we can estimate the distribution of the data by it without any previous knowledge. Another popular application is to use Cluster Analysis to perform classification. Euclidean distance is used to represent similarity in Cluster Analysis, and a small distance between two points in the feature space represents a large similarity and vice versa. By calculating the Euclidean distance between data points, similar data are classified into one category to achieve automatic classification. The clustering algorithm includes K-means [17] and Density-Based Spatial Clustering of Applications with Noise(DBSCAN) [18] [19] , etc.

2.3 Imbalanced-learning

Data plays an important role in machine learning, especially supervised learning in classification tasks. We know that the classification task in supervised learning will have several discrete values representing categories and that ideally the data used for learning will be evenly divided into several classes. However, sometimes we also encounter data imbalance, that is, the samples of each type of data are seriously unequal. For example, in the detection of financial fraud, the abnormal fraud data are only a small part of all the data, while the normal behavior data account for the large majority. In this work, my dataset is imbalance, whose number of the first three classes is twice, or even more than three times, the number of the other classes. If we do not address the problem of data imbalance, the final trained model will tend to predict the data as the majority class.

There are three schemes for dealing with imbalance data learning, which are oversampling, undersampling, and setting class weights. For over-sampling and under-sampling, their goal is to try to make dataset balanced by sampling more minority class or decreasing the proportion of majority class. Consequently, the ratio for all classes will be 1:1. Setting class weights does

not change the ratio of all classes, but assigns a minority class larger weight and a majority class smaller weight.

2.3.1 Over-sampling

The simplest way of oversampling is to randomly draw more repeated samples from the minority class [20], making the number of minority as large as the majority. In addition to random sampling with replacement, there are two popular approaches to oversampling minority classes: Synthetic Minority Oversampling Technique (SMOTE) [21] and Adaptive Synthetic (ADASYN) [22].

For SMOTE [21], the minority class is oversampled by selecting each minority class sample and introducing a composite example along the line segment that connects any/all k minority classes closest neighbors. Synthetic samples are generated as follows:

1. Calculate the difference between the feature vector (sample) considered and its nearest neighbor;
2. Multiply this difference by a random number between 0 and 1;
3. Add it to the feature vector under consideration.

If the feature dimension of the samples is two, then each sample can be represented by a point on the two-dimensional hyperplane. For instance, denote that:

- x_i is one of the samples.
- $x_{i(nn)}$ is a sample randomly selected among k nearest neighbors.
- x_{i1} is one of the synthesized data, and

$$x_{i1} = x_i + \lambda \cdot \left(x_{i(nn)} - x_i \right),$$

  where λ is a random number between 0 and 1.

A new sample x_{i1} synthesized by the SMOTE algorithm is equivalent to a point on the line segment between the point x_i and the point $x_{i(nn)}$. So, the algorithm is based on "interpolation" to synthesize new samples.

Another oversampling algorithm, ADASYN [22], is motivated by the success of synthetic approaches of SMOTE and is improving on it. ADASYN is based on the idea of adaptive

generation of minority data samples according to the distribution of minority data samples, which generates more composite data for minority class samples that are difficult to learn. The ADASYN algorithm can not only reduce the learning bias caused by the unbalanced distribution of original data but also adaptively change the decision boundary and focus on those samples that are difficult to learn. To achieve adaptively synthetic data, density distribution $\hat{r}_i$ is used as a criterion to automatically decide the number of synthetic samples that must be generated for each minority data example.

In summary, Figure 2.3 clearly illustrates the difference between resampling with random oversample, SMOTE oversample, and ADASYN oversample:

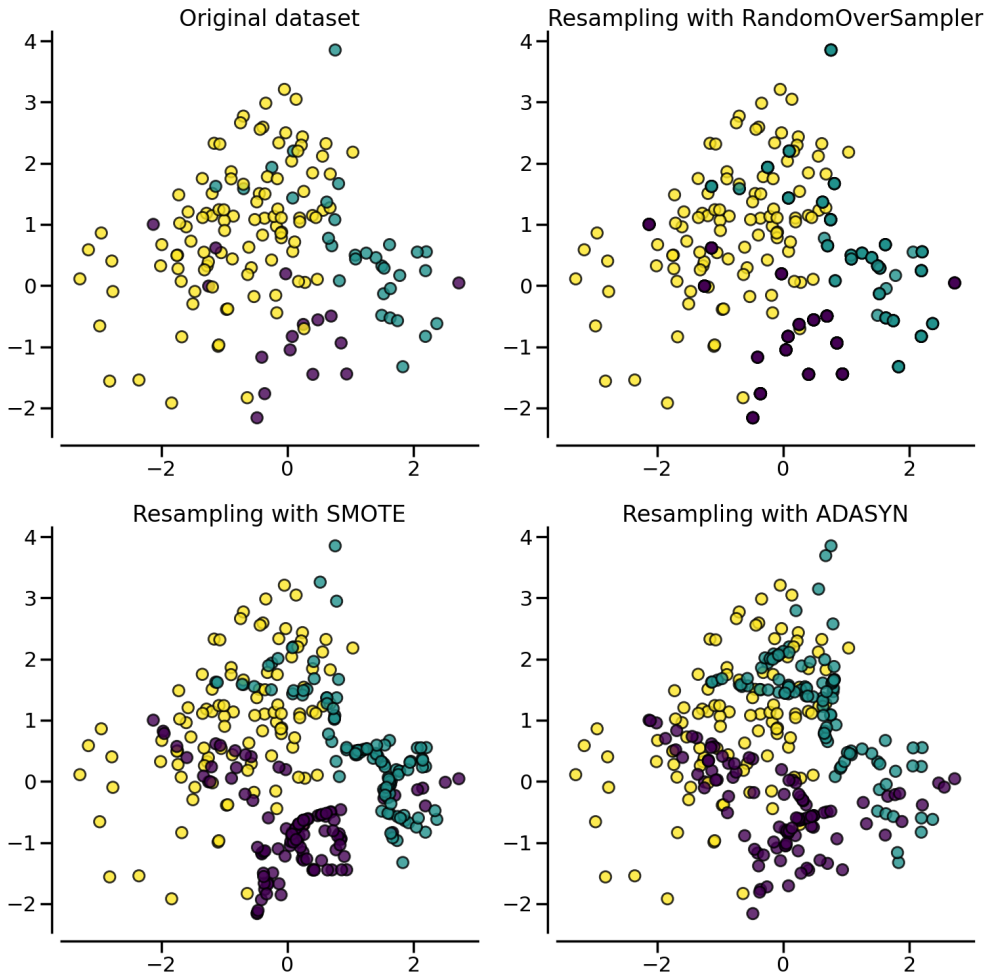

Figure 2.3: Major difference of the different over-sampling methods. (https://imbalanced-learn.org/stable/auto_examples/over-sampling/plot_comparison_over_sampling.html)

2.3.2 Under-sampling

Compared to oversampling, undersampling reduces part of the data from the majority class, to achieve the purpose of data balance. With this approach, the number of majority classes is reduced to approximately the same as that of the minority class. There are two ideas to reduce the number of majority class: generating from original data and selecting from original data.

Given an original data set S, generation algorithms will generate a new set S' where $|S'| < |S|$ and $|S'| \nsubseteq |S|$. In other words, the reduced samples obtained by this undersampling method are no longer the original samples but newly generated according to the original data characteristics. For generation algorithms, it is common to under-sample by generating centroids based on clustering methods. Algorithm *ClusterCentroids* makes use of K-means to reduce the number of samples. Therefore, each class will be synthesized with the centroids of the K-means method instead of the original samples. This algorithm keeps N majority samples by fitting the K-means algorithm with N cluster to the majority class and using the coordinates of the N cluster centroids as the new majority samples. However, the data must be grouped into clusters when applying this method. In addition, the number of centroids should be set so that the undersampled clusters are representative of the original one.

This undersampling algorithm is illustrated in Figure 2.4.

The other ideas to reduce the number of majority classes, the selection algorithm, are defined as this: given an original data set S, the algorithms will generate a new set S' where $|S'| < |S|$ and $|S'| \subseteq |S|$. Depending on whether the sample size can be decided by the user, the selection strategy can be divided into two: (i) controlled undersampling techniques and (ii) cleaning undersampling techniques. (i) allows users to decide the number of samples, while (ii) does not.

One of the (i) is to randomly draw a subset of data from the majority class, and this is an efficient and fast approach to achieve data balance. In this work, I mainly employ this undersampling method to reduce the majority, making data balanced, and making a comparison with other imbalanced-learning methods.

2.3.3 Class weighting

The last approach of imbalanced learning is setting class weights. It has a lower complexity and is already built into Scikit-learn classification models. All classification models in Scikit-learn have a hyperparameter called class weight. This hyperparameter is designed to control the

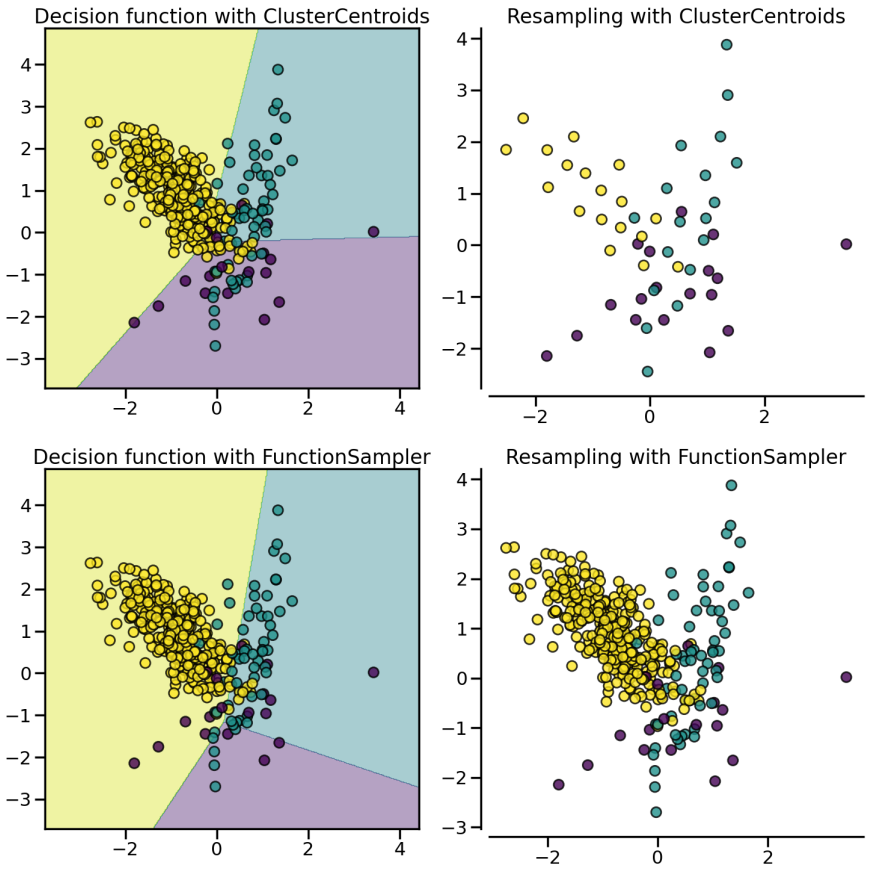

Figure 2.4: An example of ClusterCentroids.(https://imbalanced-learn.org/stable/auto_examples/under-sampling/plot_comparison_under_sampling.html)

trade-off between precision and recall, in other words, for the same purpose as dealing with data imbalances.

The idea is to assign weights to each class so that the weighted observations for all classes sum up equally. For two classes, where n is the number of observations and w is the weight:

$$n_{\text{minority}} \times w_{\text{minority}} = n_{\text{majority}} \times w_{\text{majority}}$$

Naturally, the weight of the minority class will be higher than the weight of the majority class. See Figure 2.5. When training, machine learning models pay more attention to observations with higher weights. By choosing weights in this way, we can compensate for the smaller number of observations in the minority.

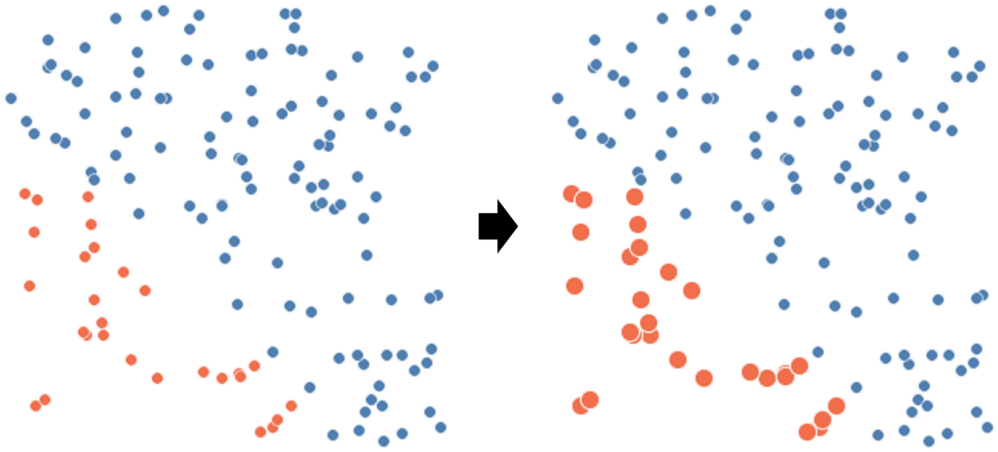

Figure 2.5: Class weight works by making minority become more important.

3 Methodology

In this chapter, I will describe all details of the text classification engineering on Zhishixingqiu's dataset, including dataset, features (text) representation and extraction, features selection and dimension reductions, classifier selection, and evaluation.

3.1 Dataset

This section is about the collection and description of data, thus the data source, Zhishixingqiu platform, data structure, and labels system are introduced.

3.1.1 What is Zhishixingqiu?

Zhishixingqiu is a Chinese Internet community. Provides a community tool for creators to connect hard-core fans, manage high-quality content, and make money by sharing knowledge. Users can capture and record inspiration by creating articles, audios, and videos at any time. In other to connect and communicate with fans, users can choose to create free or charge groups for their own, and invite their fans to join their groups. For charge groups, group members should pay the membership fee. As a group owner, you can manage the contents and members of the group, assign tasks, and answer free or paid questions from the members of the group. Moreover, the platform provides many functions and data support to group owners to facilitate group management. On this platform, there are thousands of different fields of groups,

including Finance and Economy, Internet, Artificial Intelligence, Photography, Biology, Art, Fashion, and so on.

3.1.2 Data structure

The text data used in this work is collected from Zhishixingqiu's database by SQL. There are some tables recording every group, user's post, and group's label in the database. For each group, it has a group name, unique group id, and a description of what this group is about. The description of groups has useful information because owners of group tend to introduce their specialized field, job, and interest here, which does help to build up features with these different and important keywords. Group owners can also set any post that is considered relevant and important to the group as *Important Posts*, which is like a label in groups. In other words, the *Important Posts* gathers the best and most valuable content in the group, and this part can work well to identify different fields of groups. Furthermore, Zhishixingqiu also provides *Columns* to group owners to implement classification and management of their contents. The contents of *Columns* can also represent the theme of the group well. Therefore, to collect text data from the Zhishixingqiu community to train the model, I choose to extract the first 1000 Chinese characters from each post from different groups' *Important Posts* and *Columns*.

3.1.3 Labels system

Thanks to the community management, most of the groups are labeled with different tags by staff according to the features and attributes of the groups. Community operation staff will read and consider the most posts in the groups, and decide which labels should be given to them. For example, if a group with the most posts, articles, or discussion are related to economy, then it will be labeled with Finance and Economy, thought it also has a few other topics. However, if the group has the same proportion of two or more topics of posts, it will be tagged with two or more related labels. In general, most groups are tagged with only one label, and a small number of groups are tagged with more than one label. When I collect text data, I only choose posts that come from groups with one label.

The labels of these groups can also be regarded as the labels of the posts in the group, because the manual labeling is also decided by the topic of the posts in the group. It is also useful to classify posts and use them to label groups. For example, I first classify the posts, then calculate the proportion of each post type, and use the post label with the first proportion as the group

group id	topic id	group name	description	post content	label
871	32	黄博的机器学习圈子	黄博士，国内机器学习布道者，github 上 star 排名前 60（48000+ star）。这里有名企的技术高管，国外高校的教授，数据竞赛的冠军收割者，热门公众号博主，以及 500 多位博士…	求问：使用 tensorflow+cnn 训练模型的时候每次进行到第二次迭代的时候就报内存不足，在网上查阅资料也设置对应的参数限定按需分配，也设置了最多使用多少 gpu。数据类型大概有一千二百个分类左右，数据只要超过四十万就会出现这种内存不足的问题。这可能是什么原因呢，要如何解决呢	AI 数据
871	32	Dr.Huang's Machine Learning Circle	Dr. Huang, a machine learning evangelist in China, is ranked among the top 60 stars on github (48000+ star). There are technology executives from famous companies, professors from foreign universities, reapers from data competitions, bloggers from popular public accounts, and more than 500 PhDs	Question: When using tensorflow+cnn to train the model, the memory was reported to be insufficient at the second iteration each time. I also set corresponding parameters to limit on-demand allocation and the maximum number of GPUs to be used by consulting materials online. There are about 1200 classes of data types, and this kind of out-of-memory problem occurs whenever data exceeds 400,000. What might be the reason for this, and how can we fix it?	AI and Data

Table 3.1: The example of data

label. This also has the advantage that if we need to assign multiple labels to groups, the first few labels of posts within the group can be used.

I make statistics on the number of group labels existing in the community, and the statistical results are shown in Figure 3.1.

As we can see, most groups are tagged as Finance and Economy in Zhishixingqiu's community, and the second and the third labels are Education and Industry Communication. The number of groups of these three labels is much more than others, accounting for more than

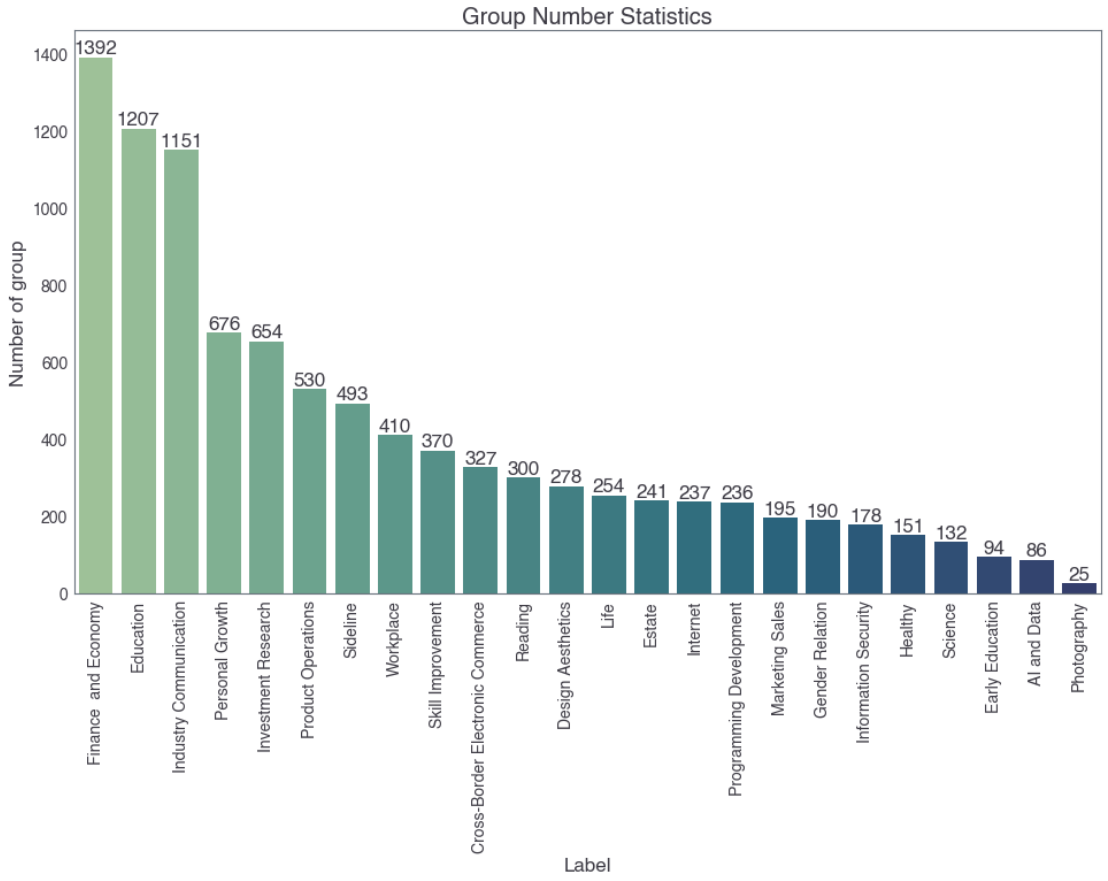

Figure 3.1: All groups Number Statistics.

a third of the total number. However, some groups are in a very small proportion of the total. Note that this is an imbalanced dataset and imbalanced-learning method should be applied when training classifiers, otherwise the classifier will be skewed towards groups with a high number of labels.

3.2 Features representation and extraction

As mentioned before, text is unstructured, especially Chinese text, because of an unremarkable boundary. Therefore, the Chinese text from Zhishixingqiu's groups should be well represented and extracted at the beginning of this project. It is important for machine learning to process string stream into words or phrases, because commonly machine learning model can not learn from sentences. The procedure by which we transform the text into another structure that ma-

chine learning model can solve is called feature engineering, and also includes feature selection and dimension reductions proposed in the next section.

For data pre-processing, special symbols, Web page links, emoji, English words, and digits by regular expression from the original text are removed. The reason why English words and digits are removed is that they account for a small proportion in Chinese text, and most of them are not that important for understanding a sentence. Also, if we do not remove digits, they will be extracted as keywords, and this scenario is very common when people talk about stock and security for a certain day in financial groups. For example, most people will write down "*2022 年 7 月 18 日*"(July 18, 2022) when they discuss stock market on this day, and "2022", "7", and "18" will be considered as keywords, which are meaningless.

The first step of feature engineering is to use *Jieba* to segregate words and extract keywords by *TF-IDF*. In this work, I decide to make *Jieba* return 10 or 20 keywords from each post/article text. Too many keywords are not helpful for representing each text because some meaningless words will also be extracted. This leads to an increase in features, which leads to a dimension explosion in the subsequent construction of the matrix representation data and introduces noise into the model. On the other hand, too few keywords are also helpless for representing documents due to missing information. After extracting keywords from all text of posts/articles of the data set, we have a corpus, or collection, of all keywords. This corpus is used to construct the bag-of-words model for text tokenization.

3.2.1 Bag-of-words model

Bag-of-words(BoW) model is a common document representation method in information retrieval and natural language processing, and it is one example of a vector space model. The bow model ignores its word order, grammar, syntax, and other elements and treats it as just a collection of several words. Therefore, a text (such as a sentence or a document) can be represented as the bag (multiset) of its words. For text classification using machine learning, it is commonly applied to complete feature extraction, in which the occurrence of (frequency of) each word is used as a feature for training a classifier.

Here is a simple example for English language:

"*It was the best of times,* "
"*it was the worst of times,*"
"*it was the age of wisdom,*"

"it was the age of foolishness,"

We treat each line as a separate document, and the four documents make up a corpus. Based on these four text documents, a list is constructed as follows for this corpus:

BoW1 = [*"it", "was", "the", "best", "of", "times", "worst", "age", "wisdom", "foolishness"*]

Therefore, this list can be used to represent each document in this corpus, and the order of words in the list is not important. Here, I use binary value, 0 or 1, to represent occurrence of each word. If the word appears in the list, assign 1, otherwise assign 0. As a result, we are able to tokenize the documents and use binary vectors to represent text, making the computer understand human language.

"It was the best of times, " = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
"it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

We can learn from this example that using the occurrence of each token(word) from corpus can represent any document in corpus. However, we lost the order information of the words in the document using this method, because a word naturally has no order at all. Alternatively, it is also common to use the *n-gram* model to store spatial information. Note that gram stands for token or word. In this example, we treat each word as a feature, thus this is a *1-gram* model. If we try the *2-gram* model, a list is constructed as follows:

BoW2(1) = [*"it was", "was the", "the best", "best of", "of times"*]
BoW2(2) = [*"it was", "was the", "the worst", "worst of", "of times"*]
BoW2(3) = [*"it was", "was the", "the age", "age of", "of wisdom"*]
BoW2(4) = [*"it was", "was the", "the age", "age of", "of foolishness"*]

In the end, we merge the *2-gram* word lists of each document to obtain the final BoW model feature.

BoW2 = [*"it was"*,

"was the", *"the best"*,

"best of", *"of times"* ,

"the worst", *"worst of"* ,

"the age", *"age of"*,

"of wisdom" , *"of foolishness"*]

Since Chinese text has no remarkable boundary, it is common to do word segregation before construct a BoW model to represent text. According to [23], it shows that a combination of *1-, 2-grams* is little better than that of *1-, 2-, 3-grams* for the classification of Chinese text. In this work, most of the keywords extracted by *Jieba* consist of two or three Chinese characters. Therefore, I use these keywords to make up the corpus and construct the BoW model to represent the text directly, without applying *n-gram*. Moreover, using keywords as features is more explicable than using *n-gram* for company business, because the extracted keyword is a meaningful and complete word in the Chinese context, while sometimes the words constructed by *n-gram* are not all complete and meaningful.

However, for a very large corpus (e.g. thousands of books), the length of the vector might be thousands or millions of positions. In addition, each document may contain only a few known words in the vocabulary. This results in vectors with many zeros, known as sparse vectors or sparse representations. When modeling, sparse vectors require more memory and computational resources, while the large number of dimensions is very challenging for traditional algorithms. In other to reduce the dimensions of BoW model, one of useful approaches is removing stop words when tokenizing text. For instance, almost every English document has these two words: "*a*" and "*the*", and these are meaningless for representing documents. Thus, we can collect stop words like these, form them into a stop words dictionary, and filter them out when we build the BoW model to represent the text. It is also significant to employ Principal Components Analysis(PCA) and Chi-square test to perform dimension reductions, and this part will be introduced in the next section.

Although the BoW model discards word order and context information, which will lead to some misunderstanding of language, it is simple and robust and it is successful in applying it to text classification.

3.2.2 Word Embedding

To deal with new words that the BoW model does not have, a pre-training word embedding model provided by Tencent is used to replace new words with similar words that the BoW model already has in this work. The pre-training model comes from Tencent's AI Lab, which provides a large-scale, high-quality Chinese word vector set. It contains a total of more than 8 million Chinese words, each of which corresponds to a 200-dimensional vector.

Word embedding is another approach to representing text and transforming text into vectors using neural networks. Compared with the BoW model's sparse representation, word embedding provides a means of efficient and dense representation where similar words have similar encodings. Word embedding can be understood as a function that maps words into another N dimension feature space, that is $f(X) \rightarrow Y$. Therefore, word embedding is also a feature learning technique, which uses an N dimension vector of real numbers to represent each word or phrase from the vocabulary.

Various word embedding methods have been proposed to convert words into understandable inputs for machine learning algorithms, including Word2Vec [24], GloVe [25], and FastText [26]. Since the Chinese word embedding model employed in this work is based on Word2Vec, some introductions about Word2Vec are described below.

The Word2Vec approach uses shallow neural networks with two hidden layers, continuous bag-of-words (CBOW), and the Skip-gram model to create a high-dimensional vector for each word.

- If a word is used as input to predict the context around it, the model is called a Skip-Gram model;
- If the context of a word is used as input to predict the word itself, it is the *CBOW* model.

Figure 3.2 shows a simple architecture of the *CBOW* model and the Skip-gram model.

For the first step of the *CBOW* model, we construct a corpus including V words and use one-hot encoding, which is similar to the BoW model, to represent each word. For example, if "语言"(language) is the first word in the corpus, then it will be represented as [1, 0, 0, ..., 0], only one for "语言" out of V units, $\{x_1, \cdots, x_V\}$, will be 1, and all other units are 0. Thus, this is the input of neural networks for "语言" when considering just one word in the context.

CBOW neural networks have three layers: input layer, hidden layer and output layer, and the units on adjacent layers are fully connected. Since the corpus has V words, the dimensions

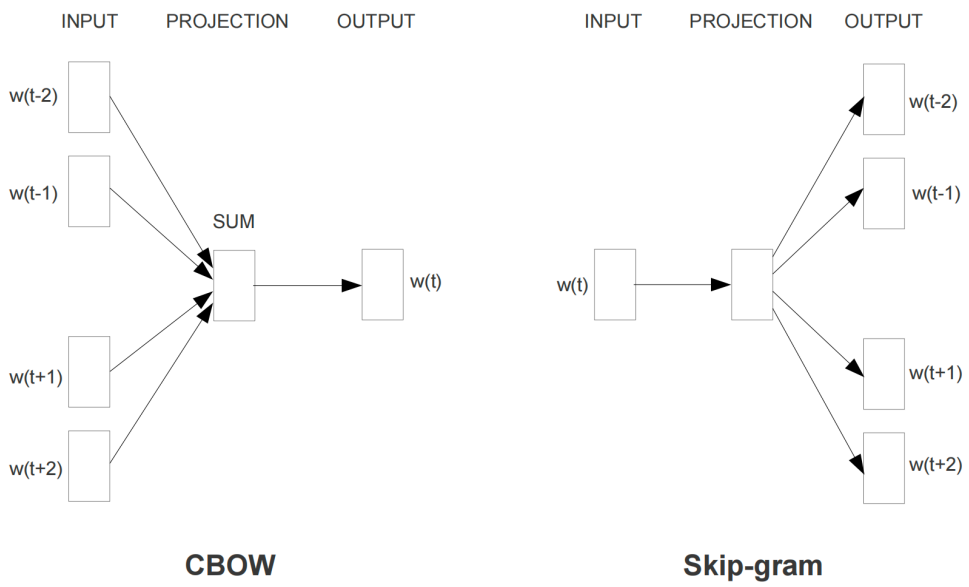

Figure 3.2: The continuous bag-of-words (CBOW) architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words based on the given current word.

of the input layer and the output layer are also V. And we set the dimension of hidden layer is N, so the weights between input layer and hidden layer can be represented as $W_{V\times N}$, and $W_{N\times V}$ represents the weights between hidden layer and output layer. For the Chinese word embedding model used in this dissertation, the N is equal to 200. The architecture of *CBOW* with one word is shown in Figure 3.3.

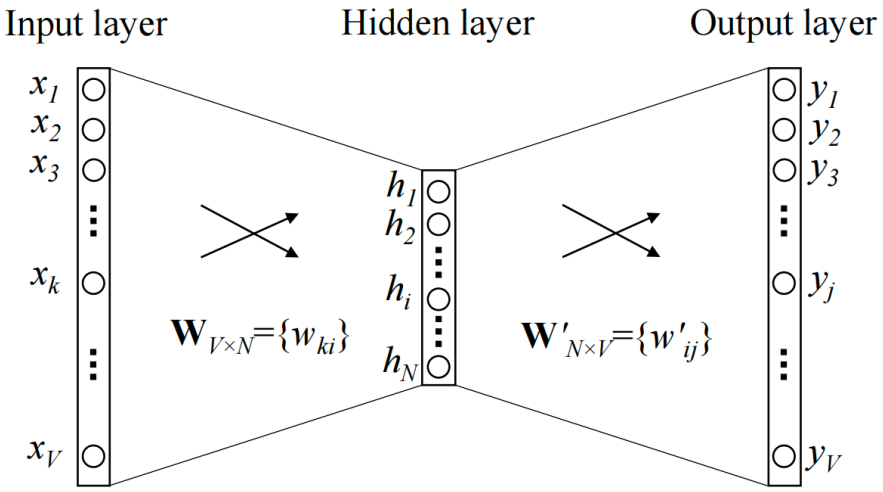

Figure 3.3: A simple CBOW model with only one word in the context. [1].

Each row of W is the N dimension vector representation v_w of the associated word of the input layer, which means row i of W is V_w^T. Given a context (a word), assuming $x_k = 1$ and $x_{k'} = 0$ for $k' \neq k$, we have

$$h = W^T X = W^T_{(k,.)} := V^T_{w_I},$$

where V_{w_I} is the vector representation of the input word w_I. Note that the activation function of the hidden layer is actually linear, which implies that the previous layer directly passes its weighted sum of inputs to the next layer.

From the hidden layer to the output layer, a score u_j can be computed for each word in the corpus by another weight matrix $W'_{N\times V} = \{w'_{ij}\}$:

$$u_j = {V'_{wj}}^T h,$$

where V'_{wj} is the j-th column of the matrix W'.

Then we can use softmax, a log-linear classification model, to obtain the posterior distribution of words, which is a multinomial distribution.

$$p(w_j|w_I) = y_j = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})},$$

where y_j is the output of the j-the unit in the output layer. Based on the softmax function, we can predict the middle word with future words and history words. The number of words used depends on the setting of the window size (the common size is 4–5 words).

As for multi-word context, the representation and process are similar to one-word context, except that the *CBOW* model takes the average of the vectors of the input context words, and uses the product of the input $\rightarrow$ hidden weight matrix and the average vector as the output. Figure 3.4 shows the *CBOW* model with a multi-word context setting.

Unlike *CBOW*, the Skip-Gram model predicts surrounding words based on the given current word. Thus, the target word is now at the input layer and the context words are on the output layer. Figure 3.5 shows the skip-gram model.

When the neural network is trained by back propagation, the final result of word vector is actually the weight of the neural network, so we can represent word w_i in the corpus by the vector w_{ki} from $W_{V\times N}$ as mentioned before.

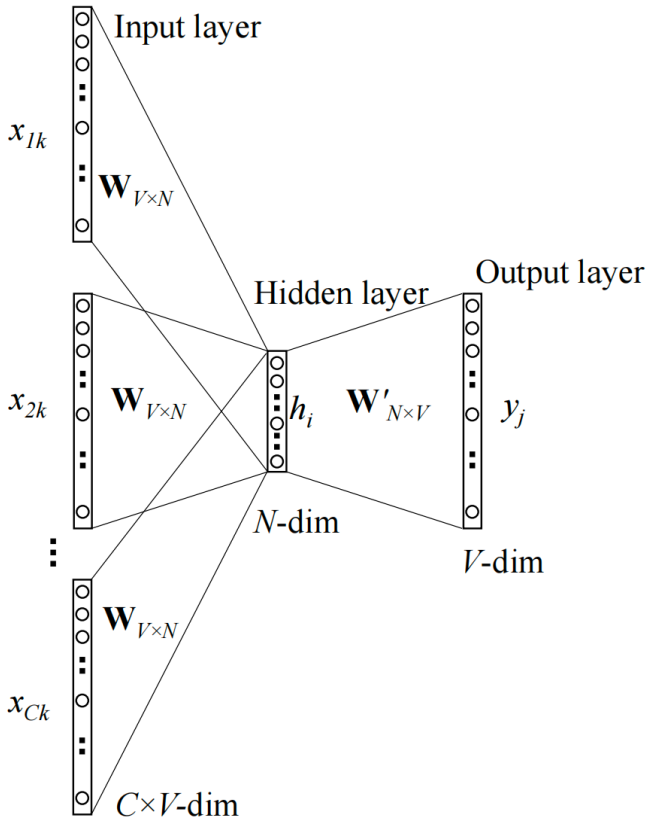

Figure 3.4: Continuous bag-of-word model . [1].

Tencent AI Lab adopts the self-developed Directional Skip-Gram (DSG) algorithm as the word vector training algorithm. The DSG algorithm is based on the Skid-Gram model. On the basis of the cooccurrence relation of word pairs in the text window, the relative position of word pairs is considered to improve the accuracy of the semantic representation of word vectors.

Gensim, a popular and free NLP module based on Python, is applied to load and employ the Chinese Word2Vec model in this work. By using this Word2Vec model, we can compute the similarity between different words. For example, the similarity between " 机器学习"(machine learning) and " 深度学习"(deep learning) is given as 0.93376374, " 男人"(man) and " 女人"(woman) is given as 0.9531477, which both imply a great similarity. I mainly use this Chinese Word2Vec model to replace new words with similar words that are in the trained BoW model words set. When the BoW model's features/words are set, it can be used to tokenize and represent some new text and then input it to the classifier to predict. However, some words

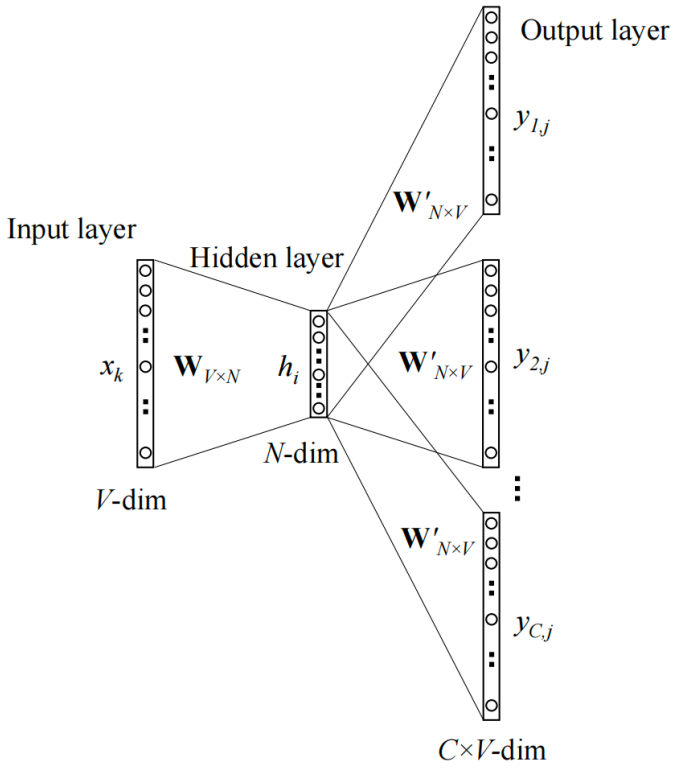

Figure 3.5: The skip-gram model. [1].

are not in the set of words of the existing BoW model, and if we ignore these words, some useful information might be lost. Therefore, I exploited Tencent's Chinese Word2Vec model to replace a new word with an existing word.

Denote that existing words set is $S = \{s_1, s_2, ..., s_k\}$ with k words, and given a new word $s_{new} \notin S$. Using the pre-training Chinese Word2Vec model, we can get the most similar word s_i to s_{new} from S, where

$$s_{new} \approx s_i \subseteq S.$$

As a result, we can use the word s_i to replace the new word s_{new}.

3.3 Features selection and dimensionality reduction

When we have a large corpus and use the BoW model to extract and represent features, a huge and sparse matrix will be generated. This is one of the limitations of BoW model, as the amount of training text increases, the dimension of the transformed vector becomes very large, so it is

easy to cause the curse of dimension. A large sparse matrix is inefficient because it has many zero values and occupies a lot of memory and computing resources. Moreover, sparse representations are difficult to model and the challenge is that the model uses very little information in such a large representation space.

Therefore, I try to perform feature selection and dimension reduction before model building. Dimension reduction is a preprocessing method for high-dimensional feature data. It retains some of the most important features of high-dimensional data and removes noise and unimportant features, to achieve the purpose of improving the data processing speed. In actual production and application, dimension reduction can save us a lot of time and cost within a certain range of information loss. In this work, the BoW model consists of hundreds of thousands of keywords, which is a challenge for modeling. In addition, not every keyword is worth considering and is useful to explain in business. Thus, it is significant to select the most important keywords/features among the whole set.

3.3.1 Principal Component Analysis

Principal Component Analysis (PCA) [27] [4] is used to decompose a multivariate data set into a set of successive orthogonal components that explains the maximum amount of variance. PCA can be thought of as an unsupervised learning problem. The main idea of PCA is to map N-dimensional features to K-dimensional features, which are new orthogonal features, also known as principal components, and to reconstruct K-dimensional features on the basis of the original N-dimensional features. In fact, PCA is equivalent to retaining only the dimensional features, which contain most of the variance, and ignoring the feature dimensions that include almost zero variance, so as to achieve dimensionality reduction processing of data features.

The realization of PCA algorithm based on eigenvalue decomposition of covariance matrix can be simplified as follows:

Input: dataset $X = \{x_1, x_2, ..., x_n\}$ with n dimensions, need to be decreased to k dimensions.

1. Compute the mean $\bar{x}_i$ for every dimension of the entire dataset, and then subtract the mean for each dimension value, which is decentration;

2. Compute the covariance matrix $\frac{1}{n}XX^T$;

3. Use the eigenvalue decomposition method to find the eigenvalues and eigenvectors of the covariance matrix $\frac{1}{n}XX^T$;

4. Sort the eigenvalues by decreasing and choose the largest k of them. Then the corresponding k feature vectors are taken as row vectors, respectively, to form the feature vector matrix P;

5. Convert the data into a new space constructed by k feature vectors, namely $Y = PX$.

In summary, properties of PCA can be listed below:

- Mitigate curse of dimensionality: PCA algorithm can increase the sampling density of samples (because the dimension decreases) by discarding part of the information, which is an important means to mitigate dimensional disasters;
- Noise reduction: when the data are affected by noise, the feature vector corresponding to the minimum eigenvalue is often related to noise, and to some extent the effect of noise reduction can be achieved by abandoning them;
- Overfitting: PCA retains primary information, but this primary information is only specific to the training set and this primary information is not necessarily important. Not having a great performance in the training set, some seemingly useless information may be discarded, which happens to be important information, so PCA may also aggravate the overfitting;
- Feature independence: PCA not only compresses the data to low dimension, but also makes the features of the data independent of each other after dimensionality reduction;
- Accelerated computing: reduce the computational overhead of the algorithm to speed up the data processing.

Furthermore, after PCA dimension reduction, the data is no longer the original data. For example, feature x_1 and x_2 represent " 数据"(data) and " 信息"(information) respectively in the BoW model, and their values are binary(0 or 1) to represent if a text has these keywords. Then after PCA, there are no more feature x_1 and x_2, and they are used to generate new features with other original features. Their values are no longer binary, but some other continuous values. Although PCA is beneficial to dimension reduction, the synthesized eigenvalues are not commercially interpretable well for Zhishixingqiu.

3.3.2 Chi-square test

Chi-square detection [5] is a statistical method based on the Chi-square distribution, which is often used to detect the relationship between categorical characteristics in classification problems, to achieve the purpose of feature selection.

If n independent random variables z_1, z_2, z_3, ..., z_n are subject to the standard normal distribution, $z_i \sim N(0,1)$. Then the sum of the squares of these n random variables subject to the standard normal distribution forms a new random variable X^2, and its distribution rule is called the chi-square distribution, which can be defined as follows:

$$X^2 = \sum z_i^2.$$

Chi-square distribution is a new distribution constructed from the normal distribution. When the degree of freedom df is large, the distribution is approximately normal. Figure 3.6 shows the chi-square distribution for different degrees of freedom.

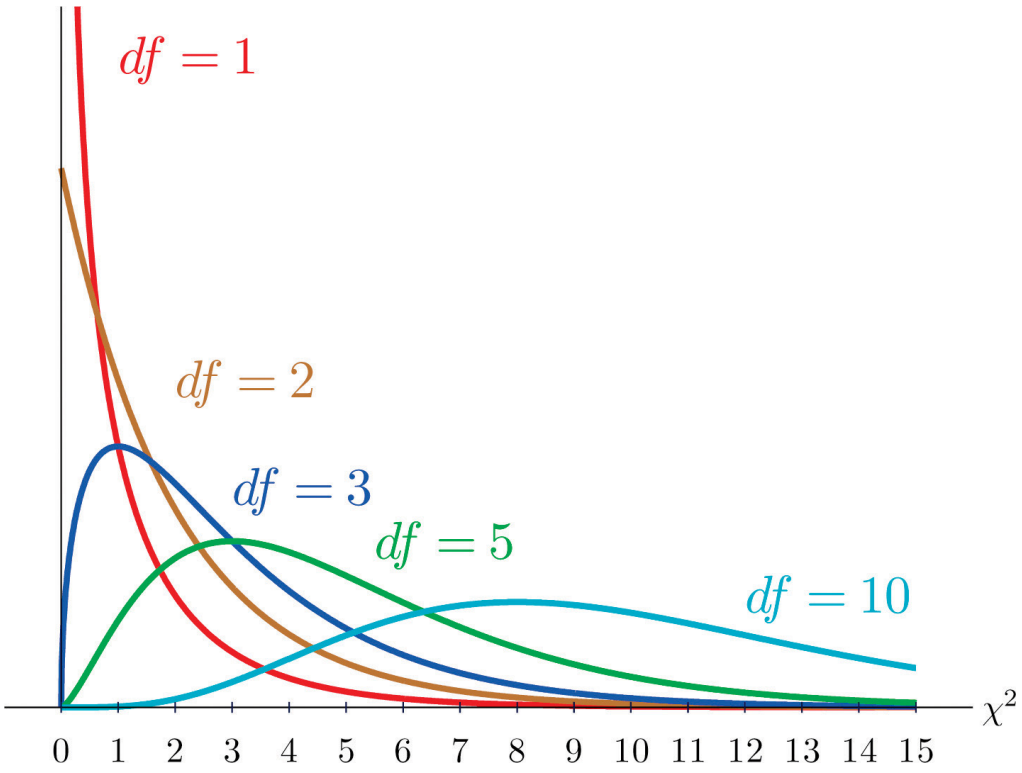

Figure 3.6: Examples of chi-square distributions with different degree of freedom.

The idea of a chi-square test is to infer whether there is a significant difference between the population distribution and the expected distribution based on the sample data. The chi-square test measures the absolute magnitude of the deviation between the actual value and the expected value, and the relative magnitude of the difference and the expected value. The mathematical expression can be written as follows:

$$X_c^2 = \sum \frac{(A_i - E_i)^2}{E_i},$$

where the A is actual value , E is expected value, and c is the degree of freedom.

When two features are independent of each other, the actual values are close to the expected values, so the chi-square values will be smaller. Therefore, a high chi-square value indicates that the independence hypothesis can be rejected. Generally speaking, the higher the chi-square value, the stronger the dependence of the feature on the response, and the feature can be selected for model training.

Steps to perform the chi-Square test:

1. Define the hypothesis.
2. Build a contingency table.
3. Find the expected values.
4. Calculate the chi-Square statistic.
5. Accept or Reject the Null Hypothesis.

In this project, the chi-square test can be exploited to remove the features that are the most likely to be independent of class, which are irrelevant for classification task. And we can keep the rest of high relevant features / keywords, so as to reduce the number of keywords to achieve dimension reduction.

3.4 Classifier selection

According to [28], there are many machine learning algorithms that can be used to solve text multiclassification problems, including but not limited to Logistic Regression [13], Support Vector Machine [2], K-Nearest Neighbor [14], Random Forest [15], Gradient Boosting and so on. In addition, *XGBoost* [6] and *LightGBM* [7] that are based on Gradient Boosting framework have gained a lot of traction in recent years. They are very common and excellent in various competitions and industries. To build a great classifier for text classification, it is a good choice to select *XGBoost* and *LightGBM* to complete this task. Before talking about some details related to these frameworks, some introductions about Ensemble Learning and Gradient Boosting are necessary to be described.

3.4.1 Ensemble Learning

In the machine learning supervised learning algorithm, our goal is to learn a stable model with good performance in all aspects, but the actual situation is often not so ideal, sometimes we can only get multiple models with preferences (weakly supervised model, better performance in some aspects). Ensemble Learning is to combine multiple weak-supervised models here in order to obtain a better and more comprehensive strong-supervised model. The underlying idea of Ensemble Learning is that even if a weak classifier gets a wrong prediction, other weak classifiers can also correct the error.

There are three common Ensemble Learning frameworks: Bagging, Boosting, and Stacking:

- Bagging [29] is the full name of Bootstrap Aggregating; Each base learner will put back sampling on the training set to get sub-training sets. Each base learner is trained on the basis of different subtraining sets, and the final prediction result is obtained by aggregating all the predicted values of the base learner. Bagging's common aggregate method is the voting method, where the categories with the most votes are the prediction categories. Bagging is used as a way to reduce the variance of a base estimator (e.g., a decision tree) by introducing randomization into its construction procedure and then making an ensemble out of it.

- The Boosting Training Process is stepwise. The training of the base model is in order; each base model will learn on the basis of the previous base model, and they will synthesize all the predicted values of the base model to produce the final predicted result, with weighting being the more common comprehensive method. Boosting is a machine learning algorithm that can reduce the deviation in supervised learning. Representatives in boost are the AdaBoost [30] (adaptive boost) algorithm and the Gradient boost algorithm [31] [32].

- Stacking is to use all the data to train the base model first, and each base model predicts each training sample, and its predicted value will be used as the feature value of the training sample, and finally, a new training sample will be obtained. Based on the new training samples, the training gets the model and the final prediction result.

3.4.2 Gradient Boosting

As we mentioned in Ensemble Learning, Gradient Boosting is one of the Boosting methods of Ensemble Learning. The basic idea of Gradient Boosting is to generate multiple weak learners

sequentially. The goal of each weak learner is to fit the negative gradient of the loss function of the previous accumulated model on the basis of using MSE as the loss function, so that the cumulative loss of the model will decrease in the direction of the negative gradient after adding the weak learner. Theoretically, Gradient Boosted Machine can choose a variety of different learning algorithms as the base learner. In reality, the most commonly used base learner is the decision tree. The algorithm first uses an initial value to learn a decision tree and then obtains the predicted value and the predicted residual on the leaves. The subsequent decision tree is continuously fitted based on the residual of the previous decision tree, to train a series of decision trees as models.

Gradient-boosted decision trees (GBDT) can be considered as an additive model of K trees:

$$\hat{y_i} = \sum_{k=1}^{K} f_k(x_i), f_k \in F,$$

where

- $\hat{y_i}$ is predicted value;
- F is the function space of all trees/estimators;
- f_k is one of the trees / estimators generated per stage.

Consider GBDT has M stages in total, for stage $m(1 \leq m \leq M)$,

$$F_m(x_i) = F_{m-1}(x_i) + h_m(x_i) = y_i,$$

where y_i is observed value. So here $h_m(x_i)$ is the residual, which is

$$h_m(x_i) = y_i - F_{m-1}(x_i).$$

In the next stage, the goal is to fit residual $h_m(x_i)$, and choose a loss function that is defined as $Loss(y_i, F_m(x_i))$. If we want to minimize the loss of the model after adding the weak learner, $h_m(x_i)$, of the m stage, the newly added model should move along the direction of negative gradient according to the gradient descent method. That is, if the weak learner of the m stage fits the negative gradient of the loss function about the cumulative model, the loss of the cu-

mulative model after adding the weak learner will be minimal. Therefore, the

$$F_m(x) = F_{m-1}(x) + \arg\min_{h \in H} Loss(y_i, F_{m-1}(x_i) + h(x_i)).$$

The target value of weak learning training in the m-th round is the negative gradient of the loss function:

$$g_m = -\frac{\partial Loss(y_i, F_{m-1}(x))}{\partial F_{m-1}(x)}.$$

For the loss function, it is common to select Mean Square Error(MSE), $Loss_{MSE} = (y - F_{m-1}(x))^2$. So its gradient is

$$g_m = -\frac{\partial Loss_{MSE}}{\partial F_{m-1}(x)} = 2(y - F_{m-1}(x)) = 2h_m(x).$$

In this situation, the negative gradient of the loss function turns out to be just the residual. Therefore, gradient boosting could be specialized to a gradient descent algorithm and be generalized by entailing "plugging in" a different loss and its gradient.

3.4.3 XGBoost and LightGBM

Both *XGBoost* [6] and *LightGBM* [7] are boosting the ensemble learning tree algorithms, which implement the GBDT algorithm efficiently and make an improvement in computing large data sets. *XGBoost* is developed as a highly scalable end-to-end tree booster system, which also supports parallel tree learning. Unlike GBDT, *XGBoost* adds a regularization term to the objective function to prevent overfitting.

The objective function to optimize can be defined as below:

$$obj = \sum_{i=1}^{n} Loss(y_i, \hat{y_i}^{(t)}) + \sum_{i=1}^{t} \omega(f_i).$$

$\omega(f_i)$ is the complexity of the tree:

$$\omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2,$$

where γ represents the complexity of the leaves, T indicates the number of leaves, λ is the

penalty parameter, and w_j is the output result of each leaf node.

The training process is similar to GDBT since they are both additive model:

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \omega(f_i) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \omega(f_t).$$

The key point of *XGBoost* is to take the Taylor expansion of the loss function up to the second order to quickly optimize the object function. So we can get:

$$obj^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \omega(f_t),$$

where the g_i and h_i are defined as

$$g_i = \partial_{\hat{y}_i^{(t-1)}} \, l(y_i, \hat{y}_i^{(t-1)})$$

$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} \, l(y_i, \hat{y}_i^{(t-1)}).$$

According to the above definition, the value of the objective function depends only on g_i and h_i, because other terms are constant. *XGBoost* uses first-order and second-order partial derivatives, and the second derivative is conducive to a faster and more accurate gradient descent. Using the Taylor expansion to obtain the function as the second derivative of the independent variable, the leaf splitting optimization calculation can be carried out only depending on the value of the input data without selecting the specific form of the loss function. In essence, the selection of the loss function is separated from the optimization of the model algorithm/parameter selection. This decoupling increases the applicability of *XGBoost*, allowing it to select loss functions on demand, for classification or regression.

In general, the main differences between GBDT and *XGBoost* are:

- GBDT is a popular machine learning algorithm, while *XGBoost* is an effective engineering implementation of GBDT.
- When using the decision tree as the base classifier, *XGBoost* adds the regularization term to control the complexity of the model, which is conducive to preventing overfitting and improving the generalization ability of the model.
- GBDT only uses the first derivative information of the loss function in model training, and *XGBoost* performs the second order Taylor expansion of the loss function, which can use the first and second derivatives at the same time.

In addition, other important features of XGBoost can be listed below.

- This model implements multi-CPU core training.
- *XGBoost* can detect and learn from nonlinear data patterns.
- Out-of-core computing when working with datasets that do not fit into memory.
- Making the best use of hardware with cache optimization.

LightGBM is similar to *XGBoost*, but it makes some effective optimization in computation speed, memory usage, and accuracy. These optimizations are mainly due to histogram-based algorithms, which are used to select and split indicators, and leafwise (best-first) tree growth.

Unlike most boost strategies using presort-based algorithms for decision tree learning, *LightGBM* uses histogram-based algorithm that buckets continuous feature (attribute) values into discrete bins to speed up training by:

- reducing the cost of calculating the gain for each split.
- reducing memory usage.
- reducing the communication cost for distributed learning.
- using histogram subtraction for further acceleration.

Another difference between them is the decision tree growth method. *XGBoost* exploits level-wise tree growth(vertically), but *LightGBM* applies the growth of the leaf-wise tree (horizontally). Figure 3.7 and Figure 3.8 show the difference between these two methods.

XGBoost adopts the level-wise growth strategy to facilitate parallel calculation of split nodes of each layer, which improves the training speed, but also increases a lot of unnecessary splits due to too small node gain. In contrast, leaf-wise growth strategy only splits the leaf which has the largest information gain on the same layer with the maximum depth limitation. In this way, it is more effective and highly efficient.

In summary, Table 3.2 shows the comparison between *XGBoost* and *LightGBM*.

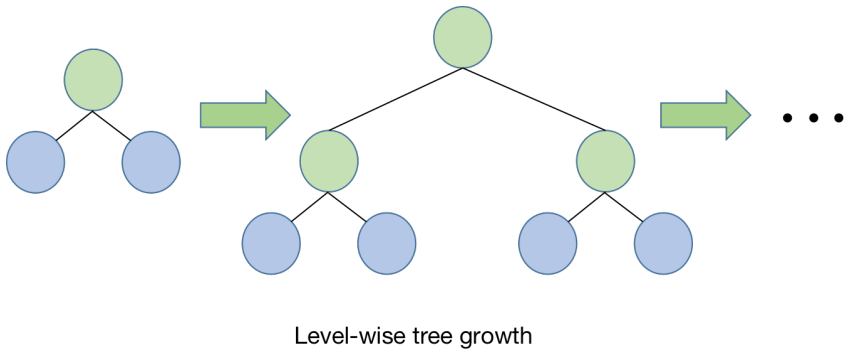

Figure 3.7: Level-wise tree growth.

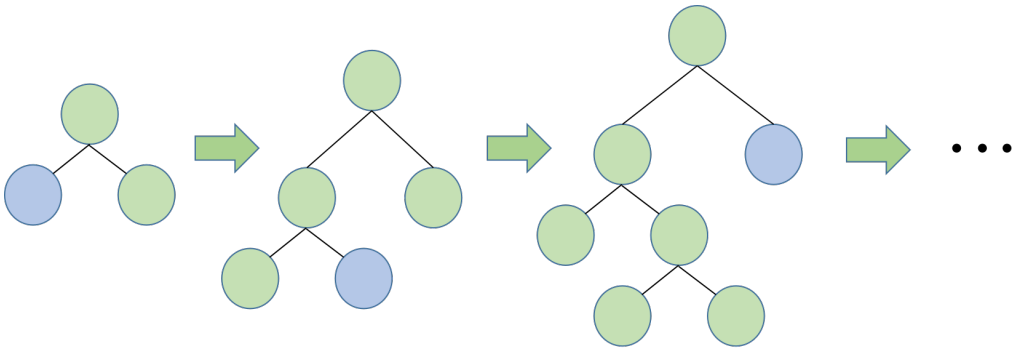

Figure 3.8: Leaf-wise tree growth.

3.5 Evaluation

The accuracy, precision, recall and F1-score are widely used for estimating the performance of models in classification problems. Accuracy measures the proportion of samples that are correctly predicted, but it only focuses on whether the final result is correctly predicted, which is easily affected by data imbalance. Precision is the ability to accurately predict samples, while recall implies the ability to correctly predict as many samples as possible. F1 score can be regarded as a harmonic average of precision and recall. Moreover, the confusion matrix is also useful for evaluating the performance of prediction. It is a contingency table with four(or more than four) different combinations of predicted and actual values.

Consider a confusion matrix as Table 3.3 :

	XGBoost	*LightGBM*
Tree growth strategy	Level-wise tree growth	Leaf-wise tree growth
Splitting method	pre-sort-based algorithms	Gradient-based One-Side Sampling (GOSS)
Handling categorical features	Yes	Yes
Handling missing values	Yes	Yes

Table 3.2: XGBoost vs LightGBM

		Actual values	
		Positive	Negative
Predict values	Positive	TP	FP
	Negative	FN	TN

Table 3.3: The example of confusion matrix.

Therefore, accuracy can be defined as below:

$$accuracy = \frac{TP + TN}{P + N}.$$

Precision can be defined as below:

$$precision = \frac{TP}{TP + FP}.$$

Recall can be defined as below:

$$recall = \frac{TP}{TP + FN}.$$

F1 score can be defined as below:

$$F1\ score = 2 \times \frac{precision \times recall}{precision + recall}.$$

In addition, to prevent the impact of data imbalance, balanced accuracy in *Scikit-learn* is also used, which is defined as the average recall obtained in each class. Using precision, recall, and F1 score, we can measure the prediction effect of the classifier for different classes from a variety of different perspectives, so as to better evaluate the performance of the classifier.

Furthermore, to better use the data and measure the accuracy of the model, the data set will be divided into the training set and the verification set. Finally, the balanced accuracy of the

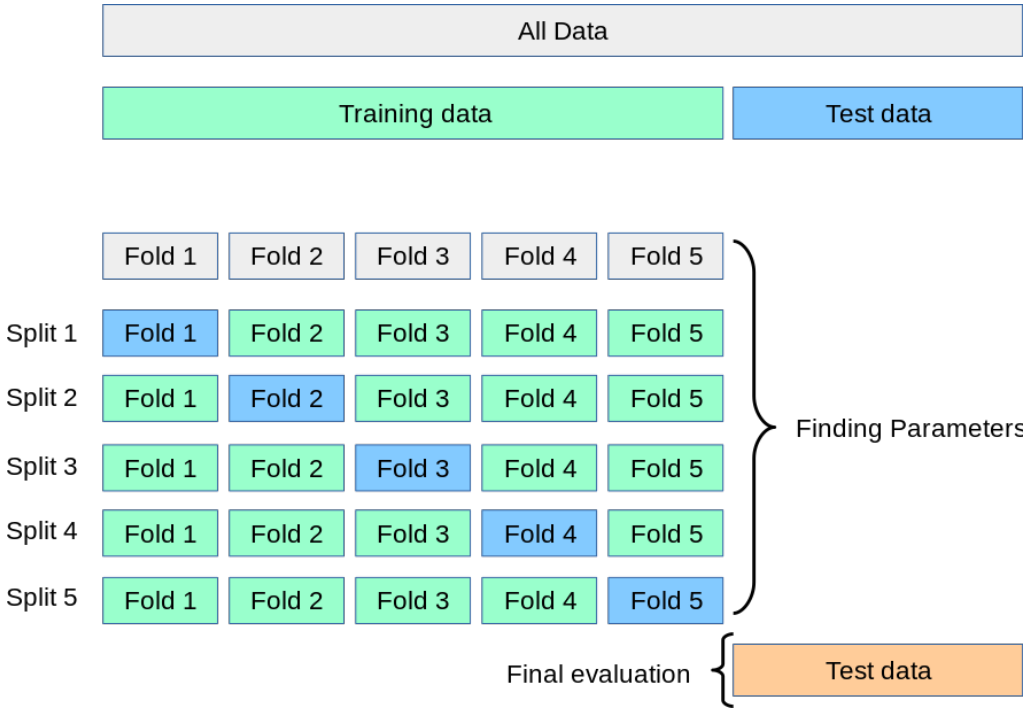

Figure 3.9: An example of 5-fold cross-validation.

5-fold cross-validation in the training set and the balanced-accuracy of verification set will be compared. The cross-validation process is shown in Figure 3.9.

4

Experiments and results

In this chapter, I report on the experiment process and results, and I discuss and analyze the results. The experiment was divided into two parts, based on two different data, one using the old five labels and the other combining the old labels to create five new labels. The first section is about overall results on five old labels dataset; the second section is related to overall results on five new labels dataset.

First, the software tools and modules used in the experiment are shown in Table 4.1.

Figure 4.1 illustrates the general procedure of the experiment.

In these two experiments, the same parameters of *LightGBM* and *XGBoost* are shown as Table 4.2, other parameters are default.

Programming Language	*Python*
Machine Learning Module	*Scikit-learn*
Chinese Word Segmentation Module	*Jieba*
Classifiers	*XGBoost, LightGBM*
Imbalance-learning Module	*Imblearn*
Word Embedding Module	*Gensim*

Table 4.1: Tools and modules in this experiment.

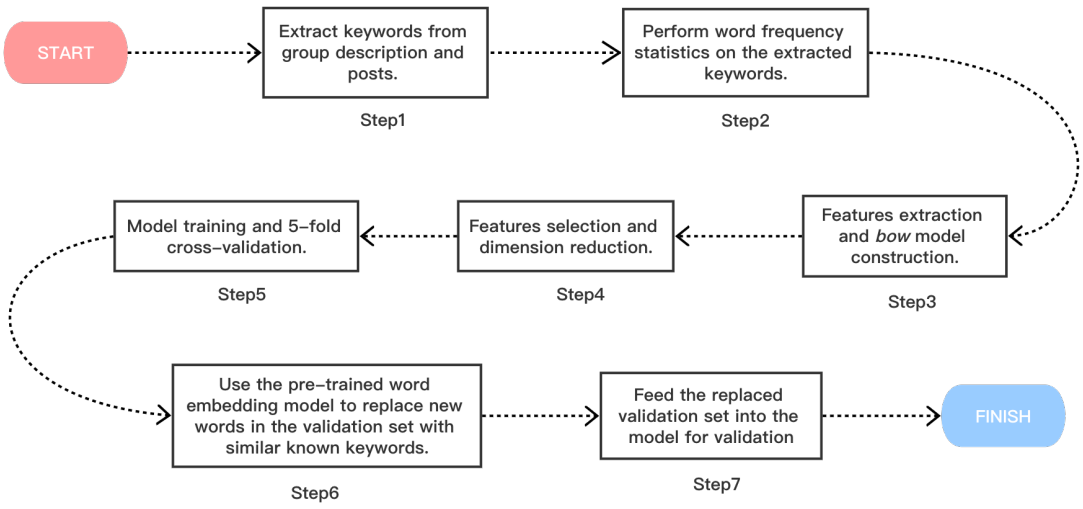

Figure 4.1: Process of model building.

n_estimators	150
max_depth	20
early_stopping_rounds	10

Table 4.2: Parameters of *LightGBM* and *XGBoost* used in the experiments.

4.1 Overall results on 5 old labels dataset

At the beginning of this work, to efficiently verify the feasibility of machine learning in the text classification of the Zhishixingqiu platform, I draw some samples from five classes to construct the dataset. This dataset has 77485 rows and 6 columns. Figure 4.2 and Figure 4.3 are the histograms of the statistics of the dataset used in this dissertation.

Figure 4.4 shows the top 20 keywords for each group in this dataset. According to the figure, we can clearly see the most important keyword of each group. For example, the keywords of Finance and Economy groups from top 1 to top 3 are " 投资"(investment), " 交流"(communication) and " 学习"(learning) respectively. For Information Security groups, the top 1 to top 3 keywords are " 安全"(security), " 技术"(technology) and " 测试"(test) respectively.

To compare different models, I use *LightGBM* and *XGBoost* without dimension reduction on the BoW model as the baseline, and perform different ways of learning imbalance and se-

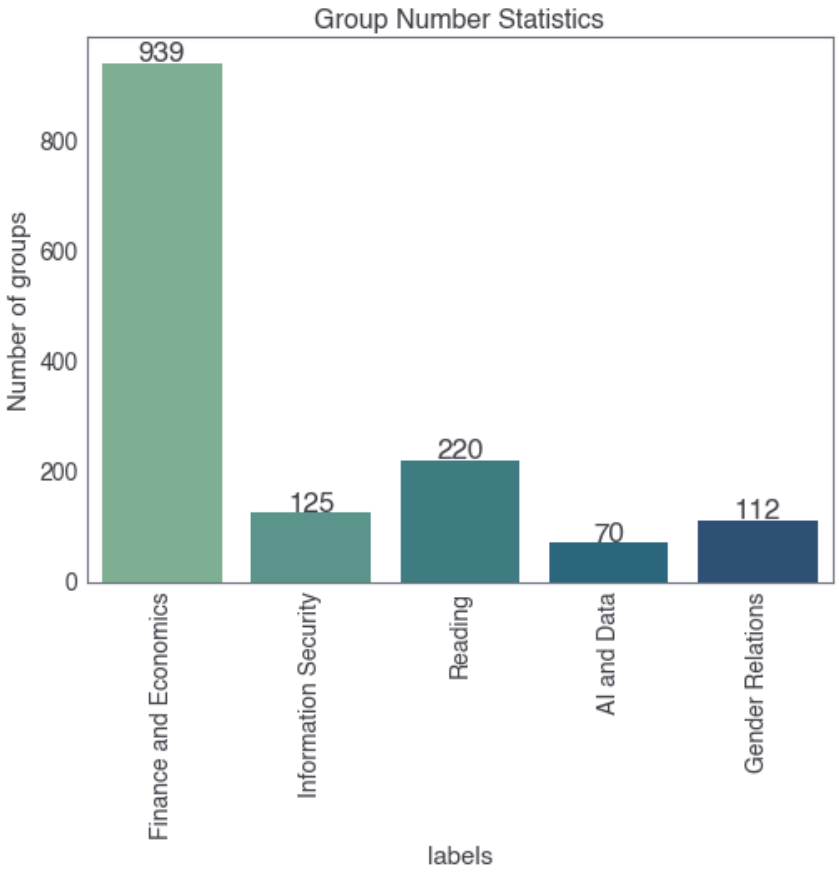

Figure 4.2: Five Groups Number Statistics.

lection of features / dimension reduction on the baseline model, respectively. The results are illustrated in Table 4.3.

According to the Table 4.3 of the results, we can learn that both *LightGBM* and *XGBoost* achieve great performance on this dataset, and their balanced accuracy is over 93%. By using imbalance learning, the mean balanced accuracy of the 5-fold cross-validation of *LightGBM* and *XGBoost* is higher than baseline, and the highest is 99.44% of *XGBoost*. However, the balanced accuracy on the verification set is worse than the baseline of *LightGBM* and *XGBoost*.

For feature selection and dimension reduction in this experiment, I perform word frequency statistics in the BoW model as Figure 4.5 shows, delete words with word frequency less than 5, and acquire 13605 features. Then I use Chi-square detection to select most important 3000 features out of 13605 features. It is interesting to learn that the performances of *LightGBM* and *XGBoost* after feature selection / dimension reduction are almost the same as baseline's.

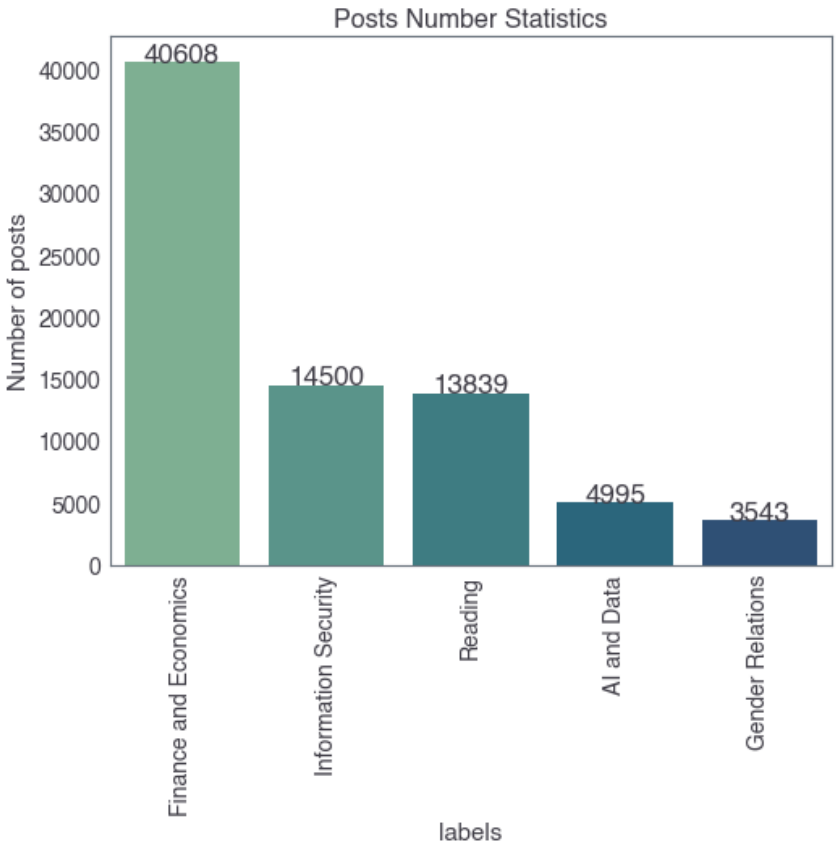

Figure 4.3: Five Groups' Post Number Statistics.

For *LightGBM*, it even achieves a slightly higher balanced accuracy than the baselines on the verification set. This implies that most keywords in the original BoW model are not important. On the contrary, due to the use of too many unimportant words, the sparsity of the BoW model is particularly large and takes up a lot of computing resources. Therefore, it is significant to try feature selection and dimension reduction when using machine learning with the BoW model in practice.

At the end, PCA is used to perform linear dimensionality reduction with 1000 components, 800 components, and 500 components on *LightGBM*. The result is illustrated in Table 4.4. It can be seen that the use of PCA can achieve a better result in 5-fold cross-validation, but a little worse in the verification set. The advantage is that we can exploit fewer dimensions to achieve similar performance to baseline, which is helpful for representing data and reducing the usage of computing resources. However, as we mentioned before, it is a challenge to explain the new features of PCA, especially in a commercial environment.

	LightGBM		XGBoost	
	5-fold cross-validation	Verification	5-fold cross-validation	Verification
Baseline	95.98%	95.59%	97.08%	95.74%
Random Oversampling	99.10%	95.31%	99.44%	95.47%
ADASYN Oversampling	98.48%	94.85%	98.92%	95.66%
SMOTE Oversampling	98.96%	95.53%	99.18%	95.50%
Random Undersampling	96.88%	93.79%	97.32%	94.56%
Class Weights	98.03%	93.70%	97.08%	95.74%
13605 Features	95.98%	95.59%	97.10%	95.48%
3000 Features	95.70%	95.67%	97.53%	93.79%
Replacement of similar words	95.70%	95.58%	97.53%	95.73%

Table 4.3: Results of the experiment: The metric used in the experiment is balanced accuracy. The baseline is based on the BoW model using 73241 features without any processes. Imbalance learning uses different methods with 73241 features. 13605 features are selected by removing keywords whose frequency k <= 5. The 3000 features are selected by Chi-square detection. The model with replacement of similar words is close to baseline except using Word2Vec model to perform replacement of similar words on the verification set. Besides, the training set and verification set have 67485 rows and 10000 rows respectively.

LightGBM With PCA

n_components	1000	800	500
5-fold cross-validation	96.44%	96.44%	96.20%
Verification	94.21%	94.32%	94.13%

Table 4.4: Results of the experiment with PCA: The metric used in the experiment is also balanced accuracy.

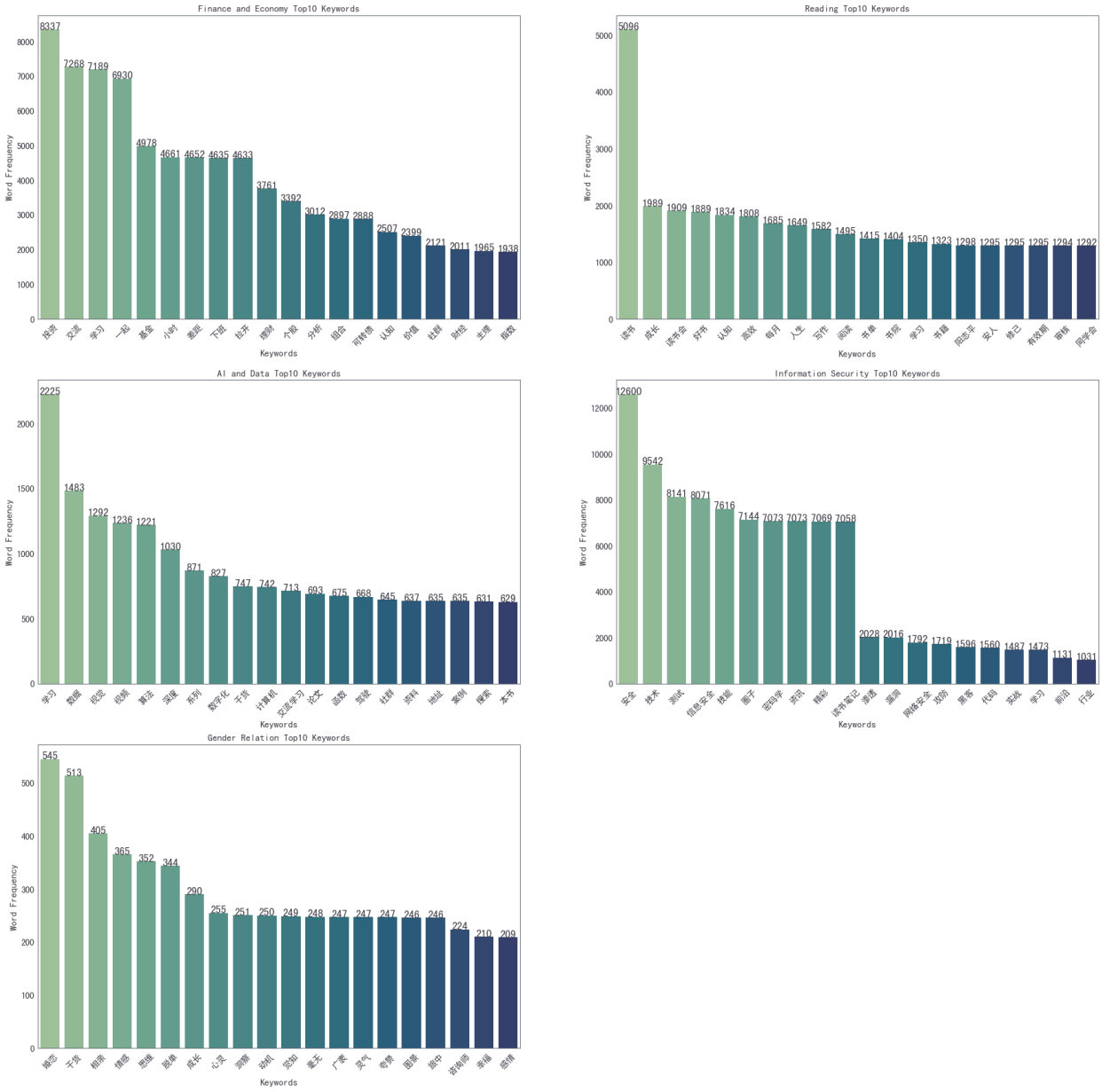

Figure 4.4: Top 20 Keywords for 5 Groups.

For the replacement of similar words by the Word2Vec model, I only load 2 million word vectors of the pre-training Word2Vec model to replace the unseen words from verification set with the most similar word in BoW model. Although this approach decreases the sparsity of the BoW model, the results of the model on the verification set are reduced by 0.01%. The BoW model will lose information when input new words that are not in the BoW model's features,

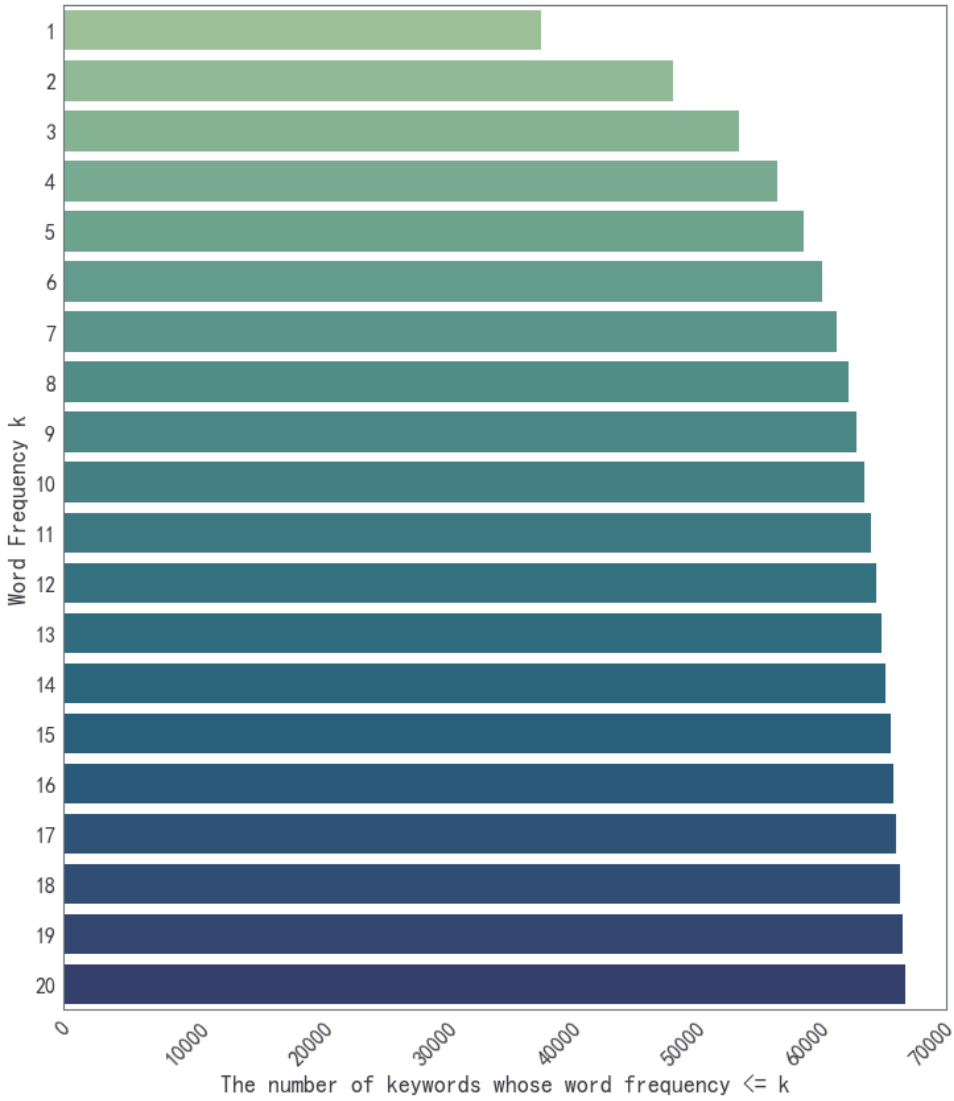

Figure 4.5: More than 80% of the keywords frequency is less than 5 in 5 old labels dataset.

and the replacement of similar words can make up for this information loss to a certain extent. However, some words may be replaced incorrectly, leading to other bias. Therefore, replacing similar words is a double-edged sword in practice.

In general, the performances between *LightGBM* and *XGBoost* are very similar. In most cases, the balanced accuracy of *XGBoost* is slightly higher than that of *LightGBM*, whether it is the average balanced accuracy of cross-validation or the balanced accuracy of the validation set. However, *LightGBM* runs faster than *XGBoost*, and outperforms *XGBoost* in some cases. Therefore, in the other experiment, it is a great choice to use *LightGBM*.

4.2 Overall results on 5 new labels dataset

According to the business of Zhishixingqiu, it is beneficial to decrease the labels by merging several similar child labels into a new parent label because some child labels are in the same field. Another reason for creating a new label system is that some groups assigned original labels are very rare, as Figure 3.1 shows, leading to a serious problem of data imbalance. The new label system is shown in Figure 4.6.

Figure 4.6: New Labels System by merging child labels.

The experiment steps are similar to those in the last section. For this dataset, I collect almost 470000 rows of data with assigning new labels, and make sure this is not a serious imbalance

dataset. In addition, I filter out posts with fewer than 200 Chinese characters to make sure each post has enough information for modeling. Also, I abandon the group description and use *Jieba* to extract 20 keywords from the post content. The dataset is split into training set with 379007 rows and verification set with 94000 rows. Figure 4.7 is the statistic histogram of this new dataset.

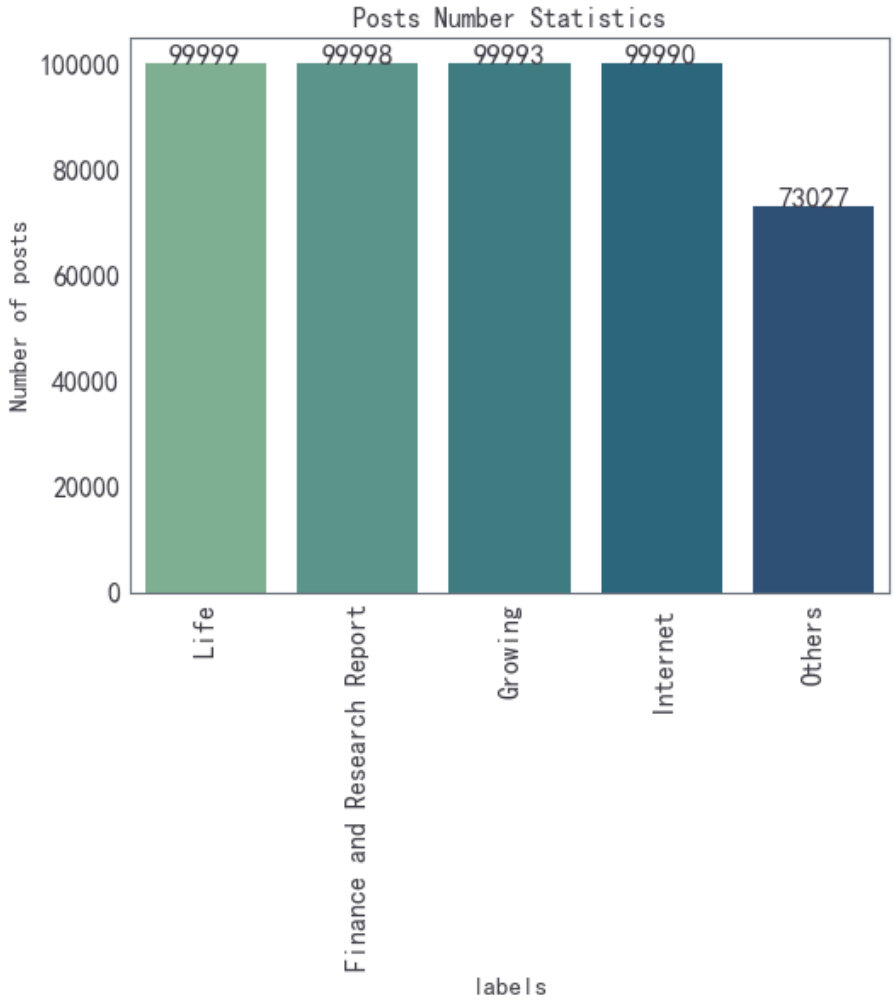

Figure 4.7: Five New Groups' Post Statistics.

Figure 4.8 shows the top 20 keywords for each group in this dataset. According to the figure, we can learn the most important keyword of each group with new labels. The top 1 keyword for each group can be listed respectively as below:

- Finance and Research Report groups: " 公司 "(company)

	Precision	Recall	F1 score	support
Finance and Research Report	0.9	0.87	0.88	2159
Internet	0.71	0.68	0.70	2085
Growing	0.61	0.60	0.60	2056
Life	0.77	0.75	0.76	2141
Others	0.53	0.62	0.58	1559
accuracy			0.71	10000
macro avg	0.70	0.70	0.70	10000
weighted avg	0.72	0.71	0.71	10000

Table 4.5: Classification report of *LightGBM* with SMOTE oversampling and replacement of similar words.

- Internet groups: " 产品"(product)
- Growing groups: " 学习"(learning)
- Life groups: " 孩子"(child)
- Others groups: " 客户 "(client)

For this dataset, I remove the keywords whose frequency is less than 10, and then use Chi-square detection to get the BoW model with 30000 features. In addition, I also apply SMOTE oversampling to generate a balanced dataset and perform the replacement of a similar word by the Word2Vec model. Finally, I get 72.01% balanced accuracy of 5-fold cross-validation and 70.44% balanced accuracy of verification set, which is a good result. The classification report and confusion matrix are shown in Table 4.5 and Figure 4.9.

From Table 4.5, it can be seen that this classifier is good at predicting the Finance and Research Report, with the highest precision(0.9), recall(0.87), and F1 score(0.88) among all 5 new labels. The predictions on Internet and Life are similar to each other with more than 0.7 F1 score. In addition, the predictions of Growing and Others are the worst with 0.6 of F1 score and 0.58 F1 score, respectively.

For Zhishixingqiu, we learn that the majority of the groups and posts are related to finance and economy; thus the most important job is to make sure that we classify Finance and Research Report correctly. According to the results, the BoW model building, imbalance learning, application of *LightGBM* and *XGBoost* and replacement of similar words are reasonable and helpful to complete text classification. As for community group classification, after classifying all the posts / articles in the group, we can use the proportion of posts / articles of different classes to define which class the group belongs to.

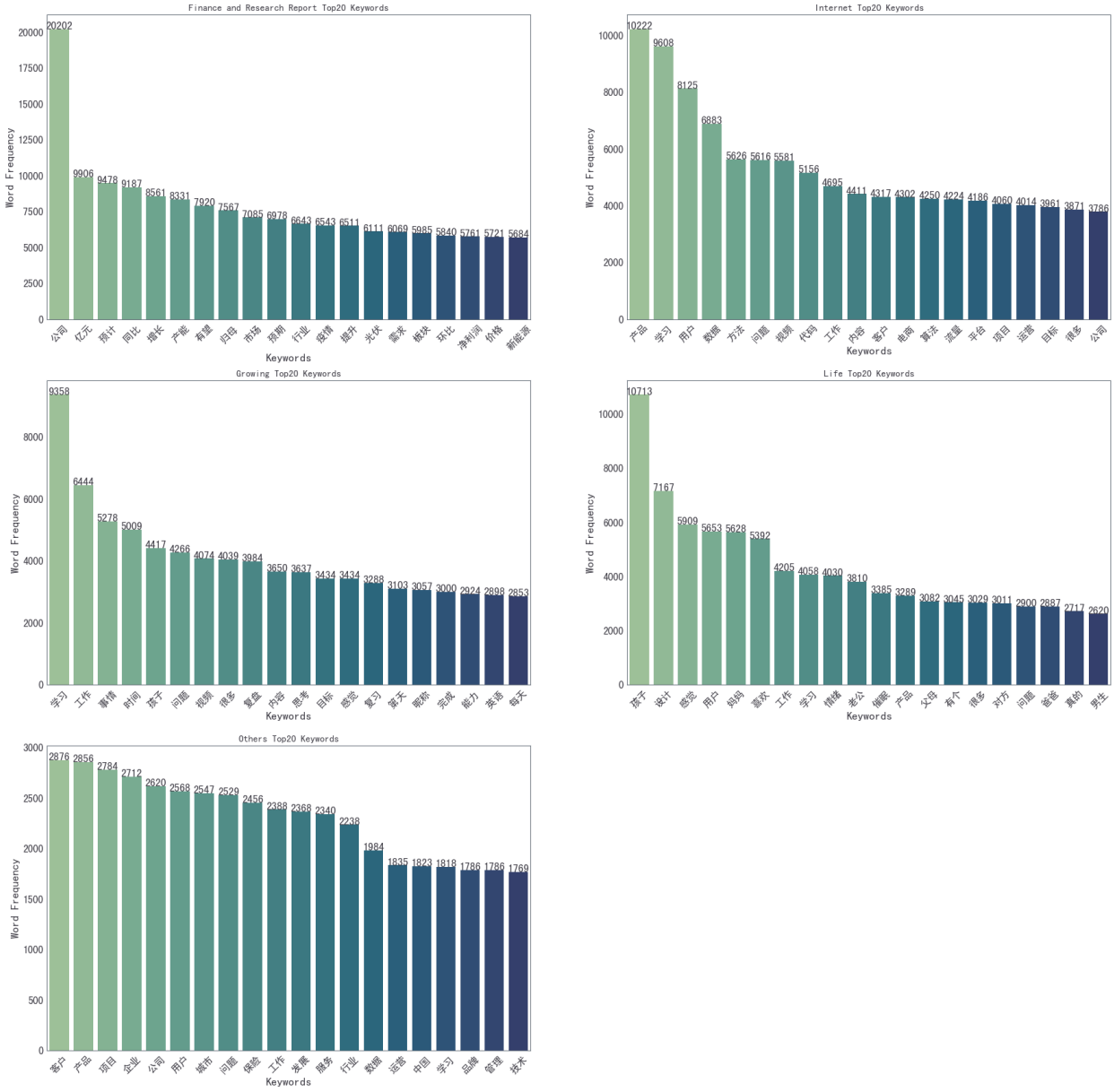

Figure 4.8: Top 20 Keywords for new 5 Groups.

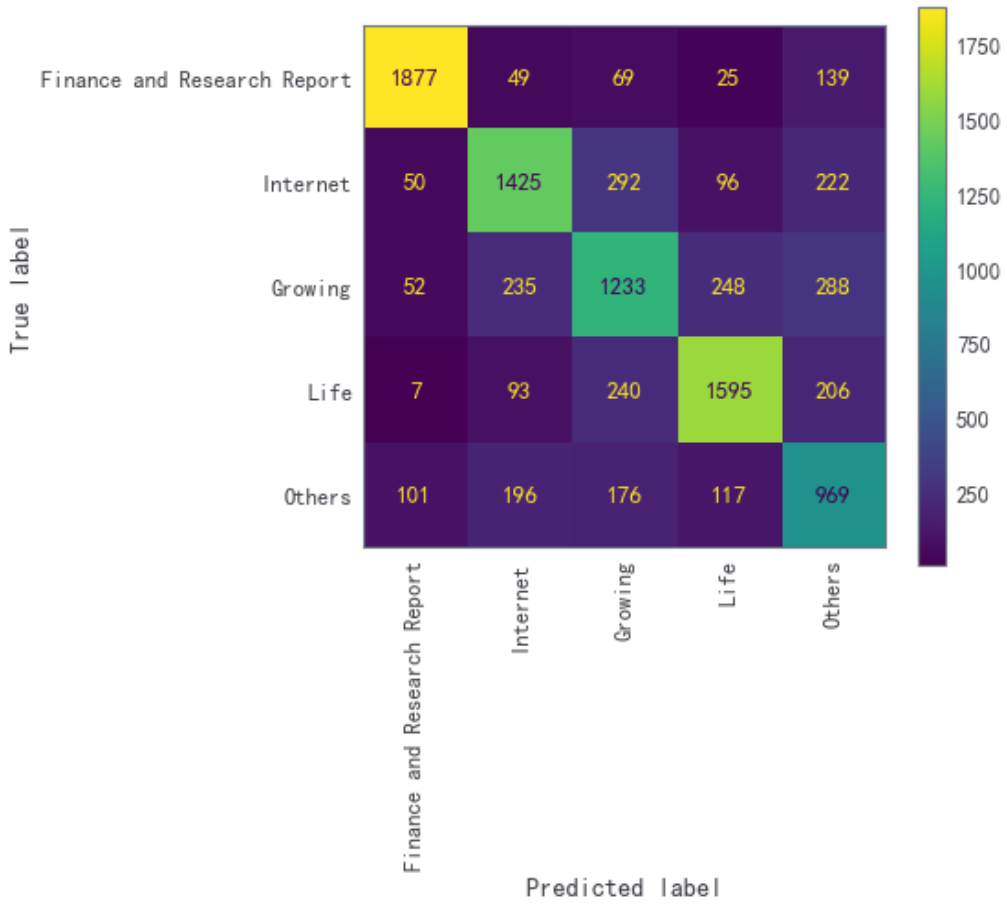

Figure 4.9: Confusion matrix of *LightGBM* with SMOTE oversampling and replacement of similar words.

5 Conclusion

To achieve Chinese text classification in this work, the bag-of-words model is used to represent the text, and machine learning is applied to achieve classification. I complete the process from data set construction to modeling classification and show that this approach is feasible and effective in practice. In addition to effectively completing the classification task using machine learning with the bag-of-words model, we can also clearly understand which keywords are important in the modeling process, and collecting these important keywords is of great significance for the platform. However, choosing keywords with more precision remains a challenging task, as some of the selected keywords will still be meaningless, while some meaningful and important keywords will be missed. Moreover, replacement of similar words can reduce the sparsity of the BoW model and make full use of the information when encountering new words, but we should be careful when replacing similar words to avoid introducing new deviations from replacement errors in practice. Although a general pre-training Word2Vec model is effective in complete replacement of similar words, it may not totally adapt to the real situation of Zhishixingqiu due to its community having some special keywords or context which can not be represented in the pre-training Word2Vec model. For modeling, *LightGBM* and *XGBoost* are excellent algorithms for performing the classification task in this work, and all achieve great performance on the datasets with 5 old labels, but there is still room for improvement on the 5 new label datasets without group description. In general, what I do in this work is try to better adapt to the business and make classification modeling more interpretable.

In the future, I will make continuous improvements based on the original model, use more

historical text data to train the model, improve its accuracy and stability on the new label dataset, and successfully apply the model in practice to support commercial development. Furthermore, to correctly replace similar words in the Zhishixingqiu community, it is also worth building a corpus and training a Word2Vec model based on a specific community. This enables the model to better adapt to the context of the platform and thus achieve better performance in practice.

References

[1] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[3] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[4] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[5] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

[6] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[7] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[8] T. Verma, R. Renu, and D. Gaur, "Tokenization and filtering process in rapidminer," *International Journal of Applied Information Systems*, vol. 7, no. 2, pp. 16–18, 2014.

[9] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[10] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[11] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[12] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *the Journal of machine Learning research*, vol. 9, pp. 1871–1874, 2008.

[14] S. Jiang, G. Pang, M. Wu, and L. Kuang, "An improved k-nearest-neighbor algorithm for text categorization," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1503–1509, 2012.

[15] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[17] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[18] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[19] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[20] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data," *Data mining and knowledge discovery*, vol. 28, no. 1, pp. 92–122, 2014.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[22] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE, 2008, pp. 1322–1328.

[23] Z. Wei, D. Miao, J.-H. Chauchat, and C. Zhong, "Feature selection on chinese text classification using character n-grams," in *International conference on rough sets and knowledge technology*. Springer, 2008, pp. 500–507.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

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

[26] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[27] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[28] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[29] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[30] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[31] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[32] ——, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

Acknowledgments

Helpful discussions and suggestions by my academic supervisor Professor Alberto Testolin and the company Zhishxingqiu are indispensable and gratefully acknowledged. This work could not have been completed without their help. In addition, I also would like to express my sincere thanks to my family and my girlfriend Ding who strongly support me during my master study.