國立交通大學

資訊科學與工程研究所

碩士論文

從 CopeOpi 純量擴充至 CopeOpi 向量：用於多類別本文分類的詞向量

From CopeOpi Scores to CopeOpi Vectors: Word Vectors for Multiclass Text Classification

研 究 生：蔡佩珊

指導教授：陳穎平 博士

中 華 民 國 一 百 零 六 年 九 月

從 CopeOpi 純量擴充至 CopeOpi 向量：
用於多類別本文分類的詞向量

From CopeOpi Scores to CopeOpi Vectors:
Word Vectors for Multiclass Text Classification

研 究 生：蔡佩珊　　Student: Pei-Shan Tsai

指導教授：陳穎平　　Advisor: Ying-Ping Chen

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial fulfilment of the requirements

for the Degree of

Master

in

Computer Science

September 2017

Hsinchu, Taiwan, Republic of China

中 華 民 國 一 百 零 六 年 九 月

從 CopeOpi 純量擴充至 CopeOpi 向量：用於多類別本文分類的詞向量

學生：蔡佩珊　　指導教授：陳穎平 博士

國立交通大學 資訊科學與工程研究所

摘　　要

在這資訊爆炸的時代，每天有大量的數位本文被產出。為了從這些資料中獲取有用的訊息，文字探勘成了當前的熱門議題，而本文分類便是其中的重要任務之一。

在本論文中，我們提出一個用於多類別本文分類的向量空間模型，詞向量 CopeOpi vectors。我們將用於中文情感分析的 CopeOpi scores，擴充至能夠用於多類別本文分類且無語言限制的 CopeOpi vectors。

我們測試 CopeOpi vectors 於英文及中文的情感分析及主題分類問題，並與幾個常用於本文分類的特徵向量進行比較，將這些特徵向量套用至不同的機器學習演算法。實驗結果顯示 CopeOpi vectors 能夠用更小的向量長度與更短的訓練時間，達到與其他特徵向量同樣水平的分類成果。CopeOpi vectors 是適用於多類別本文分類，兼具效果與效率的詞向量。

關鍵字：本文分類；向量空間模型；詞向量

From CopeOpi Scores to CopeOpi Vectors: Word Vectors for Multiclass Text Classification

Student: Pei-Shan Tsai Advisor: Dr. Ying-Ping Chen

Institute of Computer Science and Engineering
National Chiao Tung University

Abstract

In the era of technology, millions of digital texts are generated every day. To derive useful information from these textual data, text mining has become a popular area of both research and business. One of the most important task of text mining is text classification.

In this thesis, we propose a vector space model for multiclass text classification, the word vectors—CopeOpi vectors. We expand CopeOpi scores which are used in Chinese sentiment analysis, to CopeOpi vectors which can be used in multiclass text classification without the language limit.

We verify the functionality of CopeOpi vectors by a series of text classification problems, including sentiment analysis and topic categorization, in both English and Chinese. We make comparisons with several commonly-used features for text classification, and examine these features on different types of machine learning algorithms. The results show that CopeOpi vectors can produce comparable results with a smaller vector size and shorter training time. CopeOpi vectors are effective and efficient features for multiclass text classification.

Keywords: Text classification; Vector space model; Word vector

Acknowledgement

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the era of technology, millions of digital texts such as emails, social media posts, product reviews, news articles and websites are generated every day. To derive useful information from these textual data, text mining has become a popular area of both research and business. One of the most important task of text mining is text classification.

1.1 The Text Classification Problems

Text classification, or text categorization, is a task of assigning a document[1] to a set of predefined classes, categories, or labels.

As with many other classification tasks, the text classification problems have traditionally been solved manually, or by knowledge engineering approaches with hand-crafted classification rules. However, both methods are expensive to scale due to the needs of skilled labors and expert knowledge. Therefore, to deal with a large number of documents and a great diversity of contents, most recent works of text classificatioin focus on machine learning approaches, which require only a set of labeled training instances which costs less human efforts[1, 2].

1.1.1 Definition

In a text classification problem, we are given a document space $\mathbb{X}$ and a set of predefined classes $\mathbb{C}$. The task of text classification can be defined as an unknown assignment function.

$$f : \mathbb{X} \times \mathbb{C} \to \{\texttt{True}, \texttt{False}\}$$

which assigns each pair $\langle d, c \rangle \in \mathbb{X} \times \mathbb{C}$ a Boolean value `True` if the document d is in the class c and `False` otherwise.

[1] We use the term *document* in general sense as textual data.

Machine Learning for Text Classification

By using a machine learning algorithm Γ with a labeled training set $\mathbb{D} = \{\langle d, c\rangle \mid \langle d, c\rangle \in \mathbb{X} \times \mathbb{C}\}$, we wish to learn a classier, or classification function γ which approximates the unknown assignment function f as close as possible[3].

$$\Gamma(\mathbb{D}) = \gamma$$

$$\gamma : \mathbb{X} \times \mathbb{C} \to \{\texttt{True}, \texttt{False}\} \approx f$$

This type of machine learning is a form of supervised learning since a labeled training set serves as a supervisor directing the learning process.

1.1.2 Applications

Typically, the document space $\mathbb{X}$ can be any kinds of texts and the classes $\mathbb{C}$ are defined for the user needs, therefore text classification has a wide variety of applications in text mining[4].

Document Organization and Information Retrieval

When the documents are news articles, scientific literature, blog collections, etc., users may want these documents to be grouped for different topics. A hierarchical catalogue is especially useful for searching and retrieval.

Sentiment Analysis and Opinion Mining

When the documents are social media posts and customer reviews, users may want these comments to be identified for opinioned or non-opinioned, positive or negative. The mined opinions are helpful for business marketing.

Email Routing and Spam Filtering

When the documents are emails, users may want these messages to be routed for different subjects, be sorted for different priorities, or be filtered for spam or ham. It is more convenient to check out emails in classified folders than in a messy inbox.

1.2 Thesis Outlines

In this chapter, we have been talking about the motivation of this thesis, i.e., the text classification problems. The remaining chapters are organized as follows:

- Chapter 2: Related Works.

  We review the techniques for text classification, including vector space models and machine learning algorithms.

- Chapter 3: Methodology: CopeOpi Vectors.

  We first introduce CopeOpi scores, then propose augmented CopeOpi scores and CopeOpi vectors step by step.

- Chapter 4: Experiments and Results.

  We verify the functionality of CopeOpi vectors by a series of text classification problems, including sentiment analysis and topic categorization, in both English and Chinese.

- Chapter 5: Conclusions and Future Works.

  We make conclusions about this thesis and point out some future directions of research about CopeOpi vectors.

Chapter 2

Related Works

2.1 Vector Space Models

In order to teach machines to understand words, phrases, sentences, paragraphs and documents, we need to design a representation which they can manipulate. Based on a series of statistical semantics hypotheses, vector space models take event frequencies in a corpus as clues to discover latent semantic, and represent texts as vectors in a vector space for the use of algebraic operations. They derive vectors from a frequency matrix, and the structure of the frequency matrix relates to their scope of application[5].

2.1.1 Similarity of Documents: The Term-document Matrix

Hypothesis 2.1.1: Bag-of-words Hypothesis

The frequencies of words in a document tend to indicate the relevance of the document to a query[6].—If documents and queries have similar column vectors in a term-document matrix, then they tend to have similar meanings.

Vector space models were first introduced for document retrieval where a query is treated as a pseudo-document[6]. The relevance of a document to a query can be measured by the similarity of their column vectors in a term-document matrix. Latent semantic indexing (LSI)[7] uses the singular value decomposition (SVD) technique to reduce the number of words while preserving the similarity among documents.

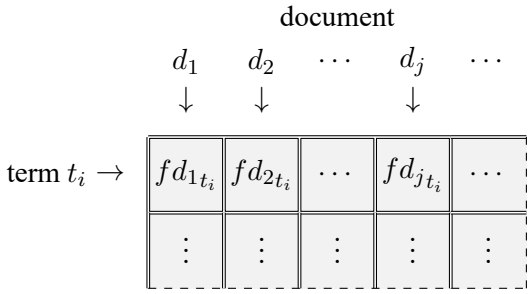


Figure 2.1: The term-document matrix.
In a term-document matrix, each row represents a unique term t_i and each column represents a document d_j. The element $fd_{j_{t_i}}$ is the frequency of term t_i in document d_j.

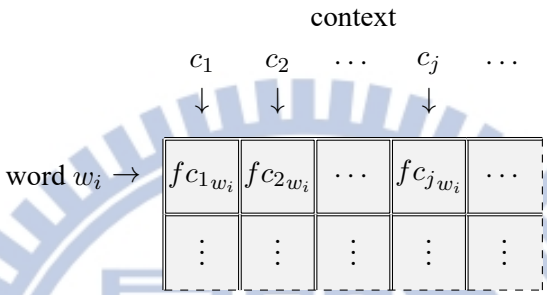


Figure 2.2: The word-context matrix.
In a word-context matrix, each row represents a unique word w_i and each column represents a context c_j. The element $fc_{j_{w_i}}$ is the frequency of word w_i in context c_j.

2.1.2 Similarity of Words: The Word-context Matrix

Hypothesis 2.1.2: Distributional Hypothesis

Words that occur in similar contexts tend to have similar meanings[8, 9].—If words have similar row vectors in a word-context matrix, then they tend to have similar meanings.

Researchers look into row vectors in a word-document matrix as word representations. However, documents are not the necessarily optimal contexts to understand word meanings, a word shall be known by the companies it keeps[9]. Windows of words are used as contexts in Word2vec[10] and GolVe[11], and other linguistic information such as grammatical dependencies can also be aggregated[12, 13].

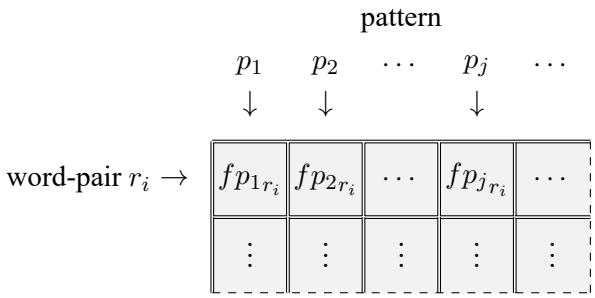


Figure 2.3: The pair-pattern matrix.
In a pair-pattern matrix, each row represents a unique word-pair r_i and each column represents a co-occurrence pattern p_j. The element $fp_{j_{r_i}}$ is the frequency of word-pair r_i in co-occurrence pattern p_j.

2.1.3 Similarity of Relations: The Pair-pattern Matrix

Hypothesis 2.1.3: Extended Distributional Hypothesis

Patterns co-occurring with similar word-pairs tend to have similar meanings[14].—If patterns have similar column vectors in a pair-pattern matrix, then they tend to express similar semantic relations.

Hypothesis 2.1.4: Latent Relation Hypothesis

Word-pairs co-occurring in similar patterns tend to have similar semantic relations[15]. —If word pairs have similar row vectors in a pair-pattern matrix, then they tend to have similar semantic relations.

These two hypothesis are complementary concepts. The pattern "*A eats B*" and "*B is eaten by A*" tend to co-occur with similar word-pairs (*A*, *B*), where the extended distributional hypothesis indicates that they may express similar semantic relations. The word-pairs (*cat*, *fish*) and (*panda*, *bamboo*) tend to co-occur with similar patter "*A eats B*", where the latent relation hypothesis indicates that they may have similar semantic relations.

2.2 Machine Learning for Text Classification

Documents are represented as vectors in a vector space model, hence any machine learning approach which works with vectors can work with vectors from a vector space model[16]. These

approaches put efforts into the construction of an automatic builder of classifiers rather than a classifier[3].

2.2.1 k-Nearest Neighbor Classifiers

Nearest neighbor classifiers use distance such as Euclidean distance, dot product and cosine, as similarity measures to perform the classification based on the idea that instances in the same class are likely to be closer to one another than to instances in different classes.

For an unlabeled document, we can search the k-nearest neighbors among the training documents and report the majority class from these k neighbors as the predicted class. The choice of k typically ranges between 20 and 40 in the previous works[17, 18, 19]. Another way is searching among representative meta-documents. We summarize a meta-document for each class, then apply the same approach as mentioned above. The preprocessing phase of summarization improves the efficiency of the classifier since it significantly reduces the number of distance computations[20, 21].

2.2.2 Naive Bayes Classifiers

Generative classifiers use an implicit mixture model for generation of the underlying instances. Each component of a mixture model is essentially a generative model for a class, which provides the probability of a particular feature for that class.

Naive Bayes classifiers are perhaps the most common generative classifiers, they model the distribution of an instance in each class by a probabilistic distribution with independence assumptions between the distributions of features. For an unlabeled document, the component generative models are used in conjunction with the Bayes rule to compute its posterior probability of each class, and the class with the highest posterior probability is reported as the predicted class[18, 22, 23, 24, 25, 26].

2.2.3 Logistic Regression Classifiers

Regressions are methods to discover the relationship between a dependent variable and one or more independent variables. Although these methods are designed for real values, we can

treat the binary value of a class as a special case of a real value and use them in classification[4].

An early application is linear least squares fit (LLSF) methods. They model the binary values of classes as a linear function of features, and try to minimize the squared errors between the modeled binary values of classes and the real ones[17, 18]. A more common application is logistic regression classifiers. Instead of modeling the binary values of classes, they model the distribution of the binary values of classes as a logistic function of a linear function, and try to maximize the probability of the real binary values[26, 27].

2.2.4 Support Vector Machine Classifiers

Support vector machines perform the classification by finding a hyperplane which best separates the different classes with the maximal margin[28]. Besides a linear separation, tehy can construct a non-linear separation in the original space by mapping instances non-linearly to an inner product space where classes can be separated linearly. However, linear support vector machines are used most often in practice because of their simplicity[4].

Since the hyperplane can be determined by examining the appropriate subset of instances as support vectors around the boundaries of classes, it is effective in high dimensional spaces and suited to text classification[18, 27, 29, 30, 31].

2.2.5 Neural Network Classifiers

Neural network is a collection of connected units called neurons. Each neuron receives inputs from predecessor neurons and produces outputs to successor neurons, one of each connection is associated with a weight used to compute a linear combination of inputs. An activation function or a threshold can be applied to neurons to induce non-linearity.

The use of multiple layers of neurons can approximate a non-linear boundary by multiple pieces of linear boundaries. The training process of such networks is more complex since the errors need to be back-propagated over different layers[18, 32, 33, 34]. However, in the case of text classification, some research observed that non-linear models do not yield significant improvement compared to linear models[35, 36].

Chapter 3

Methodology: CopeOpi Vectors

3.1 From CopeOpi Scores to Augmented CopeOpi Scores

3.1.1 CopeOpi Scores

CopeOpi scores[37] are numeric sentiment scores of Chinese characters and Chinese words, which can be used to detect the sentiment polarities and measure the strength of sentiment polarities. The basic idea is that the meaning of a Chinese word is the function of its composing characters, so the sentiment of a Chinese word can also be determined from the sentiments of its composing characters.

They assume that Chinese characters in a Chinese positive opinion word tend to be positive, and Chinese characters in a Chinese negative opinion word tend to be negative. They adopt a Chinese sentiment dictionary, the NTU sentiment dictionary (NTUSD)[38], as Chinese opinion seed words, and take the frequency of each Chinese character in these Chinese opinion seed words as clues to discover latent sentiments.

Once the CopeOpi scores of Chinese characters are available, the CopeOpi score of a Chinese word can be determined from the CopeOpi scores of its composing characters by applying a scoring function according to its morphological type[1], or simply taking the average[39].

Computation Scheme 3.1.1: CopeOpi Scores

Given two corpora of labeled Chinese opinion words $\mathbb{W}_p$ and $\mathbb{W}_n$, and the corresponding sentiment polarities $positive$ and $negative$.

- $\mathbb{W}_p = \{w \mid w \text{ is a Chinese opinion word labeled as } positive\}$
  - the vocabulary $\mathbb{V}_p$ is a set of unique Chinese characters in $\mathbb{W}_p$

[1] In linguistics, morphology is the study of the structure of words: how words are formed, and their relationship to each other in a language.

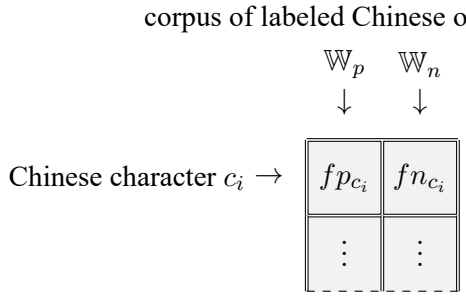


Figure 3.1: The frequency matrix of CopeOpi scores.
In the frequency matrix of CopeOpi scores, each row represents a unique Chinese character c_i and the two columns represent two corpora of labeled Chinese opinion words $\mathbb{W}_p$ and $\mathbb{W}_n$. The elements fp_{c_i} and fn_{c_i} are the frequencies of Chinese character c_i in corpus $\mathbb{W}_p$ and in corpus $\mathbb{W}_n$.

- $\mathbb{W}_n = \{w \mid w \text{ is a Chinese opinion word labeled as } negative\}$
  - the vocabulary $\mathbb{V}_n$ is a set of unique Chinese characters in $\mathbb{W}_n$

For each Chinese character $c_i \in \mathbb{V}_p \cup \mathbb{V}_n$, we can compute its CopeOpi score $\mathcal{COP}_{c_i}$.

$$\mathcal{P}_{c_i} = \frac{fp_{c_i} / \sum_{c \in \mathbb{V}_p} fp_c}{fp_{c_i} / \sum_{c \in \mathbb{V}_p} fp_c + fn_{c_i} / \sum_{c \in \mathbb{V}_n} fn_c}$$

$$\mathcal{N}_{c_i} = \frac{fn_{c_i} / \sum_{c \in \mathbb{V}_n} fn_c}{fp_{c_i} / \sum_{c \in \mathbb{V}_p} fp_c + fn_{c_i} / \sum_{c \in \mathbb{V}_n} fn_c}$$

$$\mathcal{COP}_{c_i} = \mathcal{P}_{c_i} - \mathcal{N}_{c_i}$$

where $\mathcal{P}_{c_i}$ and $\mathcal{N}_{c_i}$ are the normalized probabilities of Chinese character c_i being positive and being negative; fp_{c_i} and fn_{c_i} are the frequencies of Chinese character c_i in corpus $\mathbb{W}_p$ and in corpus $\mathbb{W}_n$; the CopeOpi score $\mathcal{COP}_{c_i}$ of Chinese character c_i is defined as the difference of the two opposite normalized probabilities, ranged from $+1$(being positive) to -1(being negative).

For each Chinese word $w_j = c_1 c_2 \cdots c_l$, we can compute its CopeOpi score $\mathcal{COP}_{w_j}$.

$$\mathcal{COP}_{w_j = c_1 c_2 \cdots c_l} = \begin{cases} S_m(c_1 c_2 \cdots c_l) & \text{if the morphological type of } w_j \text{ is } m \\ \frac{1}{l} \sum_{k=1}^{l} \mathcal{COP}_{c_k} & \text{otherwise} \end{cases}$$

where S_m is the scoring function of morphological type m.

One application of CopeOpi scores is augmented NTU sentiment dictionary (ANTUSD)[40], a collection of sentiment statistics of Chinese words in several sentiment annotation works. For each Chinese word in the dictionary, the number of positive annotations, neutral annotations, negative annotations, non-opinionated annotations and not-a-word annotations are recorded, and the CopeOpi score is also provided.

3.1.2 Augmented CopeOpi Scores

The structure of the frequency matrix of CopeOpi scores relates to its potential applications. In the frequency matrix of CopeOpi scores, the basic units are Chinese characters and the contexts are corpora of labeled Chinese opinion words, so CopeOpi scores find their applications in Chinese sentiment analysis.

However, since the core of CopeOpi scores is a bag-of-units method which is generally adopted in nature language processing, we think it is possible to augment their usefulness and widen their range of applications.

Here we propose a new computation scheme for augmented CopeOpi scores. We transform CopeOpi scores from sentiment scores to class-tendency scores by mapping the sentiment polarities in sentiment analysis, i.e., positive or negative, to the set membership in binary classification, i.e., being in a class or not. We extend the premises and assume that words in documents of some classes tend to be in those classes. We modify the structure of the frequency matrix and

- change the basic units from Chinese characters to words.
- change the contexts from corpora of Chinese opinion words to corpora of binary annotated documents.

Computation Scheme 3.1.2: Augmented CopeOpi Scores

Given two corpora of labeled documents $\mathbb{D}_p$ and $\mathbb{D}_n$, and the corresponding classes p and $not\text{-}p$.

- $\mathbb{D}_p = \{\langle d, c\rangle \mid d \text{ is a document labeled as class } c = p\}$
  - the vocabulary $\mathbb{V}_p$ is a set of unique words in $\mathbb{D}_p$
- $\mathbb{D}_n = \{\langle d, c\rangle \mid d \text{ is a document labeled as class } c = not\text{-}p\}$

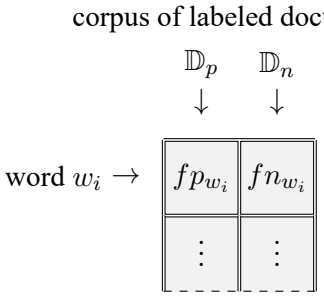


Figure 3.2: The frequency matrix of augmented CopeOpi scores.
In the frequency matrix of augmented CopeOpi scores, each row represents a unique word w_i and the two columns represent two corpora of labeled documents $\mathbb{D}_p$ and $\mathbb{D}_n$. The elements fp_{w_i} and fn_{w_i} are the frequencies of word w_i in corpus $\mathbb{D}_p$ and in corpus $\mathbb{D}_n$.

– the vocabulary $\mathbb{V}_n$ is a set of unique words in $\mathbb{D}_n$

For each word $w_i \in \mathbb{V}_p \cup \mathbb{V}_n$, we can compute its CopeOpi score $\mathcal{COP}_{w_i}$.

$$\mathcal{P}_{w_i} = \frac{fp_{w_i} / \sum_{w \in \mathbb{V}_p} fp_w}{fp_{w_i} / \sum_{w \in \mathbb{V}_p} fp_w + fn_{w_i} / \sum_{w \in \mathbb{V}_n} fn_w}$$

$$\mathcal{N}_{w_i} = \frac{fn_{w_i} / \sum_{w \in \mathbb{V}_n} fn_w}{fp_{w_i} / \sum_{w \in \mathbb{V}_p} fp_w + fn_{w_i} / \sum_{w \in \mathbb{V}_n} fn_w}$$

$$\mathcal{COP}_{w_i} = \mathcal{P}_{w_i} - \mathcal{N}_{w_i}$$

where $\mathcal{P}_{w_i}$ and $\mathcal{N}_{w_i}$ are the normalized probabilities of word w_i being in class p and being in class not-p; fp_{w_i} and fn_{w_i} are the frequencies of word w_i in corpus $\mathbb{D}_p$ and in corpus $\mathbb{D}_n$; the CopeOpi score $\mathcal{COP}_{w_i}$ of word w_i is defined as the difference of the two opposite normalized probabilities, ranged from $+1$(being in class p) to -1(being in class not-p).

Confidence in Augmented CopeOpi Scores

According to Zipf's law, given some corpora of natural language, the frequency of a word is inversely proportional to its rank in the frequency table[41]. The most frequent word occurs approximately twice as often as the second one, three times as often as the third one, etc. There are a few words that are very common and a lot of words that are very rare. Considering the later cases, we shall have less confidence in the augmented CopeOpi scores of rare words due to the lack of sufficient statistics, and besides, they are easily biased and overestimated since the occurrences of a rare word might be all in one class and absent in the other.

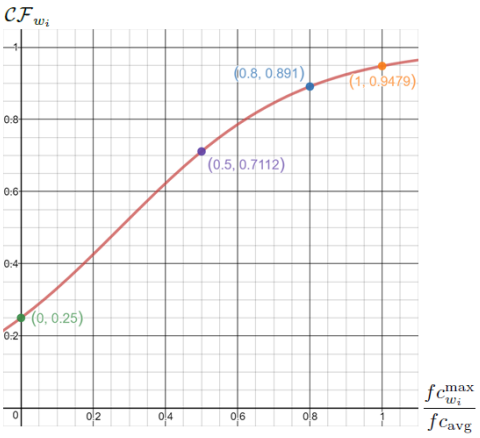


Figure 3.3: The logistic function of $\mathcal{CF}$

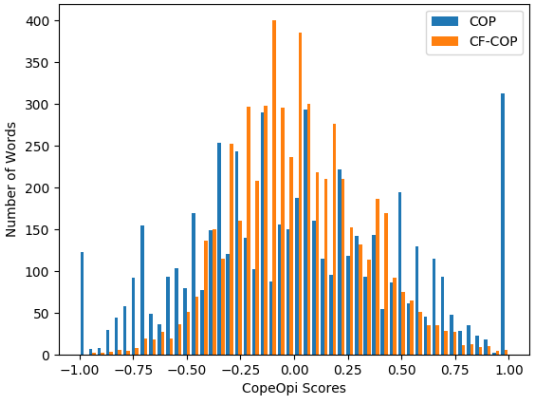


Figure 3.4: Distributions of $\mathcal{COP}$ and $\mathcal{CF}$-$\mathcal{COP}$

To reduce the effects of imprecise augmented CopeOpi scores of rare words, we can impose confidence values to penalize these values. We regard words whose maximal class frequency less than the average class frequency of all words as rare words, and smooth their augmented CopeOpi scores by multiplying their confidence values which is defined as a logistic function[2].

$$fc_{w_i}^{\max} = \max(fc_{1w_i}, fc_{2w_i}, \ldots, fc_{nw_i})$$

$$fc_{\text{avg}} = \frac{\sum_{j=1}^{m}\sum_{k=1}^{n} fc_{kw_j}}{m \times n}$$

$$\mathcal{CF}_{w_i} = \begin{cases} 1 & \text{if } fc_{w_i}^{\max} \geq fc_{\text{avg}} \\ \dfrac{1}{1 + 3\exp(-4(\frac{fc_{w_i}^{\max}}{fc_{\text{avg}}}))} & \text{otherwise} \end{cases}$$

$$\mathcal{CF}\text{-}\mathcal{COP}_{w_i} = \mathcal{CF}_{w_i} \times \mathcal{COP}_{w_i}$$

where fc_{kw_i} is the frequency of word w_i in class k; $fc_{w_i}^{\max}$ and fc_{avg} are the maximal class frequency of word w_i and the average class frequency of all words; n and m are the number of classes and the number unique words in corpora; the confidence value $\mathcal{CF}_{w_i}$ of word w_i is defined as a piece-wise function which behaves differently based on the maximal class frequency $fc_{w_i}^{\max}$ of word w_i.

[2]The mentioned scheme of confidence values is merely the one we use in our experiments but not a standard scheme. You can design one for your applications.

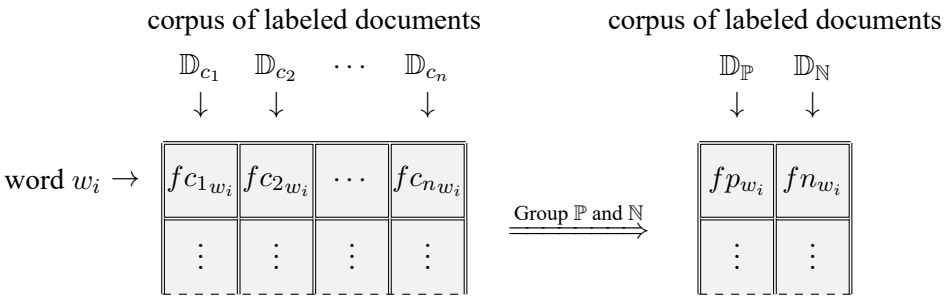


Figure 3.5: The frequency matrix of CopeOpi vectors.
In the frequency matrix of CopeOpi vectors, each row represents a unique word w_i and the n columns represent n corpora of labeled documents $\mathbb{D}_{c_1}, \mathbb{D}_{c_2}, \ldots, \mathbb{D}_{c_n}$. The element $fc_{j_{w_i}}$ is the frequency of word w_i in corpus $\mathbb{D}_{c_j}$. After $\mathbb{P}$ and $\mathbb{N}$ are grouped, the two columns represent two corpora of labeled documents $\mathbb{D}_{\mathbb{P}}$ and $\mathbb{D}_{\mathbb{N}}$. The elements fp_{w_i} and fn_{w_i} are the frequencies of word w_i in corpus $\mathbb{D}_{\mathbb{P}}$ and in corpus $\mathbb{D}_{\mathbb{N}}$.

3.2 From Augmented CopeOpi Scores to CopeOpi Vectors

3.2.1 CopeOpi Vectors

CopeOpi scores now become augmented CopeOpi scores, class-tendency scores which can be used in languages other than Chinese since we change the basic units from Chinese characters to words, and be applied to binary text classification other than sentiment analysis since we change the contexts from corpora of Chinese opinion words to corpora of binary annotated documents.

However, there are many text classification problems with more than two classes. Augmented CopeOpi scores can not help solve them due to the fact that they are scalars and can represent at most two oppositions by being positive or being negative.

Here we find a way to make augmented CopeOpi scores applicable to multiclass text classification. We expand augmented CopeOpi scores to CopeOpi vectors by utilizing divide-and-conquer techniques for multiclass classification. We decompose a multiclass text classification problem into multiple binary text classification subproblems, and compute an augmented CopeOpi score for each subproblem as a component of CopeOpi vectors. Several strategies have been proposed for such a decomposition[42].

One-versus-Rest[3] Strategy (OVR)

In an n-class text classification problem, we require n augmented CopeOpi scores, where each augmented CopeOpi score is computed to discriminate between one of the classes and the rest of the classes.

Computation Scheme 3.2.1: CopeOpi Vectors (One-versus-Rest)

Given n corpora of labeled documents $\mathbb{D} = \{\mathbb{D}_{c_1}, \mathbb{D}_{c_2}, \ldots, \mathbb{D}_{c_n}\}$, and the corresponding classes $\mathbb{C} = \{c_1, c_2, \ldots, c_n\}$.

- $\mathbb{D}_{c_i} = \{\langle d, c\rangle \mid d \text{ is a document labeled as class } c = c_i\}$
  - the vocabulary $\mathbb{V}_{c_i}$ is a set of unique words in $\mathbb{D}_{c_i}$

For each word $w_i \in \cup_{c\in\mathbb{C}}\mathbb{V}_c$, we can compute its CopeOpi vector $\overrightarrow{\mathcal{COP}}_{w_i}$ by one-versus-rest strategy.

For each class $c_j \in \mathbb{C}$, we can construct two opposite sets,

- the positive set $\mathbb{P}^{c_j}_{w_i} = \{c_j\}$
  - the positive corpus $\mathbb{D}_{\mathbb{P}^{c_j}_{w_i}} = \{\mathbb{D}_{c_j}\}$
  - the positive vocabulary $\mathbb{V}_{\mathbb{P}^{c_j}_{w_i}} = \mathbb{V}_{c_j}$
- the negative set $\mathbb{N}^{c_j}_{w_i} = \mathbb{C} \setminus \{c_j\}$
  - the negative corpus $\mathbb{D}_{\mathbb{N}^{c_j}_{w_i}} = \mathbb{D} \setminus \{\mathbb{D}_{c_j}\}$
  - the negative vocabulary $\mathbb{V}_{\mathbb{N}^{c_j}_{w_i}} = \cup_{c\in\mathbb{N}^{c_j}_{w_i}}\mathbb{V}_c$

and compute the augmented CopeOpi score $\mathcal{COP}^{c_j}_{w_i}$ of word w_i with respect to class c_j based on these two opposite sets.

$$\mathcal{P}^{c_j}_{w_i} = \frac{fp^{c_j}_{w_i} / \sum_{w\in\mathbb{V}_{\mathbb{P}^{c_j}_{w_i}}} fp^{c_j}_{w}}{fp^{c_j}_{w_i} / \sum_{w\in\mathbb{V}_{\mathbb{P}^{c_j}_{w_i}}} fp^{c_j}_{w} + fn^{c_j}_{w_i} / \sum_{w\in\mathbb{V}_{\mathbb{N}^{c_j}_{w_i}}} fn^{c_j}_{w}}$$

$$\mathcal{N}^{c_j}_{w_i} = \frac{fn^{c_j}_{w_i} / \sum_{w\in\mathbb{V}_{\mathbb{N}^{c_j}_{w_i}}} fn^{c_j}_{w}}{fp^{c_j}_{w_i} / \sum_{w\in\mathbb{V}_{\mathbb{P}^{c_j}_{w_i}}} fp^{c_j}_{w} + fn^{c_j}_{w_i} / \sum_{w\in\mathbb{V}_{\mathbb{N}^{c_j}_{w_i}}} fn^{c_j}_{w}}$$

$$\mathcal{COP}^{c_j}_{w_i} = \mathcal{P}^{c_j}_{w_i} - \mathcal{N}^{c_j}_{w_i}$$

where $\mathcal{P}^{c_j}_{w_i}$ and $\mathcal{N}^{c_j}_{w_i}$ are the normalized probabilities of word w_i being in class $\mathbb{P}^{c_j}_{w_i}$ and

[3] Also known as one-versus-all (OVA), one-against-rest (OAR), one-against-all (OAA).

being in class $\mathbb{N}_{w_i}^{c_j}$; $fp_{w_i}^{c_j}$ and $fn_{w_i}^{c_j}$ are the frequencies of word w_i in corpus $\mathbb{D}_{\mathbb{P}_{w_i}^{c_j}}$ and in corpus $\mathbb{D}_{\mathbb{N}_{w_i}^{c_j}}$; the CopeOpi score $\mathcal{COP}_{w_i}^{c_j}$ of word w_i with respect to class c_j is defined as the difference of the two opposite normalized probabilities, ranged from $+1$(being in class $\mathbb{P}_{w_i}^{c_j}$) to -1(being in class $\mathbb{N}_{w_i}^{c_j}$).

The CopeOpi vector $\overrightarrow{\mathcal{COP}}_{w_i}$ of word w_i is composed of these n augmented CopeOpi scores.

$$\overrightarrow{\mathcal{COP}}_{w_i} = (\mathcal{COP}_{w_i}^{c_1}, \mathcal{COP}_{w_i}^{c_2}, \ldots, \mathcal{COP}_{w_i}^{c_n})$$

One-versus-One[4] Strategy (OVO)

In an n-class text classification problem, we require $\frac{1}{2}n(n-1)$ augmented CopeOpi scores, where each augmented CopeOpi score is computed to discriminate between a pair of classes.

Computation Scheme 3.2.2: CopeOpi Vectors (One-versus-One)

Given n corpora of labeled documents $\mathbb{D} = \{\mathbb{D}_{c_1}, \mathbb{D}_{c_2}, \ldots, \mathbb{D}_{c_n}\}$, and the corresponding classes $\mathbb{C} = \{c_1, c_2, \ldots, c_n\}$.

- $\mathbb{D}_{c_i} = \{\langle d, c\rangle \mid d \text{ is a document labeled as class } c = c_i\}$
  - the vocabulary $\mathbb{V}_{c_i}$ is a set of unique words in $\mathbb{D}_{c_i}$

For each word $w_i \in \cup_{c\in\mathbb{C}}\mathbb{V}_c$, we can compute its CopeOpi vector $\overrightarrow{\mathcal{COP}}_{w_i}$ by one-versus-one strategy.

For each class-pair $c_j, c_k \in \mathbb{C}$ where $1 \le j < k \le n$, we can construct two opposite sets,

- the positive set $\mathbb{P}_{w_i}^{c_j,c_k} = \{c_j\}$
  - the positive corpus $\mathbb{D}_{\mathbb{P}_{w_i}^{c_j,c_k}} = \{\mathbb{D}_{c_j}\}$
  - the positive vocabulary $\mathbb{V}_{\mathbb{P}_{w_i}^{c_j,c_k}} = \mathbb{V}_{c_j}$
- the negative set $\mathbb{N}_{w_i}^{c_j,c_k} = \{c_k\}$
  - the negative corpus $\mathbb{D}_{\mathbb{N}_{w_i}^{c_j,c_k}} = \{\mathbb{D}_{c_k}\}$
  - the negative vocabulary $\mathbb{V}_{\mathbb{N}_{w_i}^{c_j,c_k}} = \mathbb{V}_{c_k}$

and compute the augmented CopeOpi score $\mathcal{COP}_{w_i}^{c_j,c_k}$ of word w_i with respect to class-pair

[4] Also known as one-against-one (OAO).

c_j, c_k based on these two opposite sets.

$$\mathcal{P}_{w_i}^{c_j,c_k} = \frac{fp_{w_i}^{c_j,c_k} / \sum_{w \in \mathbb{V}_{\mathbb{P}_{w_i}^{c_j,c_k}}} fp_w^{c_j,c_k}}{fp_{w_i}^{c_j,c_k} / \sum_{w \in \mathbb{V}_{\mathbb{P}_{w_i}^{c_j,c_k}}} fp_w^{c_j,c_k} + fn_{w_i}^{c_j,c_k} / \sum_{w \in \mathbb{V}_{\mathbb{N}_{w_i}^{c_j,c_k}}} fn_w^{c_j,c_k}}$$

$$\mathcal{N}_{w_i}^{c_j,c_k} = \frac{fn_{w_i}^{c_j,c_k} / \sum_{w \in \mathbb{V}_{\mathbb{N}_{w_i}^{c_j,c_k}}} fn_w^{c_j,c_k}}{fp_{w_i}^{c_j,c_k} / \sum_{w \in \mathbb{V}_{\mathbb{P}_{w_i}^{c_j,c_k}}} fp_w^{c_j,c_k} + fn_{w_i}^{c_j,c_k} / \sum_{w \in \mathbb{V}_{\mathbb{N}_{w_i}^{c_j,c_k}}} fn_w^{c_j,c_k}}$$

$$\mathcal{COP}_{w_i}^{c_j,c_k} = \mathcal{P}_{w_i}^{c_j,c_k} - \mathcal{N}_{w_i}^{c_j,c_k}$$

where $\mathcal{P}_{w_i}^{c_j,c_k}$ and $\mathcal{N}_{w_i}^{c_j,c_k}$ are the normalized probabilities of word w_i being in class $\mathbb{P}_{w_i}^{c_j,c_k}$ and being in class $\mathbb{N}_{w_i}^{c_j,c_k}$; $fp_{w_i}^{c_j,c_k}$ and $fn_{w_i}^{c_j,c_k}$ are the frequencies of word w_i in corpus $\mathbb{D}_{\mathbb{P}_{w_i}^{c_j,c_k}}$ and in corpus $\mathbb{D}_{\mathbb{N}_{w_i}^{c_j,c_k}}$; the CopeOpi score $\mathcal{COP}_{w_i}^{c_j,c_k}$ of word w_i with respect to class-pair c_j, c_k is defined as the difference of the two opposite normalized probabilities, ranged from $+1$(being in class $\mathbb{P}_{w_i}^{c_j,c_k}$) to -1(being in class $\mathbb{N}_{w_i}^{c_j,c_k}$).

The CopeOpi vector $\overrightarrow{\mathcal{COP}}_{w_i}$ of word w_i is composed of these $\frac{1}{2}n(n-1)$ augmented CopeOpi scores.

$$\overrightarrow{\mathcal{COP}}_{w_i} = (\mathcal{COP}_{w_i}^{c_1,c_2}, \mathcal{COP}_{w_i}^{c_1,c_3}, \ldots, \mathcal{COP}_{w_i}^{c_{n-1},c_n})$$

3.2.2 Customized CopeOpi Vectors

One-versus-rest strategy and one-versus-one strategy guide the basic way to construct CopeOpi vectors which can be applied to multiclass text classification.

However, in general, any subset of classes can be grouped as a positive set or a negative set. CopeOpi vectors can be customized based on different choices of subset-pairs.

Subset-versus-Subset Strategy (SVS)

In an n-class text classification problem, we can have at most $(2^n - 1)^2$ augmented CopeOpi scores[5], where each augmented CopeOpi score is computed to discriminate between a pair of subsets of classes.

Computation Scheme 3.2.3: CopeOpi Vectors (Subset-versus-Subset)

Given n corpora of labeled documents $\mathbb{D} = \{\mathbb{D}_{c_1}, \mathbb{D}_{c_2}, \ldots, \mathbb{D}_{c_n}\}$, and the corresponding classes $\mathbb{C} = \{c_1, c_2, \ldots, c_n\}$.

- $\mathbb{D}_{c_i} = \{\langle d, c\rangle \mid d \text{ is a document labeled as class } c = c_i\}$
  - the vocabulary $\mathbb{V}_{c_i}$ is a set of unique words in $\mathbb{D}_{c_i}$

For each word $w_i \in \cup_{c\in\mathbb{C}}\mathbb{V}_c$, we can compute its CopeOpi vector $\overrightarrow{\mathcal{COP}}_{w_i}$ by subset-versus-subset strategy.

For any subset-pair $\mathbb{J}, \mathbb{K} \subseteq \mathbb{C}$ where $\mathbb{J}, \mathbb{K} \neq \varnothing$, can construct two opposite sets,

- the positive set $\mathbb{P}_{w_i}^{\mathbb{J},\mathbb{K}} = \mathbb{J}$
  - the positive corpus $\mathbb{D}_{\mathbb{P}_{w_i}^{\mathbb{J},\mathbb{K}}} = \{\mathbb{D}_c \mid c \in \mathbb{J}\}$
  - the positive vocabulary $\mathbb{V}_{\mathbb{P}_{w_i}^{\mathbb{J},\mathbb{K}}} = \cup_{c\in\mathbb{J}}\mathbb{V}_c$
- the positive set $\mathbb{N}_{w_i}^{\mathbb{J},\mathbb{K}} = \mathbb{K}$
  - the positive corpus $\mathbb{D}_{\mathbb{N}_{w_i}^{\mathbb{J},\mathbb{K}}} = \{\mathbb{D}_c \mid c \in \mathbb{K}\}$
  - the positive vocabulary $\mathbb{V}_{\mathbb{N}_{w_i}^{\mathbb{J},\mathbb{K}}} = \cup_{c\in\mathbb{K}}\mathbb{V}_c$

and compute the augmented CopeOpi score $\mathcal{COP}_{w_i}^{\mathbb{J},\mathbb{K}}$ of word w_i with respect to subset-

[5] We only eliminate the empty set cases, but not all subset-pairs can produce a discriminative augmented CopeOpi score.

pair $\mathbb{J}, \mathbb{K}$ based on these two opposite sets.

$$\mathcal{P}_{w_i}^{\mathbb{J},\mathbb{K}} = \frac{fp_{w_i}^{\mathbb{J},\mathbb{K}} / \sum_{w \in \mathbb{V}_{\mathbb{P}_{w_i}^{\mathbb{J},\mathbb{K}}}} fp_w^{\mathbb{J},\mathbb{K}}}{fp_{w_i}^{\mathbb{J},\mathbb{K}} / \sum_{w \in \mathbb{V}_{\mathbb{P}_{w_i}^{\mathbb{J},\mathbb{K}}}} fp_w^{\mathbb{J},\mathbb{K}} + fn_{w_i}^{\mathbb{J},\mathbb{K}} / \sum_{w \in \mathbb{V}_{\mathbb{N}_{w_i}^{\mathbb{J},\mathbb{K}}}} fn_w^{\mathbb{J},\mathbb{K}}}$$

$$\mathcal{N}_{w_i}^{\mathbb{J},\mathbb{K}} = \frac{fn_{w_i}^{\mathbb{J},\mathbb{K}} / \sum_{w \in \mathbb{V}_{\mathbb{N}_{w_i}^{\mathbb{J},\mathbb{K}}}} fn_w^{\mathbb{J},\mathbb{K}}}{fp_{w_i}^{\mathbb{J},\mathbb{K}} / \sum_{w \in \mathbb{V}_{\mathbb{P}_{w_i}^{\mathbb{J},\mathbb{K}}}} fp_w^{\mathbb{J},\mathbb{K}} + fn_{w_i}^{\mathbb{J},\mathbb{K}} / \sum_{w \in \mathbb{V}_{\mathbb{N}_{w_i}^{\mathbb{J},\mathbb{K}}}} fn_w^{\mathbb{J},\mathbb{K}}}$$

$$\mathcal{COP}_{w_i}^{\mathbb{J},\mathbb{K}} = \mathcal{P}_{w_i}^{\mathbb{J},\mathbb{K}} - \mathcal{N}_{w_i}^{\mathbb{J},\mathbb{K}}$$

where $\mathcal{P}_{w_i}^{\mathbb{J},\mathbb{K}}$ and $\mathcal{N}_{w_i}^{\mathbb{J},\mathbb{K}}$ are the normalized probabilities of word w_i being in class $\mathbb{P}_{w_i}^{\mathbb{J},\mathbb{K}}$ and being in class $\mathbb{N}_{w_i}^{\mathbb{J},\mathbb{K}}$; $fp_{w_i}^{\mathbb{J},\mathbb{K}}$ and $fn_{w_i}^{\mathbb{J},\mathbb{K}}$ are the frequencies of word w_i in corpus $\mathbb{D}_{\mathbb{P}_{w_i}^{\mathbb{J},\mathbb{K}}}$ and in corpus $\mathbb{D}_{\mathbb{N}_{w_i}^{\mathbb{J},\mathbb{K}}}$; the CopeOpi score $\mathcal{COP}_{w_i}^{\mathbb{J},\mathbb{K}}$ of word w_i with respect to subset-pair $\mathbb{J}, \mathbb{K}$ is defined as the difference of the two opposite normalized probabilities, ranged from $+1$(being in class $\mathbb{P}_{w_i}^{\mathbb{J},\mathbb{K}}$) to -1(being in class $\mathbb{N}_{w_i}^{\mathbb{J},\mathbb{K}}$).

The CopeOpi vector $\overrightarrow{\mathcal{COP}}_{w_i}$ of word w_i is composed of these at most $(2^n-1)^2$ augmented CopeOpi scores.

$$\overrightarrow{\mathcal{COP}}_{w_i} = (\mathcal{COP}_{w_i}^{\mathbb{J}_1,\mathbb{K}_1}, \mathcal{COP}_{w_i}^{\mathbb{J}_2,\mathbb{K}_2}, \dots)$$

Chapter 4

Experiments and Results

To verify the functionality of CopeOpi vectors, we make comparisons with several commonly-used features for text classificaiton, and examine these features on different types of machine learning algorithms to solve text classification problems, including sentiment analysis and topic categorization, in both English and Chinese.

4.1 Flowchart and Settings

Figure 4.1 shows the flowchart of our experiments, detailed settings are described in the following sections.

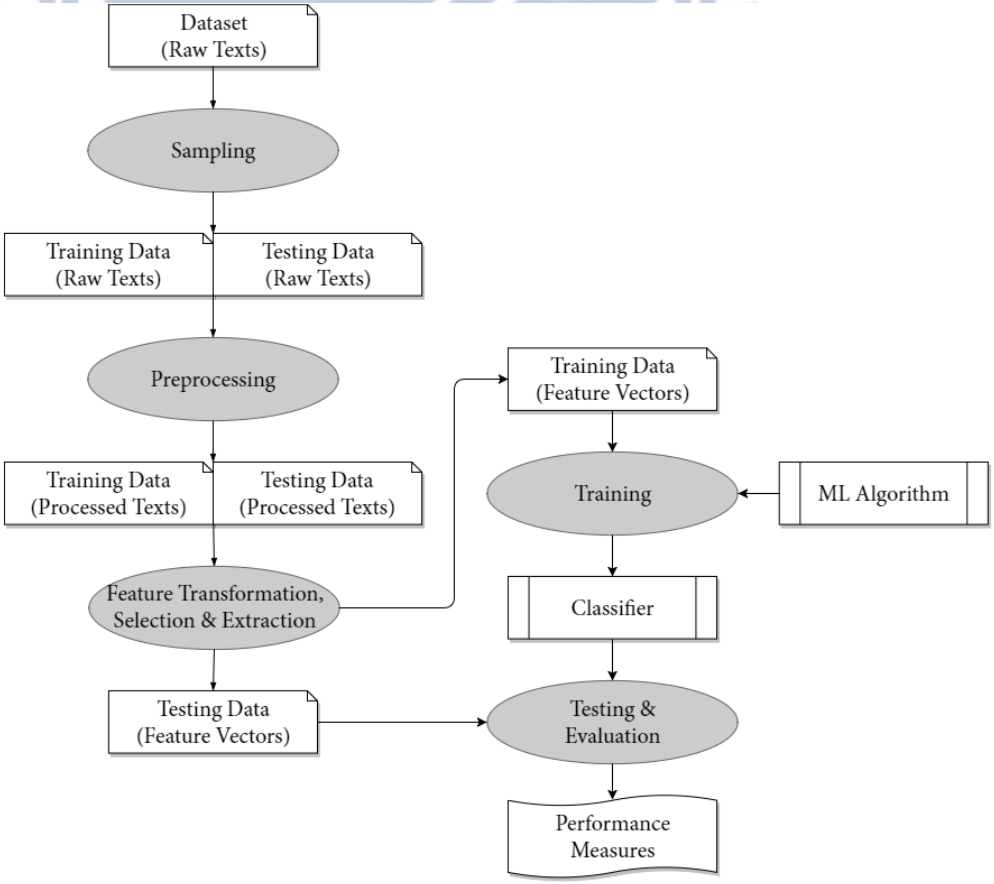


Figure 4.1: The flowchart of experiments

Table 4.1 lists the open sources used in our experiments. They provide high quality implementation and ensure the reliability of experiment results.

Table 4.1: The open sources used in experiments

Open Source	Functions	Reference
`scikit-learn`[1]	Machine learning toolbox	[43]
`gensim`[2]	Nature language processing toolbox	[44]
Stanford GloVe Project[3]	GloVe implementation	[11]
`jieba`[4]	Chinese word segmenter	[45]

4.1.1 Sampling

In most operational machine learning settings, once a classifier has been constructed, it is desirable to evaluate its performance. In this case, prior to classifier construction, the initial dataset is split into two parts[3, 46]:

- training set: a set of labeled instances used to construct a classifier, to fit the parameters of a machine learning algorithm.
  - validation set: a set of labeled instances used to tune the hyperparameters of a machine learning algorithm.
- testing set: a set of labeled instances used to evaluate a classifier, to assess the performance of a fully-specified classifier.

Experiment Settings are described in the sections of experiments.

4.1.2 Preprocessing

It is useful to apply some linguistic processing to raw texts before performing an analysis on them. Things to consider include:

- text cleaning: stripping unwanted tags, punctuation, numeric, etc.
- tokenization: deciding what constitutes a unit and how to extract units.

[1] http://scikit-learn.org
[2] http://radimrehurek.com/gensim
[3] http://nlp.stanford.edu/projects/glove
[4] http://github.com/fxsjy/jieba

- normalization: converting superficially-different units to the same form.
- stopword removal: removing frequent words which do not carry much meaning.

Experiment Settings are described in Table 4.2.

We unify the preprocessing for each language:

- for English corpora, we use `gensim.parsing.preprocessing.preprocess_string` for text cleaning, tokenization, normalization and stopword removal.
- for Chinese corpora, we use `jieba.cut` for tokenization; and strip characters outside the CJK Unified Ideographs[5] UTF-8 block.

Table 4.2: Experiment settings of preprocessings

Language	Text Cleaning	Tokenization	Normalization	Stopword Removal
English	Stripping tags, punctuation, multiple whitespaces, numeric and shorts	Yes	Lowercasing, Stemming	Yes
Chinese	`[^\u4E00-\u9FFF]`	Yes	No	No

4.1.3 Feature Transformation, Selection and Extraction

Machine learning algorithms require a numerical representation of objects, so we have to transform documents as feature vectors before text classification. Due to the high-dimensional sparse property and the existence of irrelevant features, dimension reduction is especially important for texts[4]:

- feature selection: picking a subset of features from the original feature set.
- feature extraction: creating a new set of features from the original feature set.

Experiment Settings are described in Table 4.3.

We make comparisons with several commonly-used features for text classificaiton, including:

- term-document matrix models:
  - bag-of-word (BOW)

[5]The Chinese, Japanese and Korean scripts share a common background, known as CJK characters.

  - LSI[7]-truncated BOW (BOW(LSI))
  - term frequency-inverse document frequency (TF-IDF)
  - LSI-truncated TF-IDF (TF-IDF(LSI))
- word-context matrix models:
  - Word2vec[10]
  - GloVe[11]

Table 4.3: Experiment settings of features

Feature	Implementaion	Feature Selection	Normalized
CopeOpi	Our implementaion	`min_count=5`	Yes
BOW	`sklearn.feature_extraction.text.CountVectorizer`	`min_df=0.001`	No
BOW(LSI)	`sklearn.feature_extraction.text.CountVectorizer` `sklearn.decomposition.TruncatedSVD`	`min_df=0.001`	Yes
TF-IDF	`sklearn.feature_extraction.text.TfidfVectorizer`	`min_df=0.001`	Yes
TF-IDF(LSI)	`sklearn.feature_extraction.text.TfidfVectorizer` `sklearn.decomposition.TruncatedSVD`	`min_df=0.001`	Yes
Word2vec	`gensim.models.word2vec`	`min_count=5`	Yes
GloVe	Stanford GloVe Project	`min_count=5`	Yes

4.1.4 Training

Since documents may be represented as feature vectors, it is possible to use most machine learning algorithms for text classification, as we discussed in section 2.2.

Experiment Settings are described in Table 4.4.

We examine the mentioned features on different types of machine learning algorithms, including:

- k-Nearest neighbor classifiers (kNN)
- naive Bayes classifiers (NB)
- logistic regression classifiers (LR)
- support vector machine classifiers (SVM)
- neural network classifiers (NN)

Table 4.4: Experiment settings of machine learning algorithms

Machine Learining Algorithm	Implementaion
k-Nearest Neighbor (kNN)	`sklearn.neighbors.KNeighborsClassifier(n_neighbors=40)`
Naive Bayes (NB)	`sklearn.naive_bayes.MultinomialNB` for BOW, TF-IDF `sklearn.naive_bayes.GaussianNB` for the others
Logistic Regression (LR)	`sklearn.linear_model.LogisticRegression`
Support Vector Machine (SVM)	`sklearn.svm.LinearSVC`
Neural Network (NN)	`sklearn.neural_network.MLPClassifier`

4.1.5 Testing and Evaluation

Table 4.5 is the contingency table of binary classification. Performance of a binary classifier is usually measured in terms of precision, recall and F1-score[3], which are defined as follows:

- precision: the fraction of predicted-true that are real-true.
- recall: the fraction of the real-true that are predicted-true.
- F1-score: the harmonic mean of precision and recall.

$$\text{precision}_c = \frac{TP_c}{TP_c + FP_c}$$

$$\text{recall}_c = \frac{TP_c}{TP_c + FN_c}$$

$$F_{1c} = \frac{2 \times \text{precision}_c \times \text{recall}_c}{\text{precision}_c + \text{recall}_c}$$

Table 4.5: The contingency table of binary classification

		Real	
		`True`	`False`
Predicted	`True`	True positive (TP)	False positive (FP)
	`False`	False negative (FN)	True negative (TN)

Experiment Settings

As for multiclass classification, we use F1-score as measure for each class, and take average of them as a macro F1-score.

$$\text{macro-}F_1 = \frac{1}{|\mathbb{C}|} \sum_{c \in \mathbb{C}} F_{1c}$$

4.2 Experiments: Sentiment Analysis

We use SA($lang$)(n) as the abbreviation standing for sentiment analysis experiment n of language $lang$, where EN represents English and ZH represents Chinese.

4.2.1 Datasets

We use Yelp Dataset[6][47] as the English corpus and MioChnCorp[48] as the Chinese corpus. Both are user-assigned five-star integral rating sentiment annotated, ranged from 5 as positive to 1 as negative. Table 4.6 is the descriptions about these datasets.

Experiment Datasets are summarized in Table 4.7.

We randomly select 15000 instances from the original dataset for each experiment, and split into a training set and a testing set in ratio 0.5:0.5.

- SA(EN|ZH)(A): 2 classes are positive and negative.
- SA(EN|ZH)(B): 3 classes are positive, negative and neutral.
- SA(EN|ZH)(C): 5 classes are 1-star, 2-star, 3-star, 4-star and 5-star.

Table 4.6: Sentiment analysis datasets

Dataset	Language	# of Classes	Descriptions	Source
Yelp Dataset	English	5	Customer reviews about local business such as restaurants, hair stylists, mechanics, etc.	Yelp[7]
MioChnCorp	Chinese	5	Customer reviews about hotels.	Dianping[8]

Table 4.7: Sentiment analysis experiments datasets

(a) Sampling of SA(EN) and SA(ZH).

Classes	(A)	(B)	(C)
1-star	Negative(3750:3750)	Negative(2500:2500)	1-star(1500:1500)
2-star			2-star(1500:1500)
3-star		Neutral(2500:2500)	3-star(1500:1500)
4-star	Positive(3750:3750)		4-star(1500:1500)
5-star		Positive(2500:2500)	5-star(1500:1500)
Total	7500:7500	7500:7500	7500:7500

[6]The version we use is Yelp Dataset Challenge round 9.
[7]http://www.yelp.com
[8]http://www.dianping.com/hotel

4.2.2 Results and Observations

We use macro F_1-scores as the measure of effectiveness and training CPU time as the measure of efficiency. Results[9] are shown in Table 4.8 to Table 4.13.

Here we brief our observations about the results of sentiment analysis experiments:

1. In SA(EN)(A) and SA(ZH)(A), which are binary text classification problems, the CopeOpi here are augmented CopeOpi scores. Compare the best macro F_1-score of CopeOpi and the best macro F_1-score of each experiment, we lose by 4.57% in SA(EN)(A) and 2.08% in SA(ZH)(A). This shows that the computation scheme of augmented CopeOpi scores is feasible in both English and Chinese binary text classification.
2. In SA(ZH)(A), which is a binary Chinese sentiment analysis problem, we compare with the CopeOpi scores recorded in ANTUSD[40], which are original CopeOpi scores. The augmented CopeOpi scores outperform the CopeOpi scores recorded in ANTUSD by more than 10% for each classifier. This shows that augmented CopeOpi scores function normally without manually filtering irrelevant words and are more applicable to the dataset.
3. In SA(EN)(B), SA(EN)(C), SA(ZH)(B) and SA(ZH)(C), which are multiclass text classification problems, the CopeOpi here are CopeOpi vectors constructed by one-versus-rest, one-versus-one and both strategies. Compare the best macro F_1-score of CopeOpi and the best macro F_1-score of each experiment, we lose by 2.10% in SA(EN)(B), 2.58% in SA(EN)(C), 0.42% in SA(ZH)(B) and 1.30% in SA(ZH)(C). This shows that the computation scheme of CopeOpi vectors is feasible in both English and Chinese multiclass text classification.
4. $\frac{4}{10}$ of the macro F_1-scores of CopeOpi scores in SA(EN|ZH)(A) are better than the average macro F_1-score of each classifier of each experiment, $\frac{18}{20}$ of the best macro F_1-scores of CopeOpi vectors in SA(EN|ZH)(B|C) are better than the average macro F_1-score of each classifier of each experiment. This shows that CopeOpi vectors in muticlass classification are more effective than augmented CopeOpi scores in binary classification.

[9]The values of F_1-scores are the larger the better, while the values of training CPU time are the smaller the better. In the table of results, color green represents a better result while color red represents a worse result; three bold values are the top three results of each classifier.

Table 4.8: Results of SA(EN)(A)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi	**0.8500**	**0.8567**	0.8565	0.8565	0.8563
BOW	0.5867	**0.8595**	**0.8909**	0.8711	0.8828
BOW(LSI)	0.8033	0.8050	0.8733	0.8757	0.8708
TF-IDF	0.8109	**0.8752**	**0.9024**	**0.8959**	0.8739
TF-IDF(LSI)	0.8000	0.7724	**0.8920**	**0.8908**	**0.8904**
Word2vec	**0.8301**	0.7729	0.8817	**0.8848**	**0.8840**
GloVe	**0.8465**	0.7841	0.8745	0.8812	**0.8848**
Average	0.7896	0.8180	0.8816	0.8794	0.8776

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi	**0.0075**	**0.0010**	**0.0054**	**0.0109**	**0.8644**
BOW	2.6780	0.2199	0.4979	0.4085	76.6209
BOW(LSI)	0.1176	**0.0357**	0.2554	0.2440	15.8738
TF-IDF	2.7285	0.1114	**0.1530**	**0.1397**	99.9166
TF-IDF(LSI)	0.1120	0.0369	**0.1636**	**0.2328**	**14.3500**
Word2vec	**0.0978**	**0.0367**	0.2925	0.2439	16.0138
GloVe	**0.1108**	0.0367	0.2583	0.2640	**6.8514**

Unit: second

Table 4.9: Results of SA(EN)(B)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.7645**	**0.7584**	0.7618	0.7609	0.7656
CopeOpi(OVO)	**0.7701**	**0.7567**	0.7635	0.7642	0.7664
CopeOpi(OVR+OVO)	**0.7668**	**0.7568**	0.7679	0.7730	**0.7729**
BOW	0.5748	0.7473	0.7726	0.7352	0.7545
BOW(LSI)	0.6551	0.6615	0.7607	0.7619	0.7395
TF-IDF	0.6730	0.7518	**0.7940**	**0.7774**	0.7356
TF-IDF(LSI)	0.6637	0.6640	**0.7861**	**0.7848**	**0.7695**
Word2vec	0.6694	0.6199	**0.7765**	**0.7796**	0.7567
GloVe	0.6992	0.6455	0.7661	0.7730	**0.7741**
Average	0.6929	0.7069	0.7721	0.7678	0.7594

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0060**	**0.0057**	**0.0182**	**0.0766**	**1.8836**
CopeOpi(OVO)	**0.0062**	**0.0023**	**0.0168**	**0.0719**	**1.8149**
CopeOpi(OVR+OVO)	**0.0070**	**0.0063**	**0.0347**	**0.0671**	**2.0696**
BOW	2.7120	0.2236	1.2555	1.4736	97.9738
BOW(LSI)	0.1156	0.0360	0.5752	0.7813	16.7357
TF-IDF	2.7251	0.1252	0.2994	0.2721	122.5705
TF-IDF(LSI)	0.1164	0.0502	0.4389	0.7185	16.7917
Word2vec	0.0993	0.0369	0.7833	0.7753	16.4355
GloVe	0.0994	0.0363	0.6689	0.8252	5.4702

Unit: second

Table 4.10: Results of SA(EN)(C)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.4997**	**0.4952**	0.4912	0.4853	0.4882
CopeOpi(OVO)	**0.4994**	0.4836	0.4921	0.4933	**0.4961**
CopeOpi(OVR+OVO)	**0.4987**	0.4886	0.4909	0.4916	**0.4925**
BOW	0.3194	**0.5008**	0.4820	0.4468	0.4819
BOW(LSI)	0.4030	0.4221	0.4902	0.4806	0.4536
TF-IDF	0.4184	**0.5068**	**0.5255**	0.4924	0.4614
TF-IDF(LSI)	0.4125	0.4427	**0.5172**	**0.5122**	0.4868
Word2vec	0.4316	0.4037	**0.5062**	**0.4955**	0.4781
GloVe	0.4478	0.4147	0.5003	**0.4964**	**0.5071**
Average	0.4367	0.4620	0.4995	0.4882	0.4829

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0068**	**0.0030**	**0.0392**	**0.0934**	**1.9119**
CopeOpi(OVO)	**0.0088**	**0.0031**	**0.0642**	**0.1284**	**3.5210**
CopeOpi(OVR+OVO)	**0.0102**	**0.0038**	**0.1197**	**0.1721**	**3.0343**
BOW	2.7722	0.2403	2.4426	3.4236	135.0993
BOW(LSI)	0.1250	0.0356	1.0235	1.3927	17.1394
TF-IDF	2.7773	0.1102	0.4829	0.4308	181.9951
TF-IDF(LSI)	0.1130	0.0368	0.8758	1.3449	16.9712
Word2vec	0.1004	0.0356	1.3021	1.4126	17.1082
GloVe	0.1057	0.0360	1.1555	1.5035	12.5198

Unit: second

Table 4.11: Results of SA(ZH)(A)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(ANTUSD)	0.7550	0.7268	0.7540	0.7536	0.7559
CopeOpi	**0.8682**	**0.8655**	0.8679	0.8679	0.8723
BOW	0.7779	**0.8711**	**0.8899**	0.8625	0.8703
BOW(LSI)	0.8204	0.7551	0.8688	0.8748	**0.8739**
TF-IDF	**0.8565**	**0.8788**	**0.8931**	**0.8896**	0.8544
TF-IDF(LSI)	0.8495	0.8024	**0.8856**	**0.8878**	**0.8855**
Word2vec	**0.8594**	0.8350	0.8827	**0.8875**	**0.8872**
GloVe	0.8507	0.8289	0.8631	0.8707	0.8716
Average	0.8297	0.8205	0.8631	0.8618	0.8589

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi	**0.0071**	**0.0010**	0.0053	**0.0130**	**1.2354**
BOW	2.0473	0.2022	0.4575	0.5648	90.6920
BOW(LSI)	0.1120	0.0367	0.2492	0.2680	16.2093
TF-IDF	2.0677	0.1033	**0.1248**	**0.1224**	124.8922
TF-IDF(LSI)	0.1075	**0.0361**	**0.2077**	0.2689	16.0591
Word2vec	**0.0985**	**0.0362**	0.2605	**0.2661**	**15.9893**
GloVe	**0.0985**	0.0365	**0.1945**	0.3035	**4.2587**

Unit: second

Table 4.12: Results of SA(ZH)(B)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.7656**	0.7178	0.7702	**0.7702**	**0.7648**
CopeOpi(OVO)	**0.7706**	**0.7201**	0.7656	0.7661	**0.7675**
CopeOpi(OVR+OVO)	**0.7672**	0.7174	**0.7715**	**0.7723**	**0.7685**
BOW	0.5404	**0.7672**	0.7604	0.7165	0.7302
BOW(LSI)	0.6830	0.6787	0.7489	0.7529	0.7325
TF-IDF	0.7224	**0.7676**	**0.7755**	0.7560	0.7077
TF-IDF(LSI)	0.7091	0.6852	**0.7765**	**0.7740**	0.7495
Word2vec	0.7402	0.7034	0.7695	0.7695	0.7484
GloVe	0.7254	0.6871	0.7473	0.7520	0.7580
Average	0.7138	0.7161	0.7650	0.7588	0.7474

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0059**	**0.0023**	**0.0164**	**0.0748**	**1.9661**
CopeOpi(OVO)	**0.0060**	**0.0023**	**0.0183**	**0.0644**	**1.2055**
CopeOpi(OVR+OVO)	**0.0073**	**0.0028**	**0.0249**	**0.0787**	**2.4152**
BOW	2.2442	0.1819	1.2493	1.7848	103.4950
BOW(LSI)	0.1186	0.0365	0.6958	0.8129	17.0334
TF-IDF	2.2567	0.0977	0.2455	0.2405	139.9930
TF-IDF(LSI)	0.1081	0.0371	0.5222	0.7777	17.0605
Word2vec	0.0979	0.0530	0.7019	0.7977	17.2022
GloVe	0.1055	0.0368	0.6270	0.8789	9.0869

Unit: second

Table 4.13: Results of SA(ZH)(C)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.4662**	0.4253	0.4597	0.4582	0.4644
CopeOpi(OVO)	**0.4664**	0.4308	0.4631	**0.4594**	**0.4659**
CopeOpi(OVR+OVO)	**0.4687**	**0.4343**	0.4603	0.4575	0.4619
BOW	0.3022	**0.4816**	0.4593	0.4305	0.4497
BOW(LSI)	0.3862	0.3890	0.4521	0.4461	0.4504
TF-IDF	0.4147	**0.4772**	**0.4754**	0.4509	0.4316
TF-IDF(LSI)	0.4123	0.4104	**0.4699**	**0.4621**	0.4610
Word2vec	0.4380	0.4234	**0.4729**	**0.4589**	**0.4696**
GloVe	0.4370	0.4179	0.4524	0.4463	**0.4713**
Average	0.4213	0.4322	0.4628	0.4522	0.4584

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0068**	**0.0027**	**0.0419**	**0.1064**	**1.4366**
CopeOpi(OVO)	**0.0087**	**0.0073**	**0.0740**	**0.1495**	**1.5058**
CopeOpi(OVR+OVO)	**0.0101**	**0.0035**	**0.1007**	**0.1757**	**2.3752**
BOW	2.1371	0.1789	2.1119	3.2249	150.1767
BOW(LSI)	0.1174	0.0370	1.2061	1.4276	17.1054
TF-IDF	2.0971	0.1051	0.4143	0.3606	188.9623
TF-IDF(LSI)	0.1081	0.0365	0.9024	1.4000	17.4045
Word2vec	0.0983	0.0533	1.3385	1.4142	17.1789
GloVe	0.0995	0.0359	1.0447	1.5520	10.5434

Unit: second

4.3 Experiments: Topic Categorization

We use TC($lang$)(n) as the abbreviation standing for topic categorization experiment n of language $lang$, where EN represents English and ZH represents Chinese.

4.3.1 Datasets

We use 20 Newsgroups[49] as the English corpus and Fudan University TC Corpus[50] as the Chinese corpus. Both contains 20 categories and training-and-testing splits. Table 4.14 is the descriptions about these datasets.

Experiment Datasets are summarized in Table 4.15.

We subset categories from the original dataset for each experiment, and use their default training-and-testing splits.

- TC(EN):
  - TC(EN)(A): 20 classes are all categories of the original dataset.
  - TC(EN)(B): 7 classes are the first hierarchy categories.
  - TC(EN)(C): 5 classes are categories within the first hierarchy comp.
  - TC(EN)(D): 4 classes are categories within the first hierarchy talk.
- TC(ZH):
  - TC(ZH)(A): 20 classes are all categories of the original dataset.
  - TC(ZH)(B): 9 classes are categories with more than a hundred instances.
  - TC(ZH)(C): 11 classes are categories with less than a hundred instances.

Table 4.14: Topic categorization datasets

Dataset	Language	# of Classes	Balanced	Train:Test
20 Newsgroup	English	20	Yes	0.6:0.4
Fudan University TC Corpus	Chinese	20	No	0.5:0.5

Table 4.15: Topic categorization experiments datasets

(a) Sampling of TC(EN)

Classes	(A)	(B)	(C)	(D)
alt.atheism	480:319	480:319		
comp.graphics	584:389	2936:1955	584:389	
comp.os.ms-windows.misc	591:394		591:394	
comp.sys.ibm.pc.hardware	590:392		590:392	
comp.sys.mac.hardware	578:385		578:385	
comp.windows.x	593:395		593:395	
misc.forsale	585:390	585:390		
rec.autos	594:396	2389:1590		
rec.motorcycles	598:398			
rec.sport.baseball	597:397			
rec.sport.hockey	600:399			
sci.crypt	595:396	2373:1579		
sci.electronics	591:393			
sci.med	594:396			
sci.space	593:394			
soc.religion.christian	599:398	599:398		
talk.politics.guns	546:364	1952:1301		546:364
talk.politics.mideast	564:376			564:376
talk.politics.misc	465:310			465:310
talk.religion.misc	377:251			377:251
Total	11314:7532	11314:7532	2936:1955	1952:1301

(b) Sampling of TC(ZH)

Classes	(A)	(B)	(C)
Art	740:742	740:742	
Literature	33:34		33:34
Education	59:61		59:61
Philosophy	44:45		44:45
History	466:468	466:468	
Space	640:642	640:642	
Energy	32:33		32:33
Electronics	27:28		27:28
Communication	25:27		25:27
Computer	1357:1358	1357:1358	
Mine	33:34		33:34
Transport	57:59		57:59
Environment	1217:1218	1217:1218	
Agriculture	1021:1022	1021:1022	
Economy	1600:1601	1600:1601	
Law	51:52		51:52
Medical	51:53		51:53
Military	74:76		74:76
Politics	1024:1026	1024:1026	
Sports	1253:1254	1253:1254	
Total	9804:9833	9318:9331	486:502

4.3.2 Results and Observations

We use macro F_1-scores as the measure of effectiveness and training CPU time as the measure of efficiency. Results[10] are shown in Table 4.16 to Table 4.22.

Here we brief our observations about the results of topic categorization experiments:

1. (i) In TC(EN)(A) and TC(ZH)(A), both corpora contain 20 categories. Compare the best macro F_1-score of CopeOpi and the best macro F_1-score of each experiment, we lose by 0.87% in TC(EN)(A) but 14.01% in TC(ZH)(A). CopeOpi vectors function badly in one of them. Except languages, the biggest difference between their corpus is the balance. But we doubt if CopeOpi vectors can not function well in imbalanced corpora since the influence of imbalance should be smoothed by the normalization of their formulas.

   (ii) In TC(EN)(B) and TC(ZH)(A), both corpora are imbalanced. Compare the best macro F_1-score of CopeOpi and the best macro F_1-score of each experiment, we lose by 1.03% in TC(EN)(B) but 14.01% in TC(ZH)(A). CopeOpi vectors function well

[10]The values of F_1-scores are the larger the better, while the values of training CPU time are the smaller the better. In the table of results, color green represents a better result while color red represents a worse result; three bold values are the top three results of each classifier.

in one of them. Except languages, the biggest difference between their corpus is the size of the training set. We deduce that the reason why CopeOpi vectors function badly in TC(ZH)(A) is due to the lack of training instances, not imbalance.

2. In TC(ZH)(B) and TC(ZH)(C), the former corpus contains categories with more than a hundred instances, the later corpus contains categories with less than a hundred instances. Compare the best macro F_1-score of CopeOpi and the best macro F_1-score of each experiment, we lose by 3.02% in TC(ZH)(B) but 8.95% in TC(ZH)(C). This confirms the deduction that CopeOpi vectors can not function well if there are no sufficient training instances.
3. In TC(EN)(C) and TC(EN)(D), both corpora contain categories with similar topics. Compare the best macro F_1-score of CopeOpi and the best macro F_1-score of each experiment, we lose by 2.33% in TC(EN)(C) and 0.51% in TC(EN)(D). This shows that CopeOpi vectors can function well even though categories are similar.

Table 4.16: Results of TC(EN)(A)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.8082**	**0.7779**	**0.8111**	**0.8096**	**0.8099**
CopeOpi(OVO)	**0.7778**	0.7519	0.7913	0.7914	0.7816
CopeOpi(OVR+OVO)	**0.7841**	0.7573	**0.7944**	**0.7941**	0.7831
BOW	0.3778	**0.7969**	0.7781	0.7490	**0.7962**
BOW(LSI)	0.5541	0.5867	0.6852	0.6977	0.6974
TF-IDF	0.6885	**0.7933**	**0.8011**	**0.8122**	**0.8198**
TF-IDF(LSI)	0.6180	0.6631	0.7557	0.7584	0.7544
Word2vec	0.6581	0.6090	0.7303	0.7350	0.7231
GloVe	0.6250	0.5679	0.6839	0.7051	0.7078
Average	0.6546	0.7004	0.7590	0.7614	0.7637

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0213**	**0.0092**	**1.5266**	**0.2954**	**5.9940**
CopeOpi(OVO)	**0.1707**	0.0510	7.5718	**1.1431**	**13.3517**
CopeOpi(OVR+OVO)	0.1957	0.0474	8.0230	**1.2040**	**16.2402**
BOW	12.0553	0.8421	22.6672	4.5479	220.0564
BOW(LSI)	0.2015	**0.0423**	8.1444	2.8707	27.5949
TF-IDF	11.5024	0.4743	**4.1413**	1.5051	275.8628
TF-IDF(LSI)	0.1944	**0.0422**	**7.1338**	2.2947	28.2907
Word2vec	0.1795	0.0538	8.4990	2.4934	27.5775
GloVe	**0.1765**	0.0434	7.8233	3.3244	28.0873

Unit: second

Table 4.17: Results of TC(EN)(B)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.8503**	**0.8390**	**0.8471**	**0.8492**	**0.8487**
CopeOpi(OVO)	**0.8485**	0.8184	0.8405	0.8477	**0.8476**
CopeOpi(OVR+OVO)	**0.8503**	**0.8256**	**0.8465**	**0.8501**	0.8472
BOW	0.4359	**0.8315**	**0.8440**	0.8136	0.8447
BOW(LSI)	0.6364	0.6498	0.7547	0.7652	0.7794
TF-IDF	0.7130	0.7045	0.8166	**0.8606**	**0.8548**
TF-IDF(LSI)	0.6840	0.7284	0.8081	0.8170	0.8320
Word2vec	0.7490	0.6431	0.7746	0.7854	0.8075
GloVe	0.7139	0.6333	0.7248	0.7578	0.7842
Average	0.7201	0.7415	0.8063	0.8163	0.8273

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0125**	**0.0045**	**0.1148**	**0.0736**	**3.4896**
CopeOpi(OVO)	**0.0209**	**0.0072**	**0.3796**	**0.1498**	**3.9274**
CopeOpi(OVR+OVO)	**0.0259**	**0.0090**	**0.4513**	**0.1609**	**3.5161**
BOW	11.6014	0.8025	8.4964	3.4058	195.9696
BOW(LSI)	0.2082	0.0755	2.4409	1.3690	25.8456
TF-IDF	11.4605	0.4013	1.4116	0.7822	275.7318
TF-IDF(LSI)	0.2083	0.0536	1.7775	1.1947	26.1040
Word2vec	0.1755	0.0699	2.4950	1.2888	25.4154
GloVe	0.1816	0.0541	2.3848	1.5595	25.8291

Unit: second

Table 4.18: Results of TC(EN)(C)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.7694**	**0.7745**	**0.7727**	**0.7754**	0.7733
CopeOpi(OVO)	**0.7493**	0.7497	0.7558	0.7564	0.7524
CopeOpi(OVR+OVO)	**0.7576**	**0.7605**	0.7662	0.7708	0.7700
BOW	0.4415	0.7582	0.7670	0.7470	**0.7794**
BOW(LSI)	0.6449	0.6367	0.7464	0.7550	0.7410
TF-IDF	0.7117	**0.7827**	**0.7987**	**0.7935**	**0.7808**
TF-IDF(LSI)	0.7105	0.6546	**0.7806**	**0.7784**	**0.7737**
Word2vec	0.6890	0.6730	0.7350	0.7421	0.7459
GloVe	0.6600	0.6470	0.7055	0.7148	0.7118
Average	0.6816	0.7152	0.7587	0.7593	0.7587

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0027**	**0.0016**	**0.0155**	**0.0160**	**0.9426**
CopeOpi(OVO)	**0.0034**	**0.0017**	**0.0252**	**0.0219**	**1.1210**
CopeOpi(OVR+OVO)	**0.0044**	**0.0044**	**0.0335**	**0.0221**	**1.0868**
BOW	1.6376	0.2072	1.2144	0.8721	44.4903
BOW(LSI)	0.0414	0.0179	0.3432	0.2893	6.6512
TF-IDF	1.5805	0.0928	0.2477	0.1791	103.3264
TF-IDF(LSI)	0.0376	0.0208	0.2432	0.2698	6.5757
Word2vec	0.0321	0.0224	0.3545	0.3147	6.4947
GloVe	0.0302	0.0109	0.3881	0.3797	5.9971

Unit: second

Table 4.19: Results of TC(EN)(D)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.8443**	**0.8414**	**0.8467**	**0.8434**	**0.8498**
CopeOpi(OVO)	**0.8379**	**0.8376**	0.8354	0.8371	0.8365
CopeOpi(OVR+OVO)	**0.8428**	**0.8414**	**0.8400**	**0.8425**	**0.8446**
BOW	0.4988	0.8361	0.8164	0.8045	0.8406
BOW(LSI)	0.7498	0.7248	0.7939	0.8009	0.7820
TF-IDF	0.8230	0.8306	**0.8465**	**0.8549**	**0.8519**
TF-IDF(LSI)	0.8128	0.7628	0.8301	0.8269	0.8193
Word2vec	0.7963	0.7756	0.8070	0.8042	0.7773
GloVe	0.7390	0.7183	0.7616	0.7686	0.7569
Average	0.7716	0.7965	0.8197	0.8203	0.8177

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0019**	**0.0031**	**0.0086**	**0.0057**	**0.7391**
CopeOpi(OVO)	**0.0019**	**0.0013**	**0.0087**	**0.0058**	**0.7482**
CopeOpi(OVR+OVO)	**0.0024**	**0.0014**	**0.0107**	**0.0089**	**0.6990**
BOW	1.6744	0.1768	0.9133	0.3098	45.0946
BOW(LSI)	0.0242	0.0158	0.1595	0.1170	4.3214
TF-IDF	1.6210	0.0911	0.1863	0.1621	74.8630
TF-IDF(LSI)	0.0227	0.0084	0.1142	0.1029	3.7858
Word2vec	0.0190	0.0088	0.1758	0.1363	4.3839
GloVe	0.0189	0.0088	0.1793	0.1773	3.4777

Unit: second

Table 4.20: Results of TC(ZH)(A)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.5645**	**0.6150**	0.5160	0.6464	0.6254
CopeOpi(OVO)	**0.5852**	0.5962	0.5458	0.6347	0.6360
CopeOpi(OVR+OVO)	**0.5841**	**0.6003**	**0.5474**	0.6422	0.6469
BOW	0.2789	0.5448	**0.7722**	**0.7705**	**0.7870**
BOW(LSI)	0.3613	0.3628	0.3964	0.5066	0.6300
TF-IDF	0.4145	0.3138	0.4969	**0.7848**	**0.7788**
TF-IDF(LSI)	0.5423	**0.6055**	**0.5683**	**0.7081**	**0.7706**
Word2vec	0.4468	0.4263	0.4843	0.6598	0.7603
GloVe	0.4508	0.4392	0.3719	0.5417	0.6914
Average	0.4698	0.5005	0.5221	0.6550	0.7029

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0179**	**0.0078**	**0.7900**	**0.2755**	**7.1952**
CopeOpi(OVO)	**0.1348**	0.0652	**4.9250**	**1.3058**	**13.6160**
CopeOpi(OVR+OVO)	0.1529	**0.0512**	5.2028	**1.3762**	**13.5071**
BOW	251.2883	41.0996	225.6336	58.1989	593.9647
BOW(LSI)	14.1084	0.2723	6.0229	2.4928	29.9708
TF-IDF	82.5466	3.6343	65.8209	12.8500	1889.1603
TF-IDF(LSI)	1.0824	0.1299	5.1678	1.6444	25.8863
Word2vec	**0.1419**	0.0551	5.6908	1.8916	25.0606
GloVe	0.1443	**0.0507**	**4.4622**	2.7077	22.0329

Unit: second

Table 4.21: Results of TC(ZH)(B)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.9048**	**0.8270**	0.8955	0.9120	0.9103
CopeOpi(OVO)	**0.8876**	0.7824	0.8722	0.9001	0.9008
CopeOpi(OVR+OVO)	**0.8926**	0.7965	0.8843	0.9091	0.9116
BOW	0.5939	**0.8811**	**0.9293**	**0.9182**	**0.9253**
BOW(LSI)	0.8178	0.5949	0.8617	0.8923	0.9128
TF-IDF	0.8088	0.7141	**0.9183**	**0.9400**	**0.9422**
TF-IDF(LSI)	0.8735	**0.8065**	**0.9081**	**0.9218**	**0.9346**
Word2vec	0.8188	0.6723	0.8579	0.8993	0.9186
GloVe	0.8533	0.7178	0.8191	0.8876	0.9086
Average	0.8279	0.7547	0.8830	0.9089	0.9183

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0119**	**0.0048**	**0.1440**	**0.0831**	**3.7704**
CopeOpi(OVO)	**0.0257**	**0.0126**	**0.6906**	**0.2351**	**3.9419**
CopeOpi(OVR+OVO)	**0.0315**	**0.0111**	**0.7911**	**0.2585**	**7.6480**
BOW	192.5686	27.9069	145.5820	93.9456	477.7269
BOW(LSI)	8.5769	1.4740	7.9850	1.7934	24.5064
TF-IDF	97.6548	3.9224	44.5169	10.6979	1733.3296
TF-IDF(LSI)	0.9676	0.0822	2.6450	1.0308	23.7611
Word2vec	0.1359	0.0658	2.6201	1.0924	17.6179
GloVe	0.1361	0.0436	2.2012	1.4262	13.8404

Unit: second

Table 4.22: Results of TC(ZH)(C)

(a) Macro F_1-score

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.7490**	**0.7282**	**0.7722**	0.8118	0.7725
CopeOpi(OVO)	**0.7158**	0.7080	0.7355	0.7545	0.7547
CopeOpi(OVR+OVO)	**0.7173**	**0.7246**	0.7370	0.7763	0.7618
BOW	0.3102	**0.7269**	**0.8103**	**0.8174**	**0.8230**
BOW(LSI)	0.4751	0.5603	0.6817	0.8107	0.8051
TF-IDF	0.6027	0.4734	0.7398	**0.9014**	**0.8702**
TF-IDF(LSI)	0.6095	0.5451	**0.8353**	**0.8899**	**0.8849**
Word2vec	0.5753	0.6411	0.6683	0.7696	0.7766
GloVe	0.4679	0.4732	0.3328	0.5783	0.5467
Average	0.5803	0.6201	0.7014	0.7900	0.7773

(b) Training CPU Time

Feature	kNN	NB	LR	SVM	NN
CopeOpi(OVR)	**0.0008**	**0.0014**	**0.0067**	**0.0062**	**0.5709**
CopeOpi(OVO)	**0.0016**	**0.0023**	**0.0239**	**0.0111**	**0.7250**
CopeOpi(OVR+OVO)	**0.0018**	**0.0018**	**0.0266**	**0.0125**	**0.7456**
BOW	0.6514	0.2512	0.8639	0.7423	25.5131
BOW(LSI)	0.0054	0.0056	0.0791	0.0834	1.2624
TF-IDF	0.3605	0.0704	0.2458	0.1609	90.0523
TF-IDF(LSI)	0.0052	0.0026	0.0781	0.0627	1.2447
Word2vec	0.0039	0.0027	0.0702	0.0581	1.2163
GloVe	0.0038	0.0026	0.0624	0.1237	1.2096

Unit: second

4.4 Summary

Here we summarize our observations about CopeOpi vectors:

1. CopeOpi vectors can produce comparable results with a smaller vector size and shorter training time in multiclass text classification.
   (i) $\frac{49}{55}$ of the best macro F_1-scores of CopeOpi vectors in multiclass classification are better than the average macro F_1-score of each classifier of each experiment.
   (ii) The vector size of CopeOpi vectors is the smallest in all experiment, as listed in Table 4.23.
   (iii) The training time of CopeOpi vectors is the shortest in all experiment.
2. Compared with the other features, CopeOpi vectors provide stabler results when applied to different types of machine learning algorithms. There are some results deviating, but in those cases the deviations are general phenomenons for most of the features.
3. Compared with the winner TF-IDF, CopeOpi vectors lose by at most 3.02% except the experiments without sufficient training instances. Although TF-IDFs perform the best, they also costs most in terms of memory space and training time. Since the vector size of TF-IDFs is proportional to the number of unique words in a corpus, in the cases of large corpus, CopeOpi vectors will have advantages in efficiency.

Table 4.23: The vector sizes in experiments

	SA						TC						
	(EN)			(ZH)			(EN)				(ZH)		
	(A)	(B)	(C)	(A)	(B)	(C)	(A)	(B)	(C)	(D)	(A)	(B)	(C)
CopeOpi(OVR)	1	3	5	1	3	5	20	7	5	4	20	9	11
CopeOpi(OVO)		3	10		3	10	190	21	10	6	190	36	55
CopeOpi(OVR+OVO)		6	15		6	15	210	28	15	10	210	45	66
BOW	3746	3846	3942	3108	3145	3148	8647	8647	8284	11851	46409	45992	23920
BOW(LSA)	200												
TF-TDF	3746	3846	3942	3108	3145	3148	8647	8647	8284	11851	46409	45992	23920
TF-TDF(LSA)	200												
Word2vec	200												
GloVe	200												

Chapter 5

Conclusions and Future Works

5.1 Conclusions

In this thesis, we propose a vector space model for multiclass text classification, the word vectors—CopeOpi vectors. We expand CopeOpi scores which are used in Chinese sentiment analysis, to CopeOpi vectors which can be used in multiclass text classification without the language limit.

We verify the functionality of CopeOpi vectors by a series of text classification problems, including sentiment analysis and topic categorization, in both English and Chinese. We make comparisons with several commonly-used features for text classification, and examine these features on different types of machine learning algorithms. The results show that CopeOpi vectors can produce comparable results with a smaller vector size and shorter training time. CopeOpi vectors are effective and efficient features for multiclass text classification.

5.2 Future Works

Some issues about augmented CopeOpi scores and CopeOpi vectors are listed below as a reference for those who are interested in.

More Careful Term-weighting Schemes

Although CopeOpi vectors function normally without manually filtering irrelevant words, it does not mean these irrelevant words do not effect the resulting values.

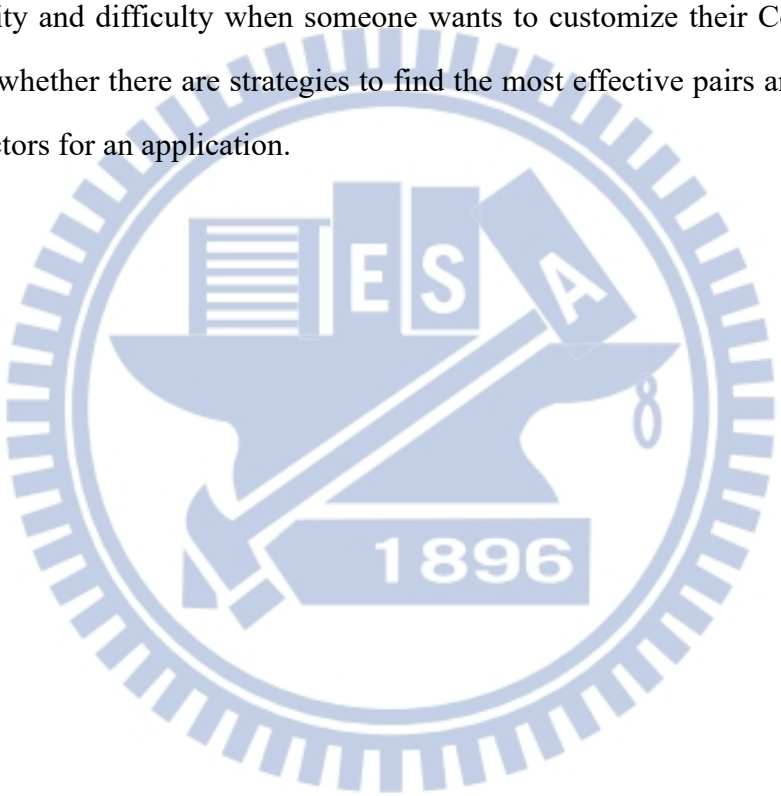
Since the original CopeOpi scores are computed from dictionaries, their term-weighting scheme is simply dividing the frequency of a word by the total frequency of all words. Now we compute augmented CopeOpi scores from nature language corpora, there are a lot of irrelevant words. Augmented CopeOpi scores may need a more careful term-weighting scheme to

improve their precision.

Strategies to Customize CopeOpi Vectors

In section 3.2.2, we mentioned that CopeOpi vectors can be customized based on different choices of subset-pairs. However, we have not comprehensively explored the capacity of customized CopeOpi vectors in our experiments.

Since the number of subset-pairs is exponential to the number of classes, and the number of structures of CopeOpi vectors are also exponential to the number of subset-pairs. It provides both flexibility and difficulty when someone wants to customize their CopeOpi vectors. We wonder that whether there are strategies to find the most effective pairs and construct the best CopeOpi vectors for an application.



References

[1] R. Feldman and J. Sanger, *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2007.

[2] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[3] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[4] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," *Mining Text Data*, pp. 163–222, 2012.

[5] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *Journal of Artificial Intelligence Research*, vol. 37, pp. 141–188, 2010.

[6] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, p. 391, 1990.

[8] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[9] J. R. Firth, *A Synopsis of Linguistic Theory, 1930-1955*, 1957.

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

[11] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.

[12] D. Lin, "Automatic retrieval and clustering of similar words," in *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*. Association for Computational Linguistics, 1998, pp. 768–774.

[13] S. Padó and M. Lapata, "Dependency-based construction of semantic space models," *Computational Linguistics*, vol. 33, no. 2, pp. 161–199, 2007.

[14] D. Lin and P. Pantel, "DIRT@ SBT@ discovery of inference rules from text," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 323–328.

[15] P. D. Turney and M. L. Littman, "Measuring praise and criticism: Inference of semantic orientation from association," *ACM Transactions on Information Systems*, vol. 21, no. 4, pp. 315–346, 2003.

[16] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.

[17] Y. Yang and C. G. Chute, "An example-based mapping method for text categorization and retrieval," *ACM Transactions on Information Systems*, vol. 12, no. 3, pp. 252–277, 1994.

[18] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1999, pp. 42–49.

[19] E.-H. S. Han, G. Karypis, and V. Kumar, "Text categorization using weight adjusted k-nearest neighbor classification," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2001, pp. 53–65.

[20] J. J. Rocchio, "Relevance feedback in information retrieval," *The Smart Retrieval System-experiments in Automatic Document Processing*, 1971.

[21] E.-H. S. Han and G. Karypis, "Centroid-based document classification: Analysis and experimental results," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 424–431.

[22] T. Joachims, "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization," in *Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 143–151.

[23] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," in *Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 170–178.

[24] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *European Conference on Machine Learning*. Springer, 1998, pp. 4–15.

[25] A. McCallum, K. Nigam *et al.*, "A comparison of event models for naive Bayes text classification," in *AAAI-98 Workshop on Learning for Text Categorization*, vol. 752. Madison, WI, 1998, pp. 41–48.

[26] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes," in *Advances in Neural Information Processing Systems*, 2002, pp. 841–848.

[27] J. Zhang and Y. Yang, "Robustness of regularized linear classification methods in text categorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. ACM, 2003, pp. 190–197.

[28] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[29] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," *Proceedings of the 10th European Conference on Machine Learning*, pp. 137–142, 1998.

[30] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proceedings of the Sixteenth International Conference on Machine Learning*, vol. 99, 1999, pp. 200–209.

[31] T. Joachims, “A statistical learning learning model of text classification for support vector machines,” in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2001, pp. 128–136.

[32] S. L. Lam and D. L. Lee, “Feature reduction for neural network based text categorization,” in *Database Systems for Advanced Applications, 1999. Proceedings., 6th International Conference on*. IEEE, 1999, pp. 195–202.

[33] M. E. Ruiz and P. Srinivasan, “Hierarchical neural networks for text categorization,” in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1999, pp. 281–282.

[34] A. S. Weigend, E. D. Wiener, and J. O. Pedersen, “Exploiting hierarchy in text categorization,” *Information Retrieval*, vol. 1, no. 3, pp. 193–216, 1999.

[35] H. Schütze, D. A. Hull, and J. O. Pedersen, “A comparison of classifiers and document representations for the routing problem,” in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1995, pp. 229–237.

[36] E. Wiener, J. O. Pedersen, A. S. Weigend *et al.*, “A neural network approach to topic spotting,” in *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, vol. 317. Las Vegas, NV, 1995, p. 332.

[37] L.-W. Ku, Y.-S. Lo, and H.-H. Chen, “Using polarity scores of words for sentence-level opinion extraction,” in *Proceedings of NTCIR-6 Workshop Meeting*, 2007, pp. 316–322.

[38] L.-W. Ku and H.-H. Chen, “Mining opinions from the web: Beyond relevance retrieval,” *Journal of the American Society for Information Science and Technology*, vol. 58, no. 12, pp. 1838–1850, 2007.

[39] L.-W. Ku, T.-H. Huang, and H.-H. Chen, “Using morphological and syntactic structures for Chinese opinion analysis,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*. Association for Computational Linguistics, 2009, pp. 1260–1269.

[40] S.-M. Wang and L.-W. Ku, "ANTUSD: A large Chinese sentiment dictionary," in *The Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2016, pp. 2697–2702.

[41] C. D. Manning, H. Schütze *et al.*, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[42] M. Aly, "Survey on multiclass classification methods," *Neural Netw*, vol. 19, 2005.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[44] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, 2010, pp. 45–50.

[45] J. Sun, "jieba Chinese text segmentation: Built to be the best Python Chinese word segmentation module," 2017. [Online]. Available: http://github.com/fxsjy/jieba

[46] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 2007.

[47] "Yelp Dataset Challenge," 2017. [Online]. Available: http://www.yelp.com/dataset_challenge

[48] Y. Lin, H. Lei, J. Wu, and X. Li, "An empirical study on sentiment classification of Chinese review using word embedding," *arXiv preprint arXiv:1511.01665*, 2015.

[49] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the 12th International Conference on Machine Learning*, vol. 10, 1995, pp. 331–339.

[50] R.-L. Li, "Fudan University TC Corpus," 2017. [Online]. Available: http://tjzhifei.github.io/resources/fudancorp.zip