



**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**SENTIMENT ANALYSIS BASED ON
COMBINATION OF TERM WEIGHTING
SCHEMES AND WORD VECTORS**

JIN LINBO

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2015



**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**SENTIMENT ANALYSIS BASED ON
COMBINATION OF TERM WEIGHTING
SCHEMES AND WORD VECTORS**

JIN LINBO

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

**A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN COMPUTER CONTROL AND
AUTOMATION**

2015

Contents

Abstract	i
Acknowledgements	ii
Abbreviations	iii
List of Figures	iv
List of Tables	v
1 Introduction	1
1.1 Background	1
1.2 Motivation	2
1.3 Objectives	3
1.4 Scope	4
1.5 Layout	5
2 Literature Review	7
2.1 Sentiment Analysis	7
2.1.1 Subjective and objective Information Classification	8
2.1.2 Subjective Emotional Polarity Information Classification	8
2.2 Term Weighting Methods	9
2.3 Word Vectors Technology	11
2.4 Statistical Language models	12
2.5 Classification Technologies	13
2.5.1 Support Vector Machine	13
2.5.2 Cross Validation	17

3	Term Weighting Schemes for Sentiment Analysis	20
3.1	Bag of words Model	20
3.2	Unsupervised Term Weighting Schemes	22
3.2.1	Average Weighting Scheme	22
3.2.2	TF-IDF Weighting Scheme	23
3.3	Supervised Term Weighting Schemes	23
3.3.1	Log-Count Ratio Weighting Scheme	24
3.3.2	Odds Ratio Scheme Weighting Scheme	25
3.3.3	Weighted Frequency and Odds Weighting Scheme	26
4	Document Representation Learning for Sentiment Analysis	28
4.1	Word Vectors	28
4.2	Combination of Unsupervised Term Weighting Schemes and Word Vectors	31
4.2.1	Average Weighting Scheme and Word Vectors	31
4.2.2	TF-IDF Weighting Scheme and Word Vectors	33
4.3	Combination of Supervised Term Weighting Schemes and Word Vectors	34
4.3.1	LCR Weighting Scheme and Word Vectors	34
4.3.2	OR Weighting Scheme and Word Vectors	36
4.3.3	WFO Weighting Scheme and Word Vectors	37
4.4	Combination of Supervised Term Weighting Schemes and Binary Vectors	38
5	Experimental evaluation	42
5.1	Experiment Setup	42
5.1.1	Implementation	42
5.1.2	Datasets and Word Vectors	44
5.2	Experiment Results and Analysis	45
5.2.1	Sws-b-w2v performs better on long-length datasets	45
5.2.2	Word Vectors based methods perform better with super- vised schemes	47
6	Conclusion and perspectives	50

6.1	Summary	50
6.2	Future Work	51
	Bibliography	52
	Appendix A	58

Abstract

Term weighting schemes are widely used in text mining tasks and supervised term weighting schemes have better performances on sentiment analysis task because the available labels of training documents make the learned model more discriminative. In this thesis, based on bag of words model, we introduced three supervised term weighting schemes and have shown their effectiveness for sentiment analysis in experiments. We also introduced the advanced word vectors technology and used the cosine similarity technique to measure intrinsic relationship between words to overcome the data sparsity problem.

Based on term weighting schemes and word vectors technology, we proposed two kinds of ideas to utilize word vectors in sentiment analysis systems. The first idea lies that we combined word vectors and our introduced term weighting schemes by vector multiplication operation to generate effective document feature vectors. The second one is that, we applied these introduced supervised weighting schemes on bag of words models where binary term frequencies are the features and word vectors are used as a measure to correlate unknown test document words with training document words and predict the weights of unknown testing words.

Our experiment results show supervised term weighting schemes and the intrinsic information among words discovered by word vectors can really improve the performance of sentiment analysis system jointly. Our methods outperform the state of the art methods on long-length document datasets and have competitive performances on short-length document datasets.

Acknowledgements

The whole work proposed in this thesis cannot be completed without the professional instructions and support from my supervisor, Associate Professor, Mao Kezhi. His tireless guidance has helped me grow a lot throughout my all Masters study. I greatly appreciate his patience and kindness. Here, I want to express my sincere thanks to him.

My senior, PhD student Zhao Rui also helped me a lot in my research work. He gave me many study materials and precious resources. His advices have inspired me a lot. The academic communication experience with him in the lab was so precious.

There are also many others who have given me a hand during my study. Among them, especially I would like to thank Angel Huang, Xu Shipeng and Amir for their advices and accompany.

For my beloved family standing by me all the time, I will always be grateful and pray for them.

Abbreviations

NLP: natural language processing

SNS: social network sites

BoW: bag of words

TF-IDF: Term frequency-inverse document frequency

CBOW: Continuous Bag-of-Words Model

Skip-gram: Continuous Skip-gram Model

IR: Information Retrieval

SVM: Support vector machine

VC dimension: Vapnik-Chervonenkis dimension

SRM: structural risk minimization

SMO: Sequential Minimal Optimization

CV: Cross Validation

LOOCV: leave-one-out cross-validation

k-CV: k-fold cross validation

AWS: Average Weighting Scheme

LCR: Log Count Ratio

OR: Odds ratio

WFO: Weighted Frequency and Odds

w2v-average: Average Weighting Scheme and Word Vectors Method

w2v-tfidf: TF-IDF Weighting Scheme and Word Vectors Method

w2v-LCR: LCR Weighting Scheme and Word Vectors Method

w2v-OR: OR Weighting Scheme and Word Vectors Method

w2v-WFO: WFO Weighting Scheme and Word Vectors Method

Uws-w2v: Unsupervised Weighting Schemes and Word Vectors Methods

Sws-w2v: Supervised Weighting Schemes and Word Vectors Methods

Sws-b-w2v: Supervised Weighting Schemes and Binary Vectors Methods Expanded by Word Vectors

List of Figures

1.1	Thesis Layout	5
2.1	SVM for linearly separable problems	15
2.2	SVM for linearly non-separable problems	16
3.1	LCR Function	25
3.2	Comparison between LCR and OR	26
3.3	WFO functions	27
4.1	Cosine Similarity of Words	29
4.2	Cosine Similarity of Phrases	31
4.3	Word and Vector Mapping	32
4.4	Average Weighting Scheme and Word Vectors Method	33
4.5	TF-IDF Weighting Scheme and Word Vectors Method	34
4.6	LCR and Word Vectors Method	36
4.7	OR and Word Vectors Method	37
4.8	WFO and Word Vectors Method	38
4.9	LCR/OR and Binary Vectors Method	40
4.10	WFO and Binary Vectors Method	40
5.1	Experiment Implementation	43
A.1	Project Screenshot	58
A.2	Similarity Words with Distances	59
A.3	Discriminating Words with Weights	60

List of Tables

3.1	Document Set	21
3.2	Vocabulary and Mapping Dictionary	21
5.1	Datasets Summary	44
5.2	Word Vectors File	45
5.3	Results of Sws-b-w2v methods against baselines	46
5.4	Results of Uws-w2v and Sws-w2v methods against baselines	48

Chapter 1

Introduction

1.1 Background

With the rapid growth of social media, e.g., blogs, reviews, tweets, in Web 2.0 era, a huge volume of digital data with subjective feelings are generated by netizens. Furthermore, smart phones become integral to peoples daily life, opinionated data are produced all the time and can be obtained easily and quickly. Such huge data stream seems trivial and complex, but there exists the enormous potential values hidden within it. Understanding peoples emotions within these Big Data and the prediction of the public opinion give new chances for organizations and firms to determine their wise development strategy. Sentiment analysis is a powerful natural language processing (NLP) method to give solutions for such problems. Via analysing these data, users can expand their range of options and find their real interests; enterprises can understand the psychological needs of consumers which is helpful to make the active responses and enhance their brand benefits; governmental authorities can understand the views of the people and grasp the psychology of people to make public opinion monitoring and handle social problems properly. On the other hand, nowadays, E-commerce is more and more popular in our daily lives. The current recommendation system is not powerful, which is time-consuming and inconvenient. Sentiment analysis defines a new way to deliver online advertisements. Customers do online shopping and give product

reviews as well as sharing their tendencies and preferences on social network sites (SNS), e.g., Facebook, Twitter. If enterprises can understand what their customers are thinking, they can show the most appropriate advertisement to the right customers and achieve maximum benefits. Meanwhile, customers can also benefit from it as they will find your desired things more easily. There is no doubt that sentiment analysis gives a good and simple solution for such situations.

Sentiment analysis is usually based on machine learning and data mining technologies which have gained widespread attentions and developed greatly since 2010. Under the Moore’s Law which reveals the development speed of semiconductor technology, computer hardware performances have been improved significantly in the past years. High performance computing and cloud computing accelerate the development of NLP technologies which are more easily to be adopted in practical applications. Therefore, more and more research efforts have been done on Sentiment Analysis.

1.2 Motivation

Baseline classification methods for sentiment analysis usually including training a linear model over bag of words (BoW) representations of documents, where a document is converted into a numerical vector related to the occurrences of its tokens. Given certain documents, some tokens are more meaningful than others. Intuitively, it will achieve good classification performances by assigning larger weights to discriminative tokens and smaller weights to non-discriminative ones during training. Term frequency-inverse document frequency (TF-IDF) [1] is a commonly used unsupervised term weighting scheme technique. As in most sentiment analysis tasks, opinion labels of training documents are known in advanced, supervised term weighting schemes are more useful for practical applications [2]. How to design such weight schemes for sentiment analysis is still a popular research topic. However, both the unsupervised and supervised term weighting schemes are based on numerical statistic methods, where the intrinsic relationship between tokens is missing. The rapidly evolving Deep Learning technologies provide a new view to understand the intrinsic information among

words. In 2013, Mikolov, et al.[3] proposed efficient methods to compute Word Vectors. Such word representations can reveal the relationship between words. As a result, it will make sense by combining supervised term weighting schemes and these intrinsic information contained in words to achieve a good sentiment understanding for unstructured text.

1.3 Objectives

The target of this research is developing data mining methods by combining term weighting schemes and Word Vector technologies to build a competitive sentiment analysis system.

The task including the following parts:

- Seeking suitable supervised term weighting schemes on BoW model for sentiment analysis;
- Combining the vector representations of words gained from Word Vector tools and term weighting schemes to generate new document feature vectors for sentiment analysis;
- Implementing designed sentiment analysis methods on different benchmark datasets;
- Comparing performances and analysing reasons while adopting different methods.

The major contributions of this thesis are: three supervised term weighting schemes are designed for sentiment analysis and two kinds of ideas are proposed to use Word Vectors for document feature learning. It provides a new viewpoint for further NLP research to utilize the intrinsic relationship between words. Previous NLP classification methods fail to deal with unknown test set words that do not occur in training set and just abandon them. However, such unseen words in testing documents is also the basis for sentiment predication. Such unknown words can be linked to training words based on intrinsic information among words,

which will make the analysis system more powerful and robust. Our methods use Word Vectors to expand the BoW representation model where binary counts of tokens that occur in the document are used as features. And the achieved results of our methods are close to or even slightly better than the one of state of the art sentiment analysis methods.

1.4 Scope

Sentiment analysis tasks normally consist of three following parts: data collection and cleaning, feature learning, model training and evaluation. Because the feature learning process is the most important and challenging part among them, this paper focuses on the feature learning process. Our works focus on designing effective methods to learn feature vectors for documents representation. The other two processes will be only briefly discussed in the experiment chapter, which just adopt those general methods. For feature learning, two unsupervised term weighting schemes and three supervised term weighting schemes are discussed in detail within this paper. A straightforward graphical way will be exploited in this paper to prove the feasibility of each supervised term weighting scheme. Furthermore, we will compare the performances of different weighting schemes.

The second contribution of this thesis is combining those term weighting schemes and Word Vectors to get new document feature vectors. Each weighting scheme is fine-tuned to incorporate it with Word Vectors optimally. This thesis will show that our methods can get good results (the state of the art for certain dataset) by using Word Vectors and our designed supervised term weighting schemes under BoW model where binary counts of document tokens are treated as learned features. Training large-corpus-based Word Vectors is beyond the scope of this article and it has already been a mature technique[3]. We will directly use the Word Vectors file released by Google¹. Practically, performances on different datasets are affected to some extent by the trained Word Vectors. For example, if some words in certain dataset are not included in the vectors file, it may cause some errors.

¹<https://code.google.com/p/word2vec/>

The primary contribution in this paper is designing supervised term weighting schemes and combining them with Word Vectors to generate effective document feature vectors or expanding traditional classification methods for sentiment analysis systems. Methods achieved satisfactory performances and at the same time provided some instructive significances for future research.

1.5 Layout

The thesis is arranged by the following layout:

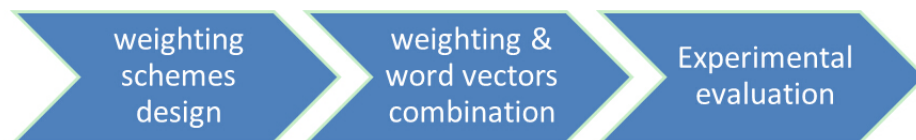


Figure 1.1: Thesis Layout

Chapter 1 gives an overview of the general concepts of the relative backgrounds as well as the main concerns and contributions of this paper.

Then related work is stated in Chapter 2. It covers sentiment analysis , term weighting methods, Word Vectors, Language models and classification technologies.

In the Chapter 3 we mainly discussed two unsupervised term weighting schemes and three supervised term weighting schemes.

Chapter 4 is focused on introducing two kinds of ideas to utilize Word Vectors in sentiment analysis. One is combining Word Vectors and term weighting schemes to get meaningful document feature vectors. The other one is that applying supervised term weighting schemes in BoW models with binary term frequency features, which is expanded by Word Vectors.

Experimental evaluation is proposed in Chapter 5. It introduces the complete processes of normal sentiment analysis tasks. Performance comparisons and methods

analysis are also shown in this chapter.

Finally, in Chapter 6 it comes to the conclusion and summary of the paper.

Chapter 2

Literature Review

In this chapter, related theories and technologies of our sentiment analysis research will be introduced. It includes the sentiment analysis, term weighting methods, word vectors technology, statistical language models and classification technologies.

2.1 Sentiment Analysis

Sentiment analysis (also known as opinion mining) is a simple but powerful natural language processing (NLP) method to identify and extract emotion information from documents. Some text documents (typically user reviews) will express authors' options about certain movies, events or goods. Opinions are important to human activities and are key effects of our behaviors. To a great extent, the decisions we make, beliefs and perceptions of the real world are based on how others think and assess the world [4]. Since pioneer works [5] [6] were published in early 2000, sentiment analysis has been developed into one of the most active and fascinating research areas in NLP[4]. Normally, sentiment analysis includes two tasks: subjective and objective information classification, subjective emotional polarity information classification [7].

2.1.1 Subjective and objective Information Classification

The research object of emotional tendency study is subjective texts. Such texts contain the subjective feelings of individuals, organizations, etc. as well as their views on things. Conversely, objective texts describe the fact or truth about the phenomenon. And in people's daily terms, subjective perception and objective description are often mixed together. This will cause errors in the judgement about emotional tendency of texts. Therefore, it is very important to classify objective and subjective information from given texts.

Some scholars treat subjective and objective text classification as a binary classification task which can be classified by a classifier. The key parts of such methods are the classifier design and feature selection. Yu, et al. [8] used words as features and classified full-length objective and subjective text by Naive Bayes classifier. Yao, et al. [9] chose some special exceptions rather than words to distinguish subjective and objective texts, such as punctuations, personal pronouns, digital angles and so on. And Pang, et al. [10] did subjective and objective analysis on sentences by graph-based classification method. In addition, some research focused on building emotional templates to identify the subjectivity and objectivity of emotional texts.

2.1.2 Subjective Emotional Polarity Information Classification

To judge whether a text is positive or negative, researchers usually analyze subjective information within subjective texts and employ binary classification methods. Subjective emotional polarity information classification is a further subdivision after subjective and objective information classification. Barbosa, et al. [11] used two-step method to implement sentiment analysis: the first one is to distinguish subjective and objective sentences and the second one is to identify positive and negative sentences. Currently, there are two mainly research ideas for text sentiment analysis: based on sentiment words dictionary or machine learning classification methods. The former one is using emotion dictionary or the combi-

nation of evaluation units to obtain the final emotional polarity of given text. The latter one (machine learning based) is to select the appropriate characteristics to represent texts and identify the emotional polarity of texts by learning models on such features. In this thesis, only the machine learning based sentiment analysis methods will be discussed.

2.2 Term Weighting Methods

The representation of documents is the most important part of text mining tasks. Vector Space Model (VSM) is the common method to represent documents, which turns the documents into numerical feature vectors. For example, in BoW model, a special case of VSM, each document is represented by a vector of the occurrences of terms. The value of each vector element is normally known as the term weight. Term weighting scheme is the key component of document representation by VSM. Term weights reflect the significance and contribution of terms with respect to the documents in many text mining tasks, such as sentiment analysis, information retrieval (IR), text classification and so on. An effective weighting scheme can boost the performances for text mining tasks.

Term weighting scheme technologies have been developed in IR research field during many years. Salton, et al. [12] first proposed that term frequency, normalization and inverse document frequency are the three main factors for term weighting methods. It is the fundamental theory for the subsequent studies of different term weighting schemes. Then many term weighting methods have been developed for IR. TF-IDF is one of the most widely used term weighting schemes. Robertson, et al. [13] proposed TF-IDF's famous variant named BM25 which is a probabilistic model for IR. It is a very effective weighting method for IR tasks. Recently, term weighting technologies continue to evolve and be improved for IR [14] [15].

Text classification (sentiment analysis) is similar to IR in some respects. Recent research has managed to solve sentiment analysis tasks by using term weighting methods from IR [16] [17]. It is proved that those term weighting methods

are effective and have the state of the art accuracies. However, Paltoglou, et al. [17] stated that such methods are not sufficiently intuitive as they only calculate the general distribution of document words excluding any additional information about class preference. In most cases, sentiment analysis can be treated as supervised learning tasks where training labels are already available. There are sound reasons for using such known information (labels) to learn better term weights. Works on sentiment analysis have mainly focused on supervised learning techniques [18]. Debole, et al. [19] proposed a supervised term weighting scheme, where the IDF portion of TF-IDF classic scheme is replaced by three term selection functions (information gain, chi-square, and gain ratio). However, three supervised term weighting methods in [19] have not a consistently superior to TF-IDF. Later, Soucy, et al. [20] proposed a new term weighting scheme named ConfWeight. The method is based on statistical confidence intervals. The results reported in [20] showed that ConfWeight had better performances than TF-IDF on several document datasets. Lan, et al. [21] introduced $tf \cdot rf$ supervised term weighting scheme which aims to improve the terms discriminating power for text classification. The intuitive idea of this method is that the more concentrated a high frequency term is in the positive class than in the negative one, the more contributions it makes in selecting the positive samples from the negative samples. It is reported that $tf \cdot rf$ consistently outperformed TF-IDF and other supervised term weighting methods. Recently, Deng, et al. [22] proposed a supervised term weighting scheme. In this method, each term weight is computed by the product of two elements: $ITD \cdot ITS$. The experimental results showed this method produced the excellent accuracy on two certain datasets. Kim, et al. [23] provided a simple but novel supervised weighting scheme by adjusting TF in TF-IDF. The results show it is robust and performs well on both snippets and longer documents. Wang, et al. [24] proposed the excellent binary Naive Bayes SVM method which takes log-count ratios as features. This method achieves state of the art performances on many benchmarks.

2.3 Word Vectors Technology

In NLP tasks, we need to convert natural language words into mathematics expressions which can be addressed by machine learning algorithm. Word Vector is an effective solution to fulfil such tasks. One-hot representation is the simplest Word Vector method: setting the length of feature vector equal to the size of the vocabulary and only a vector component is on (all the rests are 0). The location of this component is equal to the index of this word in the vocabulary. One-hot representation has an obvious drawback: it will suffer from data sparsity (or named curse of dimensionality), namely, for words that are rare in the labeled training data, their corresponding model parameters will be poorly estimated [25]. Furthermore, it is not able to represent word similarity. To solve these defects, Hinton proposed the distributed representation of words [26]. This method projects every word to a fixed short length vector (with respect to one-hot representation), so all these word vectors are in the same vector space, which can be viewed as space points and the point distances can be used to measure the (semantic or accident) similarity. Moreover, distributed representation is more robust and dense because word vectors have more no-zero components while one-hot representation word vectors only have one no-zero component.

In contrast to term weighting scheme technologies for generating document feature vectors, researchers focus on unsupervised methods for learning word vectors by inducing word representations over large unlabeled corpora [25]. One common approach is to use clustering (hierarchical) which tries to reduce the vocabulary size of one-hot representation vectors [27] [28]. Others are based on Neural language models, which generate dense real-valued low-dimensional word embeddings using unsupervised methods [29] [30]. Mikolov, et al. [31] [3] proposed effective methods to learn word vectors from document corpus by unsupervised methods. They introduced two architectures: Continuous Bag-of-Words Model (CBOW) and Continuous Skip-gram Model (Skip-gram) [32]. Each architecture has three layers (input, projection, output). CBOW is used for predicting current word vector based on word vectors of its context words while Skip-gram is used for predicting word vectors of context words based on current word vector. These two architectures (models) can be trained by two algorithms: Hierarchical

Softmax (works well for rare words) [31] and Negative Sampling (works well for common words) [3]. The common application of Word Vectors is for machine translation. Mikolov and his partners used Word Vectors to find the linear transformation which maps one language to another one. The method can achieve almost 90% precision for translation of words between English and Spanish [31].

2.4 Statistical Language models

Statistical Language Model (SLM) is generated from statistic-based NLP system research. SLM is the foundations of all NLP tasks, which is widely used in speech recognition, machine translation and IR. Normally, SLM is used to calculate the probability that a sentence occurs based on some corpus. Suppose that a sentence W is made up of k sequential words: w_1, w_2, \dots, w_k , then the probability for this sentence can be calculated by the following SLM formula [33]:

$$p(w_1^k) = p(w_1) \cdot p(w_2|w_1) \cdot p(w_3|w_1^2) \cdots p(w_k|w_1^{k-1}) \quad (2.1)$$

where w_1^{k-1} represents the string $w_1 w_2 \cdots w_{k-1}$.

In equation 2.1, those conditional probabilities means that the occurrence of current word is based on all the previous words. In practical, it can be assumed that the occurrence of one word is only effected by a fixed length of words before it. So in N-gram language model, a word only depends on the $n-1$ previous words (known as Markov assumption) [33]. Mathematically, conditional probabilities in equation 2.1 can be replaced as following:

$$p(w_k|w_1^{k-1}) = p(w_k|w_{k-n+1}^{k-1}) \quad (2.2)$$

where w_{k-n+1}^{k-1} represents the string $w_{k-n+1} w_{k-n+2} \cdots w_{k-1}$.

Smoothing techniques are often used in N-gram models, because such models will suffer from zero-probability problem (rare words counts) caused by data sparsity [34]. Bengio, et al. [35] proposed the neural probabilistic language model whose advantage is that the similarity between words can be calculated and it has build-in smoothing functions. Recently, Mikolov, et al. [36] proposed the more advanced Recurrent Neural Network based Language model which can make the

best use of context information. Recurrent Neural Network based language models significantly outperform N-grams and have the state of the art performances for statistical Language modeling.

2.5 Classification Technologies

2.5.1 Support Vector Machine

Support Vector Machine (SVM) is the most famous classifier since it is proposed by Cortes and Vapnik in 1995 [37]. It has many unique advantages for solving small sample, nonlinear and high dimensional pattern recognition problems. SVM can also be used in other machine learning problems such as function fitting.

SVM is based on two statistical learning theories: Vapnik-Chervonenkis (VC) dimension theory and structural risk minimization (SRM) principle. VC dimension is used to measure the complexity of a certain problem. The higher VC dimension the problem has, the more complex the problem is. Because what SVM focuses on is VC dimension rather than the dimension of the samples, SVM is very suitable for solving text classification problems (normally features are in high dimension). The essence of Machine Learning is building an approximate model to approach the true model of the problem. This approximate model is called a hypothesis. However, there is no doubt that the true model can not be known (if we know it, machine learning makes no sense, because we can directly solve the problem using this real model). As a result, we are not able to measure the difference between our hypothesis and the real model. This error between computed solution and real one is called risk (more strictly speaking, the cumulation of error is called risk). But once we assume a hypothesis (a classifier), we can calculate the difference between the classification results obtained by our classifier and the actual labels of the sample data (training data is labeled in advance). This difference is called the empirical risk $R_{emp}(\mathbf{w})$. Previous machine learning methods try to minimize this empirical risk and set it as the sole optimization target, but it has an obvious defect that many classification classifiers can be easily reached 100% accuracy on training sample set while their performances on real classifi-

cation tasks are very pool, which is called poor generalization or overfitting. For this case, the classifier is sufficiently complex (with high VC dimension), it can accurately distinguish every sample, but unknown data which are outside of the training samples will be almost misclassified. Actually, empirical risk minimization principle works well only if empirical risk is indeed close to (or consistent with) the real risk. However, the number of samples is a drop in the bucket with respect to the number of text to be classified in the real world, so empirical risk minimization principle only can make sure that there are no errors on these small percentage training samples, of course, it can not guarantee that there are also no errors on a greater proportion of unknown texts.

To compensate for this disadvantage, statistical learning theory introduces the SRM principle. It indicates that the true risk should include two parts: one is the empirical risk which represents the errors of this classifier on the given samples; the second one is called VC confidence which represents how much the classification results on unknown texts by this classifier can be trusted. Obviously, VC confidence can not be accurately calculated while only the rank of the estimated values can be computed. So we can only calculate the upper bound of the sum risk (called generalization error bounds) rather than the exact risk value. VC confidence is associated with two factors: the first one is the number of samples and it is apparent that the larger the number of samples is, the more correct the classification results on unknown texts is and the smaller the VC confidence value is; the second one is the VC dimension of the classifier and higher VC dimension will result in poorer generalization and the VC confidence value will become bigger. SRM principle provides a trade-off between hypothesis space complexity (the VC dimension of approximating functions) and the quality of fitting the training data (empirical error). Generalization error bound is defined as formula 2.3:

$$R(\mathbf{w}) \leq R_{emp}(\mathbf{w}) + \phi(n/h) \quad (2.3)$$

where $R(\mathbf{w})$ is actual risk, $R_{emp}(\mathbf{w})$ is empirical risk and $\phi(n/h)$ is VC confidence.

SVM is a statistical learning algorithm that manages to minimize the structural risk (sum of empirical risk of VC confidence). It has an objective function and several constraints. Figure 2.1 shows a linearly separable problem solved by SVM. The star points represent negative samples which satisfy $\mathbf{w} \cdot \mathbf{x} + b \leq -1$, where \mathbf{w}

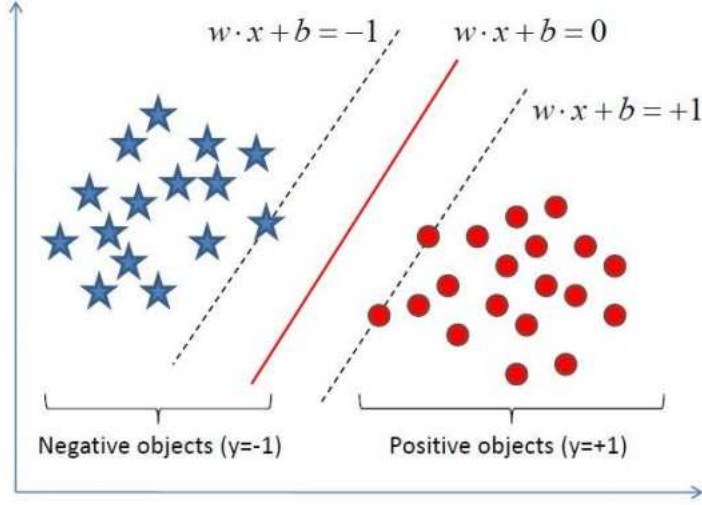


Figure 2.1: SVM for linearly separable problems

and b are parameters and \mathbf{x} represents point coordinate (can be high dimension). The circle points represent positive samples which satisfy $\mathbf{w} \cdot \mathbf{x} + b \geq 1$. The middle solid line is called hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$. So in order to classify the negative and positive well, we should make distances between points on the positive and negative sample dotted lines ($|\mathbf{w} \cdot \mathbf{x} + b| = 1$) and hyperplane as large as possible. Because once sample points on two dotted lines are separated, the other points on two sides are naturally separated. The distance between two dotted lines can be calculated by expression 2.4 (basic geometry knowledge):

$$\frac{2|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \quad (2.4)$$

Note that $|\mathbf{w} \cdot \mathbf{x} + b| = 1$ and $\|\mathbf{w}\|$ is the 2-norm of vector \mathbf{w} .

So the objective function is to maximize this geometric margin, which is equivalent to minimize $\frac{\|\mathbf{w}\|^2}{2}$ and the constraint condition is that all sample points must be in both sides of negative sample dotted border line ($\mathbf{w} \cdot \mathbf{x} + b = -1$) and positive samples dotted border line ($\mathbf{w} \cdot \mathbf{x} + b = +1$) rather than falling into the interval area. Mathematically, it can be written into a standard optimization model as shown in equation 2.5:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{\|\mathbf{w}\|^2}{2} \\ \text{s.t.} \quad & y_i(\mathbf{w}_i \cdot \mathbf{x}_i + b) \geq 1 \quad (i = 1, 2, \dots, N) \end{aligned} \quad (2.5)$$

where $y_i \in \{-1, +1\}$ represents the label of sample i and N is the total number of samples. For negative samples, we have $y_i = -1$ and $\mathbf{w}_i \cdot \mathbf{x}_i + b \leq -1$; for positive samples, we have $y_i = +1$ and $\mathbf{w}_i \cdot \mathbf{x}_i + b \geq 1$. So all samples should be subject to $y_i(\mathbf{w}_i \cdot \mathbf{x}_i + b) \geq 1$.

In fact, model 2.5 corresponds to empirical risk $R_{emp}(\mathbf{w})$, so in order to have better generalization, a slack variable should be introduced into this optimization model to minimize structural risk. In other words, we allow some sample points not satisfying the constraint condition but they should be punished. The improved optimization model is defined as follows:

$$\begin{aligned} \min_{\mathbf{w}, b, \epsilon} \quad & \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^N \epsilon_i \\ \text{s.t.} \quad & y_i(\mathbf{w}_i \cdot \mathbf{x}_i + b) \geq 1 - \epsilon_i \quad (i = 1, 2, \dots, N) \\ & \epsilon_i \geq 0 \quad (i = 1, 2, \dots, N) \end{aligned} \quad (2.6)$$

where ϵ_i represents the slack variable corresponding to sample i and C is called penalty factor which is a preset constant used for providing a trade-off between empirical risk and VC confidence to avoid overfitting or underfitting.

Optimization model 2.6 can be used to solve linearly separable problems. For linearly non-separable problems, SVM will normally introduce kernel functions to map non-separable input space into separable high dimension feature space ($\mathbf{x} \rightarrow \phi(\mathbf{x})$) and then use previous discussed model 2.6 to solve the problem. For

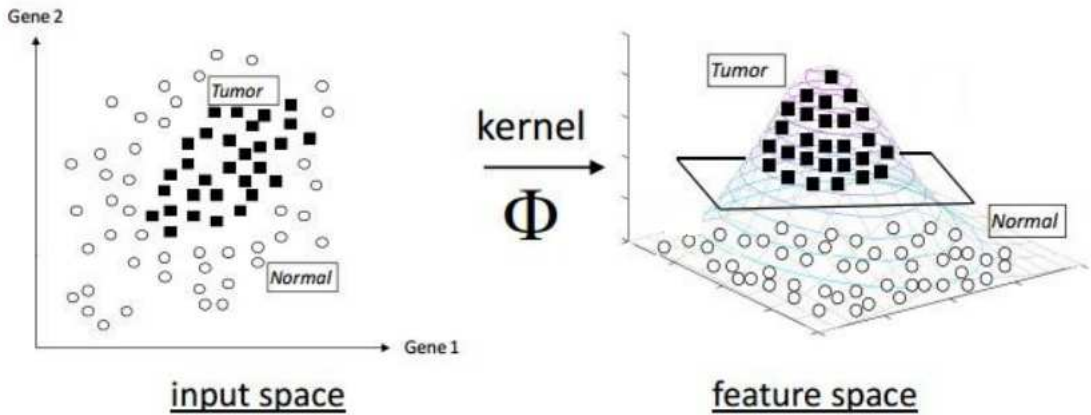


Figure 2.2: SVM for linearly non-separable problems

example, Figure 2.2 shows a linearly non-separable problem about tumor prediction. Tumor is affected by two genes and their relationships are shown in the left sub graph named input space, but tumor and normal samples are not linearly separable. So the input space samples are mapped into three dimension feature space by a kernel function. Based on that, two class samples can be linearly separated from each other by a 3D plane. As in this paper, we assume that sentiment analysis tasks are linearly separable, so we will only use the optimization model 2.6 and detailed knowledge about kernel functions are beyond the scope of this paper. Finally, if the optimization model is built, which is a convex quadratic programming problem, solutions can be directly calculated by Sequential Minimal Optimization (SMO) algorithm [38].

2.5.2 Cross Validation

Cross Validation (CV) is a model validation technique which assesses the generalization ability of an statistical analysis system or machine learning algorithm on unknown data set (independent of training data set). CV aims to solve over-fitting problems. In machine learning tasks, a data set is normally separated into three independent parts: training set, validation set and test set. The training set is used for model estimation and validation set is used for model selection (parameters or structures). The last part, test set, is used for examining the performance of the final selected optimal model (classifier). In this paper, we use CV techniques in two ways: one is for providing credible experimental results on unseparated benchmark datasets and the other is for model selection in SVM.

Normally, in order to employ CV, there are two basic requirements:

1. the ratio of training set should be great enough (generally greater than half);
2. training and validation sets should be uniform sampled.

Cross-validation techniques can be divided into three types: The first one is called double cross validation which is also know as 2-fold cross-validation (2-CV). This method randomly separates data set (excluding test set) into two equal-size

subsets and conducts two rounds of model training by the same classifier. In the first round, one subset is set as the training set and the other one as the validation set; in the second round, swap training set and validation set and train the model again. What we are really concerned about are the recognition accuracies on these two validation sets. However, 2-CV is not commonly used in real applications, because the training set has too few samples to represent the distribution of the whole data set, which will result in very low recognition accuracy on test data set. Moreover, the variability of separated subset in 2-CV is large so it often can not achieve the requirement that experiment processes must be able to be replicated.

The second one is called k-fold cross validation (k-CV) which extends the 2-CV method. K-CV separates data set (excluding test set) into k subsets and each subset will be set as validation set while the rest as the training set (k-1 folds). So k-cv will be repeated k times and each time a subset is selected as a validation set. The average recognition accuracy of these k experiments is the final result of k-fold. The advantage of this method is that all samples are treated as training and validation sets and each sample is validated once. In practical, 10-fold CV is widely used.

The third one is called leave-one-out cross-validation (LOOCV). Suppose that there are n samples in the data set (excluding test set), LOOCV is equivalent to n-CV. So in LOOCV method, each individual sample will be set as the validation set and the remaining n-1 samples will be treated as the training set. LOOCV has two advantages:

- For each validation round, almost all samples are used to train the model, so the distribution of the training set is very close to the one of population. The measured generalization ability is very credible. Therefore, when sample set of experimental data is small, LOOCV is preferred.
- there is no random factor affecting experimental data during the experiment, which makes the experiment replicated.

However, LOOCV also has an obvious disadvantage: highly computing-time consuming. Because the total number of models needed to be trained is equal to the

number of samples, when data set is quite large, it is difficult to employ LOOCV in real application.

After comprehensive comparison among these CVs, in this paper, we select k-fold CV in our experiments.

Chapter 3

Term Weighting Schemes for Sentiment Analysis

In this chapter, the Bag of words model will be firstly introduced, which is the foundation of all the sentiment analysis methods in this thesis. Then two simple unsupervised term weighting schemes (Average Scheme, TF-IDF) and three supervised term weighting schemes: Log-Count Ratio, Odds Ratio and Weighted Frequency Odds will be discussed in details.

3.1 Bag of words Model

The Bag of words model is a widely used effective method in NLP and IR classification tasks to give a numerical representation of a document. In BoW, a text is treated as a set of words ignoring the word order and grammars. Therefore, each word in the text is independent of others, which means the occurrence of a word has nothing to do with occurrences of other words. Term frequency (TF) represents the number of times a term occurs in a document. For document classification, Bow takes TFs as features to generate document feature vectors. As a result, text contents are converted into numerical feature vectors which can be addressed by machine learning algorithms further. Normally, documents will be loaded as file stream and the vocabulary of this corpus will be found on-line and

the token frequencies will be calculated. In the vocabulary, each term will include the word and its total count. Then a fixed integer id is assigned to each word in the vocabulary so that a dictionary mapping from words to integer indices is built. For each document $\#i$, TF of each term w is stored in $\mathbf{X}[i, j]$ where j represents the corresponding index of word w in the mapping dictionary. Without much surprise, such document vectors (each row in \mathbf{X}) will have the same dimension equal to vocabulary size regardless of the length of each document.

For example, suppose we have a small document set as shown in the Table 3.1:

Table 3.1: Document Set

Index	document
#1	Jack loves to eat apples. Lucy loves apples too.
#2	Jack also loves to drink orange juice.
#3	Lucy also loves to eat bananas.

Table 3.2: Vocabulary and Mapping Dictionary

Index(#j)	Term	Total TF
1	“Jack”	2
2	“loves”	4
3	“to”	3
4	“eat”	2
5	“apples”	2
6	“also”	2
7	“orange”	1
8	“juice”	1
9	“Lucy”	2
10	“too”	1
11	“drink”	1
12	“bananas”	1

we can easily get the vocabulary for the whole corpus as well as the total count of each term. Table 3.2 is the vocabulary and corresponding mapping dictionary (mapping order is random). In BoW, each document can be represented by a 12-entry vector:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 1 & 1 & 2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

where each column of the matrix refers to TF of the corresponding term in the mapping dictionary. For example, in the first row (which represents document #1), the first two elements are “1” and “2”. Based on the mapping dictionary, the first element corresponds to term “Jack”. The value equals to “1” shows “Jack” occurs in the document #1 one time. Similarly, the second term in mapping dictionary is “loves” so that the second column represents word “loves”. And “loves” appears in document #1 two times, so the second value in first row is “2”. It can be seen that such vector representation does not keep the order of the words in the original documents. In real applications, users often combine BoW and n-grams models into Bag of n-grams models, where term dictionary includes combinations of continuous words. For example, phrase “orange juice” will be a new column of \mathbf{X} .

3.2 Unsupervised Term Weighting Schemes

3.2.1 Average Weighting Scheme

The simplest term weighting scheme is Average Weighting Scheme (AWS). Intuitively, average weights are calculated as TFs divided by the document length, which is defined by equation 3.1:

$$AWS^{(i)} = \frac{\mathbf{f}^{(i)}}{N_w} \quad (3.1)$$

where $\mathbf{f}^{(i)}$ is the TF feature vector of document i and N_w is the total number of words in this document. Therefore, this weighting scheme can be regarded as a

L1-norm normalization operation.

3.2.2 TF-IDF Weighting Scheme

Besides AWS Scheme, Term Frequency Inverse Document Frequency (TF-IDF) Term Weighting Scheme is another widely used unsupervised weighting scheme for text classification. TF-IDF unsupervised weighting scheme is based on the obvious fact that some words occurring in many documents are less informative while those that occur only in a smaller portion of the corpus are more meaningful. TF-IDF includes two parts term frequency and inverse document frequency. The IDF part decreases the weight of terms that occur very frequently in the whole document set and increases the weight of terms that occur rarely so as to reflect how important a word is to a document in a corpus. In TF-IDF, the weight for token with index j in document i is normally computed as follows:

$$tfidf_{j,i} = tf_{j,i} \cdot idf_{j,i} = tf_{j,i} \cdot \log \frac{N_d}{df_j} \quad (3.2)$$

where $tf_{j,i}$ is the TF, $idf_{j,i}$ is the inverse document frequency, N_d is the total number of documents in the corpus, and df_j is the number of documents in which token with index j appears (i.e., for those documents $tf_j \neq 0$).

Taking the document set in Table 3.1 for example, TF-IDF weight of the word "apples" in document #1 can be calculated based on equation 3.2. From Table 3.2, we know TF of term "apples" is 2, total number of documents N_d is 3 and the number of documents including this term df_j is 1, so the resulting value is $tfidf_{5,1} = 2 \cdot \log \frac{3}{1}$. Intuitively, $tfidf_{j,i}$ values increase proportionally to the TF and inversely to the number of documents containing the corresponding token. Tokens with the highest TF-IDF values often characterize the document optimally.

3.3 Supervised Term Weighting Schemes

Different from previous unsupervised term weighting schemes, supervised term weighting schemes are able to utilize the label existed in the training corpus. Via

the introduction of label information, these weighting schemes aim to discover the discriminative information behind words

3.3.1 Log-Count Ratio Weighting Scheme

Log Count Ratio (LCR) is a simple supervised term weighting scheme based on prior knowledge about training data class labels. Normally, sentiment analysis task can be simplified into a kind of binary classification task (Negative against Positive or subjective against objective). In BoW, set $\mathbf{f}^{(i)} \in \mathbf{R}^{|V|}$ be the feature vector of training document i and its label is $y^{(i)} \in \{-1, 1\}$. V is the term vocabulary (feature set) of training documents and $\mathbf{f}_j^{(i)}$ represents TF of term V_j in training document i . Then define the sum feature vectors for positive and negative documents respectively:

$$\mathbf{F}_p = \boldsymbol{\alpha} + \sum_{i:y^{(i)}=1} \mathbf{f}^{(i)} \quad (3.3)$$

$$\mathbf{F}_n = \boldsymbol{\alpha} + \sum_{i:y^{(i)}=-1} \mathbf{f}^{(i)} \quad (3.4)$$

where $\boldsymbol{\alpha}$ is the smoothing parameter vector.

When one vocabulary word does not appear in all of the test documents, smoothing parameter vector can avoid zero components for unseen words in testing domain and normally it can be set to $\mathbf{1} \in \mathbf{R}^{|V|}$. With these class label statistical information, LCR is defined by following equation 3.5:

$$LCR = \log \left(\frac{\mathbf{F}_p / \|\mathbf{F}_p\|_1}{\mathbf{F}_n / \|\mathbf{F}_n\|_1} \right) \quad (3.5)$$

where $\|\mathbf{F}_p\|_1$ represents the L_1 norm of vector \mathbf{F}_p which is the sum of the absolute values of the components and here log is a vector operator.

Figure 3.1 shows relationship between LCR function values and TF in two contrary class documents (positive and negative). In order to simplify the illustration, we set \mathbf{F}_p and \mathbf{F}_n as scalars (that is TF of certain term) and normalize them into $(0, 1)$. Furthermore, it is also assumed $\mathbf{F}_p + \mathbf{F}_n = 1$. It can be seen that when TFs in these two are similar, the LCR weighting value is close to zero while if one part is dominant (much large than the other one) the weight will increase

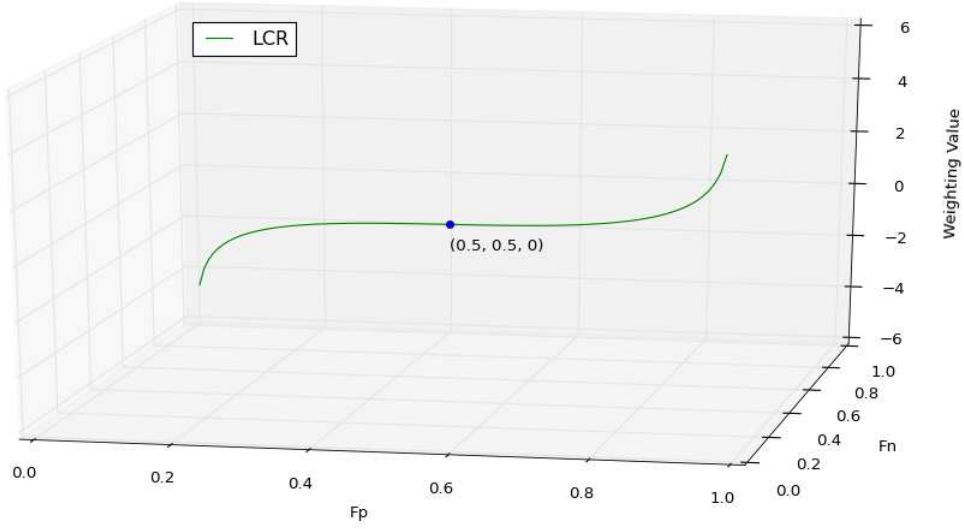


Figure 3.1: LCR Function

along the corresponding direction. In other words, as expected positive terms have larger weighting values as positive values and negative terms have smaller weighting ones as negative values.

3.3.2 Odds Ratio Scheme Weighting Scheme

Odds ratio (OR) is another commonly used technique in information retrieval, where the problem is to rank out documents according to their relevance for the positive class value using occurrences of different words as features [39]. Given training document feature vector $\mathbf{f}^{(i)} \in \mathbf{R}^{|V|}$, sum feature vectors \mathbf{F}_p and \mathbf{F}_n can be calculated by equations 3.3 and 3.4. Then they can be normalized to average feature vectors through two following equations:

$$\bar{\mathbf{F}}_p = \frac{\mathbf{F}_p}{\|\mathbf{F}_p\|_1} \quad (3.6)$$

$$\bar{\mathbf{F}}_n = \frac{\mathbf{F}_n}{\|\mathbf{F}_n\|_1} \quad (3.7)$$

where $\bar{\mathbf{F}}_p$ and $\bar{\mathbf{F}}_n$ are normalized feature vectors.

After that, we improved the OR formula using these two average feature vectors, which is proved more effective than the origin one in the sentiment analysis

experiments. Formula 3.8 shows how to compute OR weighting vectors in our methods.

$$\mathbf{OR} = \log \frac{\bar{\mathbf{F}}_p(1 - \bar{\mathbf{F}}_n)}{(1 - \bar{\mathbf{F}}_p)\bar{\mathbf{F}}_n} \quad (3.8)$$

Note that, \log is a vector operation.

Figure 3.2 shows a comparison between OR and LCR function. Note here is also simplified by the same way as is done in figure 3.2. The figure shows that

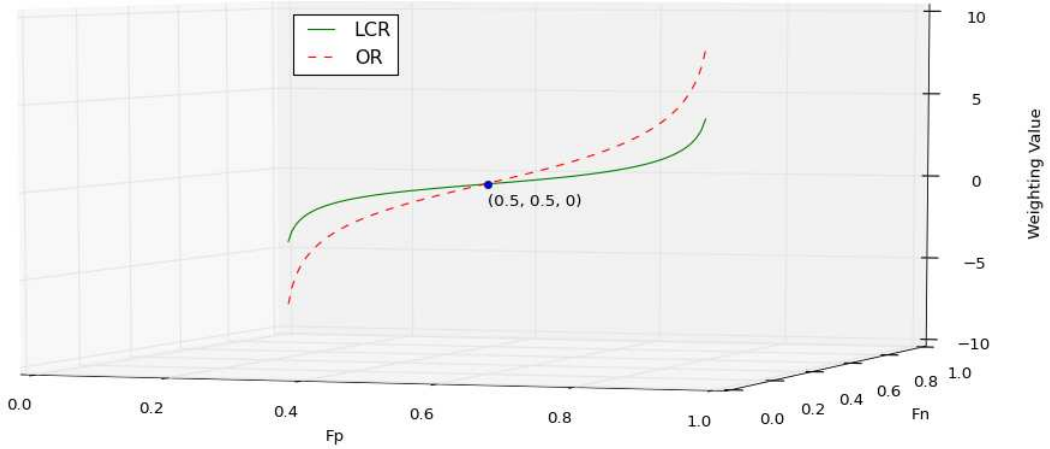


Figure 3.2: Comparison between LCR and OR

OR function curve has the similar trend to LCR function curve, which coincides with what is expected (contrary classes have extension directions). Moreover, OR function curve is steeper than LCR function curve, which implies it is more sensitive to TF changes. In practice, OR has a better performance for long length documents while LCR performs better for short length ones.

3.3.3 Weighted Frequency and Odds Weighting Scheme

Weighted Frequency and Odds (WFO) is a new supervised statistic methods proposed by Li et al. in 2009 [40]. Normally, good features will have higher document frequency as well as higher category ratio. But the relative importance should depend on specific applications. Therefore WFO combines document frequency and category ratio, and the corresponding importance for each one is tuned through

an hyper-parameter based on the application. The reported results show WFO has robust excellent performances on different data sets for text Categorization tasks. Given training document feature vector $\mathbf{f}^{(i)} \in \mathbf{R}^{|V|}$, average feature vectors $\bar{\mathbf{F}}_p$ and $\bar{\mathbf{F}}_n$ can be calculated by equations 3.3, 3.4, 3.6 and 3.7. For sentiment analysis, we modified the category ratio to get better performances, the improved WFO is defined by formula 3.9:

$$\mathbf{WFO}(\lambda) = \bar{\mathbf{F}}_p^\lambda \log \left(\frac{\bar{\mathbf{F}}_p}{\bar{\mathbf{F}}_n} \right)^{(1-\lambda)} \quad (3.9)$$

where λ is the adjustable parameter for tuning the weight between document frequency and category ratio. In our method, λ is between 0 and 0.1.

Figure 3.3 shows our WFO function under different λ values. Note here is simplified by the same way as is done in figure 3.1. It can be seen that λ values

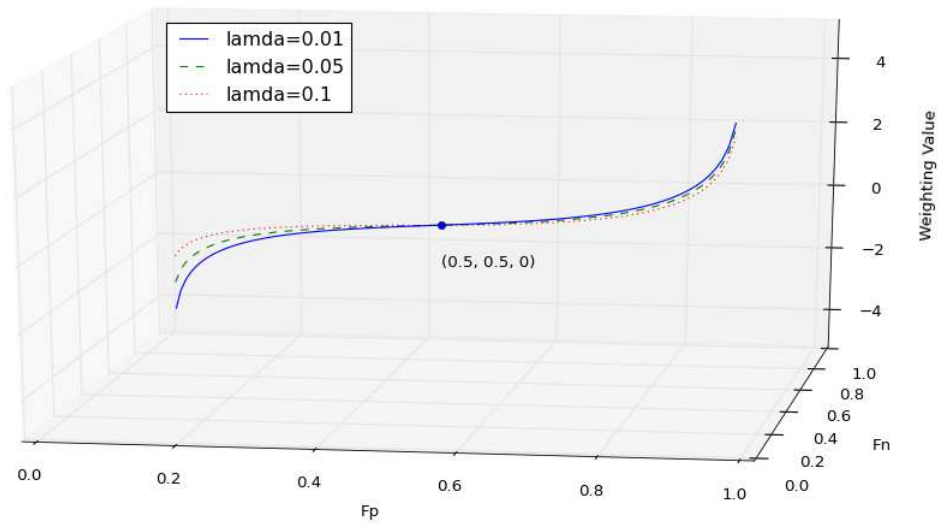


Figure 3.3: WFO functions

alter the slope of function curve to adapt with different data sets. Figure 3.3 only shows the intuitive feasibility of WFO for sentiment analysis tasks and the best λ value will vary from application to application.

Chapter 4

Document Representation Learning for Sentiment Analysis

In this chapter, our proposed methods for modeling documents in sentiment analysis will be introduced, which are based on combination of term weighting schemes and word vectors. Firstly, a brief introduction about word vector will be given because it is the theoretical foundation of our methods. Then it comes to two kinds of basic ideas to utilize word vectors for sentiment analysis. The first one is combining term weighting schemes and word vectors by multiplication operation and the second one is that word vectors are used to expand the BoW models with binary features.

4.1 Word Vectors

In 2013, Google released an open source NLP tool named word2vec¹ which provides an efficient implementation of CBOW and Skip-gram for computing word vectors. The word2vec tool takes a document corpus as input and generates word vectors. The resulting word vector file can be directly used as features for numbers of NLP and machine learning applications. The simplest usage is to find the most similar words for a certain word based on measures for vector distance.

¹word2vec open source project: <https://code.google.com/p/word2vec/>

Cosine similarity is a commonly used numerical metrics to measure the geometrical relationship between word vectors, which represents the semantic or linguistic similarity of the corresponding words. Given two n-dimensional word vectors W_1 and W_2 , Cosine Similarity between these two words can be defined by the cosine of the angle between two word vectors as shown in following equation 4.1:

$$CoSim(W_1, W_2) = \cos(\theta) = \frac{W_1 \cdot W_2}{\|W_1\| \|W_2\|} = \frac{\sum_{i=1}^n W_{1i} \times W_{2i}}{\sqrt{\sum_{i=1}^n (W_{1i})^2} \times \sqrt{\sum_{i=1}^n (W_{2i})^2}} \quad (4.1)$$

where CoSim denotes Cosine Similarity.

Cosine similarity values are between 0 and 1 for word vectors. If angle is 0 degree, the similarity is 1, which means vectors are very relevant to each other. Similarly, the greater angle means smaller cosine similarity. If vectors are orthogonal with 90 degrees, the similarity is 0 and two words are irrelevant. As word vectors obtained from word2vec tool are already normalized (length of vectors are 1.0), equation 4.1 can be simplified as dot product of unit vectors:

$$CoSim(\hat{W}_1, \hat{W}_2) = \cos(\theta) = \hat{W}_1 \cdot \hat{W}_2 = \sum_{i=1}^n \hat{W}_{1i} \times \hat{W}_{2i} \quad (4.2)$$

where \hat{W}_i is the unit vector in the direction of W_i ($i = 1, 2$).

Based on equation 4.2, the similarity between certain word and other words can be easily calculated and the most similar word to this word can be found by

word	Cosine distance
spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
russia	0.571507
germany	0.563291
catalonia	0.534176

Figure 4.1: Cosine Similarity of Words

ranking the result cosine values. Figure 4.1 shows the top-10 most similar words

for word "france", which is obtained by distance function in word2vec tool. It was also shown that word vectors have some linguistic regularities, for example, vector operations $\text{vector('man')} - \text{vector('king')} + \text{vector('queen')}$ is very close to vector('woman') . This analogy question (answering king is to queen as man is to -) can be converted into an optimization problem that aims to find the word vector W_h of a hidden word which is most similar to the word vector $W_{man} - W_{king} + W_{queen}$. As we discussed in the previous paragraphs, vector similarity can be measured by $CoSim()$ function, this optimization problem is defined by expression 4.3:

$$\arg \max_{W_h \in V} CoSim(W_h, W_{man} - W_{king} + W_{queen}) \quad (4.3)$$

where V is the word2vec vocabulary excluding the three known words. If W_h is a normalized unit vector (default in word2vec tool), expression 4.3 can be rewritten based on expression 4.1:

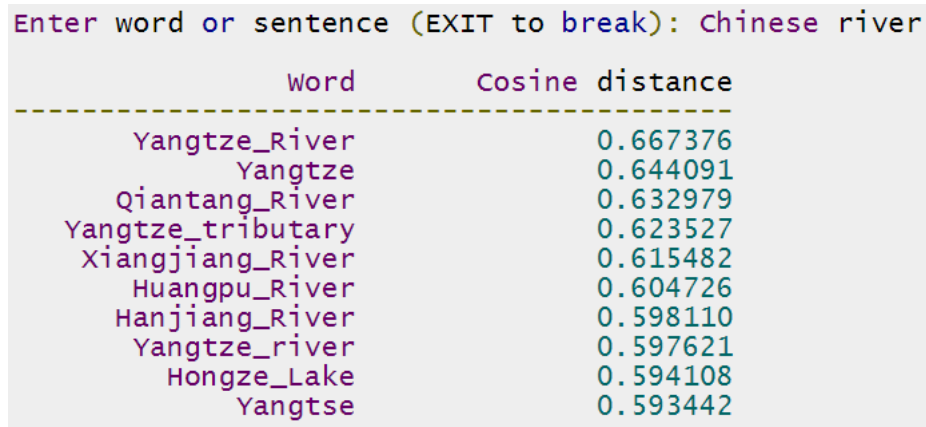
$$\arg \max_{W_h \in V} ((W_h \cdot (W_{man} - W_{king} + W_{queen}))) \quad (4.4)$$

As shown above, with the assistance of word vectors, an linguistics problem can be transformed to a mathematical operation. Sanjeev, et al. [41] provide a more rigorous explanation of why such corresponding relationship comes about. Similarly, this additive character of word vectors can be used to find the most relevant phrases of one phrase (two or more words). Figure 4.2 shows the top-10 most relevant phrases for phrase "Chinese river". The method involves adding word vectors $W_{Chinese}$ and W_{river} and seeking the most similar word vector to the resulting word vector. Mathematically, it can be expressed by expression 4.5:

$$\arg \max_{W_h \in V} CoSim(W_h, W_{Chinese} + W_{river}) \quad (4.5)$$

Therefore, word vector is able to represent the semantic and synaptic information behind words in a low-dimensional and dense space. As word vector arithmetic remains some semantic logic, it can be used to compute document feature vectors. In this paper, we directly adopted Google published pre-trained word vectors [3]. This file includes 300-dimensional word vectors for 3 million words and phrases, which is trained on Google News dataset (about 100 billion words). Although we

can get our word vectors by training models on our local data set, it is more time-consuming and is easily overfitting considering the small size of training corpus.



Enter word or sentence (EXIT to break): Chinese river

word	Cosine distance
Yangtze_River	0.667376
Yangtze	0.644091
Qiantang_River	0.632979
Yangtze_tributary	0.623527
Xiangjiang_River	0.615482
Huangpu_River	0.604726
Hanjiang_River	0.598110
Yangtze_river	0.597621
Hongze_Lake	0.594108
Yangtse	0.593442

Figure 4.2: Cosine Similarity of Phrases

4.2 Combination of Unsupervised Term Weighting Schemes and Word Vectors

This section is about document representation learning methods based on combining unsupervised term weighting schemes introduced in Chapter 3 and Word Vectors.

4.2.1 Average Weighting Scheme and Word Vectors

One challenge for sentiment analysis tasks is how to represent documents. We proposed to derive document vector from word vectors. The simplest idea is just using vector operations to average all the word vectors occurred in the document, which derives the Average Weighting Scheme and Word Vectors method (w2v-average). Documents vocabulary will be firstly built and w2v-average will extract the corresponding word vectors from the pre-trained word vectors file based on this vocabulary. In order to improve the query speed, two mapping dictionaries

are built to connect terms and word vectors. If one word occurs in the document, its index will be firstly found in first mapping dictionary and then the corresponding word vector will be searched based on this index in another mapping dictionary. The relationship between them is shown in Figure 4.3:

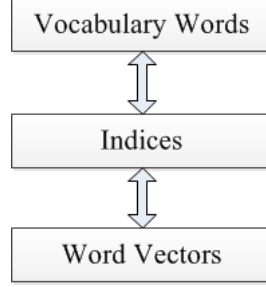


Figure 4.3: Word and Vector Mapping

Once word vectors are obtained, the w2v-average based document feature vector for document i can be calculated as following equation:

$$\mathbf{Docvec}^{(i)} = \sum_{1 \leq j \leq N_t} \frac{\mathbf{f}_j^{(i)}}{N_w} \cdot W_j \quad (4.6)$$

where $\mathbf{f}_j^{(i)}$ is the term frequency of the token with index j occurred in document i , N_w is the total number of words in this document, N_t is the number of tokens (each one is unique) in this document and as expected W_j represents the word vector of token with index j .

It can be easily seen that such document feature vectors will have the same dimension with word vectors regardless of the length of each document. For instance, if we have document set as shown in Table 3.1 and its statistical vocabulary and mapping dictionary as shown in Table 3.2. For document #1 ("Jack loves to eat apples. Lucy loves apples too"), $N_w = 9$, $N_t = 7$ and its document feature vector can be obtained based on equation 4.6: $\mathbf{Docvec}^{(1)} = \frac{1}{9} \cdot W["Jack"] + \frac{2}{9} \cdot W["loves"] + \frac{1}{9} \cdot W["to"] + \frac{1}{9} \cdot W["eat"] + \frac{2}{9} \cdot W["apples"] + \frac{1}{9} \cdot W["Lucy"] + \frac{1}{9} \cdot W["too"]$, where $W["x"]$ is the word vector.

Document feature vectors obtained by w2v-average method have different scales (although the dimensions are the same). So they should be normalized before training models on them. Flow Chart 4.4 shows the processes to calculate a

certain document feature vector by combining average weighting scheme and word vectors:

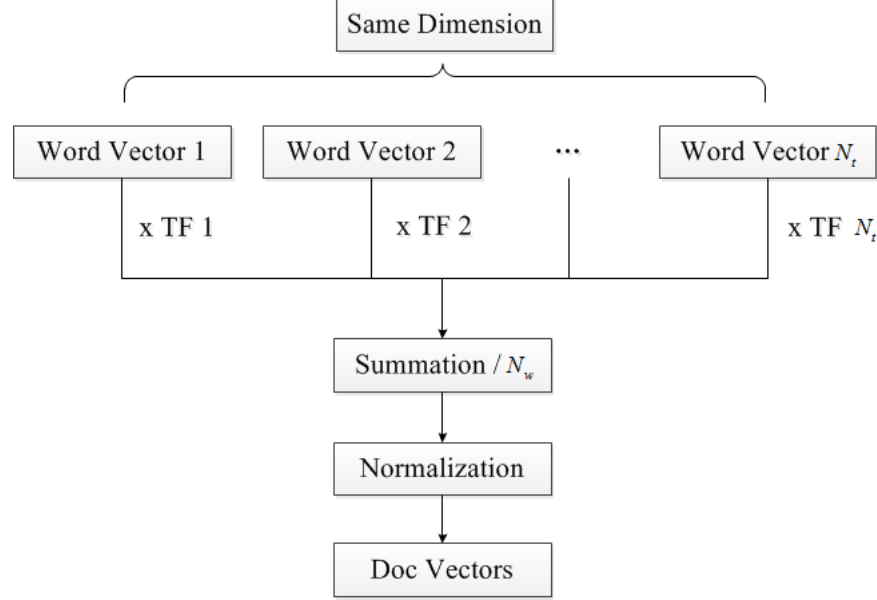


Figure 4.4: Average Weighting Scheme and Word Vectors Method

4.2.2 TF-IDF Weighting Scheme and Word Vectors

This subsection introduces the the document representation learning methods based on combining TF-IDF Weighting Scheme with Word Vectors (w2v-tfidf) for generating document feature vectors. For sentiment analysis tasks, in order to employ w2v-tfidf method, the BoW model should be firstly built on the corpus as discussed in Chapter 3.1. Then, TF-IDF weight for each term can be calculated using equation 3.2. Once TF-IDF weights are given, the w2v-tfidf based feature vector of document i is defined in equation 4.7:

$$Docvec^{(i)} = \sum_{1 \leq j \leq N_t} tfidf_{j,i} \cdot W_j \quad (4.7)$$

where N_t is the number of tokens (each one is unique) in this document and W_j is the word vector of token with index j .

Flow chat 4.5 summarizes the processes to calculate document feature vectors by TF-IDF Weighting scheme and word vectors Methods:

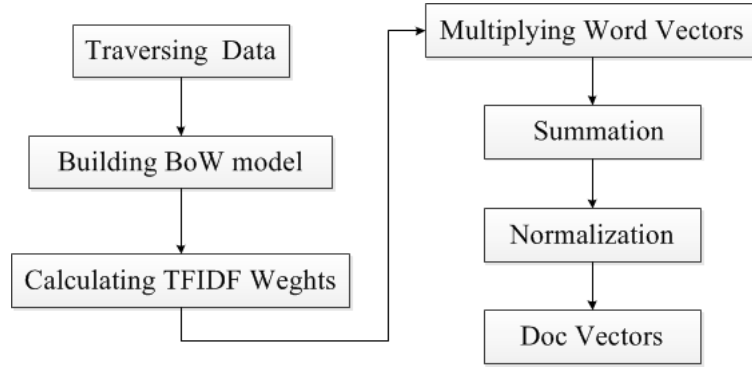


Figure 4.5: TF-IDF Weighting Scheme and Word Vectors Method

4.3 Combination of Supervised Term Weighting Schemes and Word Vectors

As in most sentiment analysis tasks, opinion labels of training documents are known in advanced. Supervised term weighting schemes are able to utilize these labels. So the next three subsections discuss about how to combine three proposed supervised term weighting schemes with word vectors.

4.3.1 LCR Weighting Scheme and Word Vectors

LCR Weighting Scheme and Word Vectors Method (w2v-LCR) is based on LCR term weighting scheme as defined in formula 3.5. W2v-LCR method firstly builds BoW model on training data set with class labels and then calculates the *LCR* weighting vector. The dimension of this weighting vector is equal to the size of training data set vocabulary. As a result, there is a unique certain weight for each term within training data vocabulary. To calculate the document feature vectors of training data set, w2v-LCR method just puts the word vector multiplied the word vector by the absolute value of the corresponding LCR weight and sums all the weighted vectors corresponding to words.

As we know that cosine similarity of word vectors learned by word2vec tool ranges from 0 to 1, which means such vectors are not completely opposite. The resulting feature vectors should remain this property. So in order to combine weight scheme and word vectors better, w2v-LCR transforms LCR weights into absolute values of themselves. And, to calculate the document feature vectors of test data set, there will be unknown words in test documents, which are outside of the training data vocabulary. In other words, such words do not have corresponding weights. Other conventional classification methods, such as BoW or TF-IDF based methods, directly abandon these unknown words. This is obviously unreasonable and will cause errors. When we human beings make a decision under a certain situation, we will refer to the given conditions as well as our previous experience. Due to a limited size of training data, the trained model has its limits. Word vectors are learned from huge volume of corpora data and they hold some intrinsic information among words which can be used to handle with unknown-words problem. So to calculate document feature vectors of test data set, w2v-LCR method puts the word vector multiplied by the corresponding LCR weight (absolute value) whenever this test document word is included in training vocabulary and then computes the sum of those weighted vectors and all the word vectors whose corresponding test words are beyond the training vocabulary. In other words, we set the weights of these unknown words equal to 1, because the class preference of those unknown words is unavailable. Mathematically, both training and test document feature vectors generated by w2v-LCR method can be calculated by a unified formula as follows:

$$Docvec^{(i)} = \sum_{1 \leq j \leq N_t} |\mathbf{LCR}_j| \cdot W_j + \sum_{k \in S(i)} W_k \quad (4.8)$$

where N_t represents the number of document terms which are included in the training vocabulary. $|\mathbf{LCR}_j|$ and W_j are the absolute value of corresponding LCR weight and word vector for the term with index j . $S(i)$ represents the unknown word set of document i , so for all the training documents $S(i) = \emptyset$.

Flow chart 4.6 summarizes the processes for generating document feature vectors by w2v-LCR method. Note that $\mathbf{LCR}_{[j]}$ means the corresponding LCR weight of document term with index j in mapping dictionary instead of the j th component of LCR weight vector. Normalization is a necessary process for achieving better

performance.

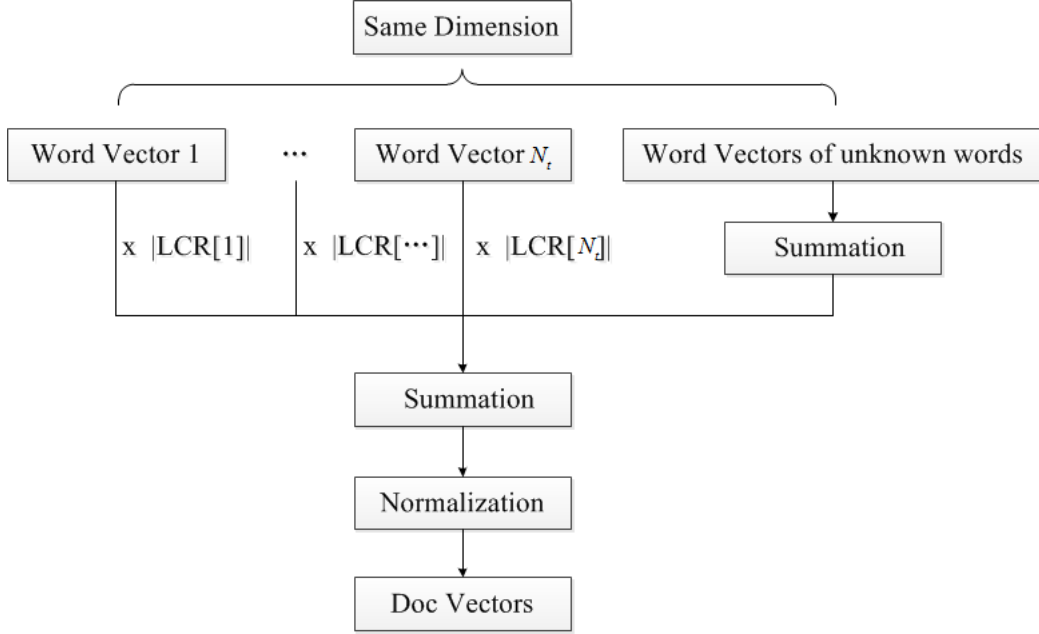


Figure 4.6: LCR and Word Vectors Method

4.3.2 OR Weighting Scheme and Word Vectors

OR Weighting Scheme and Word Vectors method (w2v-OR) is based on OR term weighting scheme as defined in Formula 3.8. As we discussed in the previous chapter, OR supervised term weighting scheme has similar good distinguishing ability with LCR scheme. So similarly, w2v-OR firstly computes the absolute values of OR weight vector obtained by Formula 3.8, and then combines this new weight vector and word vector in such way: for document terms among training vocabulary, the word vector is multiplied by the corresponding weight while for document terms beyond training vocabulary, the word vector is multiplied by 1 and the last step is calculating the sum of all those resulting vectors. So document feature vectors can be generated by w2v-OR method through the following formula:

$$Docvec^{(i)} = \sum_{1 \leq j \leq N_t} |OR_j| \cdot W_j + \sum_{k \in S(i)} W_k \quad (4.9)$$

where $|\mathbf{OR}_j|$ represent the absolute value of corresponding OR weight for the term with index j . $S(i)$ represents the unknown word set of document i .

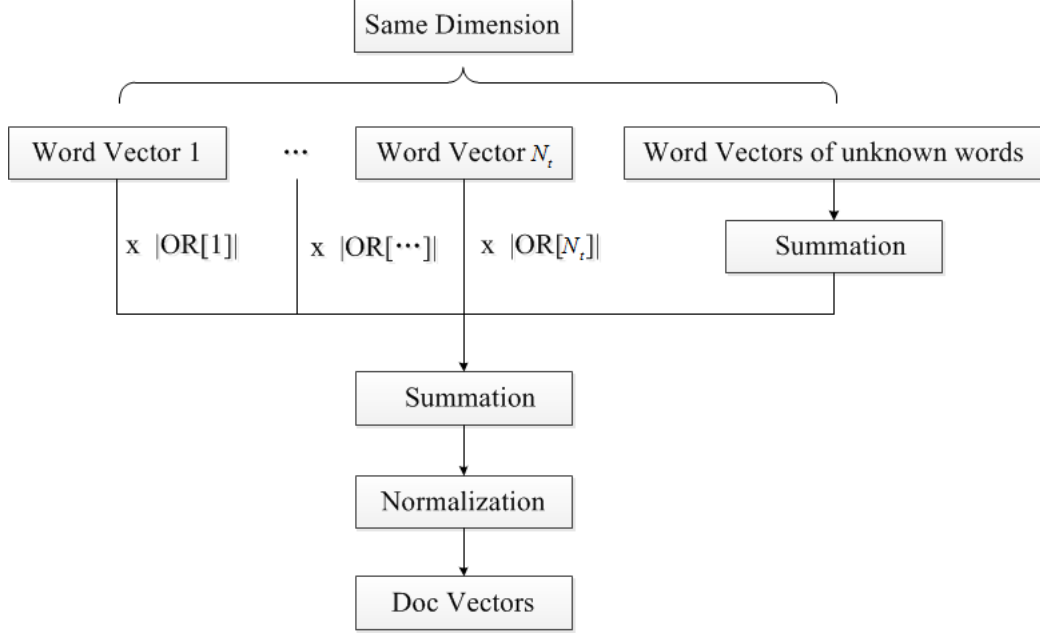


Figure 4.7: OR and Word Vectors Method

Flow chat 4.7 summarizes the processes in w2v-OR methods defined in formula 4.9. As shown, the difference between figure 4.7 and 4.6 is only the weight vector.

4.3.3 WFO Weighting Scheme and Word Vectors

The last method for combining supervised weighting scheme and word vectors is called WFO Weighting scheme and word vectors method (w2v-WFO) which is base on WFO term weighting scheme defined by formula 3.9. Flow Chart 4.8 (in the next page) shows the processes for employing w2v-WFO method. Because there is an adjustable parameter in WFO supervised term weighting scheme, w2v-WFO method is more flexible than previous two methods. And the price is that extra work is needed to select the best parameter for one particular data set, which corresponds the feedback component in Flow Chart 4.8 for selecting the best parameter (model).

Similarly, once a certain adjustable parameter is selected, w2v-WFO method

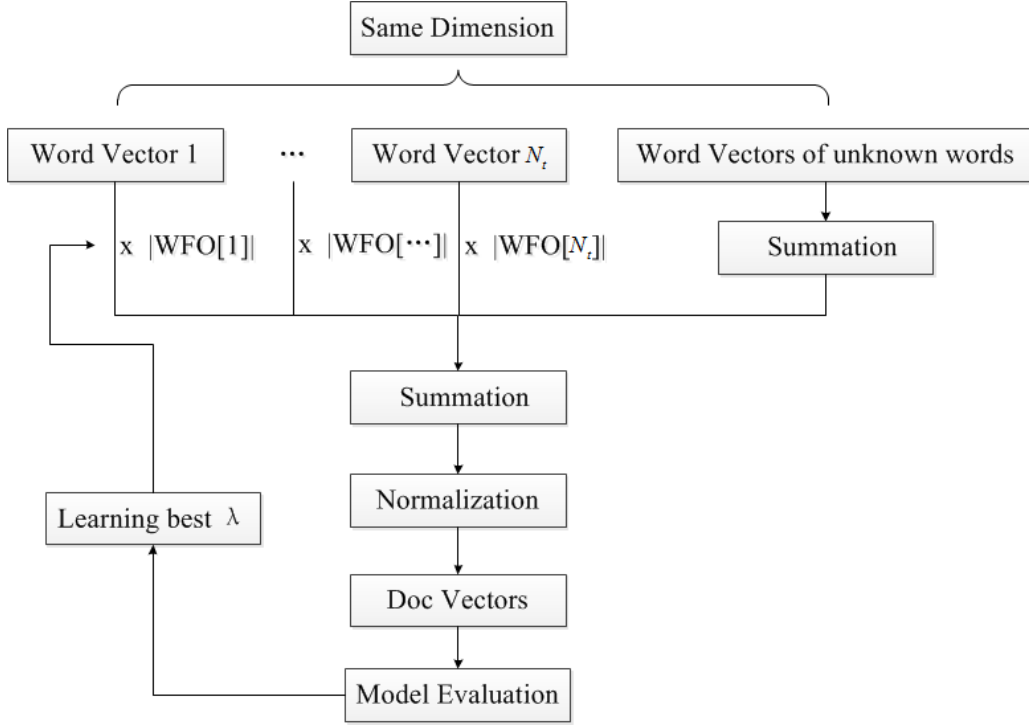


Figure 4.8: WFO and Word Vectors Method

transforms WFO weights into absolute values and assigns unknown terms unit weights. The processes is defined by formula 4.10:

$$Docvec^{(i)} = \sum_{1 \leq j \leq N_t} |WFO(\lambda)_j| \cdot W_j + \sum_{k \in S(i)} W_k \quad (4.10)$$

where $|WFO(\lambda)_j|$ represent the absolute value of corresponding WFO weight for the term with index j and λ represents the adjustable parameter.

4.4 Combination of Supervised Term Weighting Schemes and Binary Vectors

The second idea about incorporating word vector is to expand traditional methods by building links between unknown test words and training words. More specifically, the unknown test words can be replaced by the most similar words in training vocabulary, which can be found based on cosine similarity word vector arithmetic defined in formula 4.2.

Early research [42] shows that BoW model gives better performance with binary TF features for sentiment analysis. So in our methods, TF features in BoW model is firstly converted to binary values. Equation 4.11 is used to binarize the feature vectors:

$$\hat{\mathbf{f}}^{(i)} = \mathbf{1}\{\mathbf{f}^{(i)} > 0\} \quad (4.11)$$

where $\mathbf{f}^{(i)}$ is the TF feature vector of document i and $\hat{\mathbf{f}}^{(i)}$ represents binary TF feature vector. $\mathbf{1}\{\}$ is the indicator function: If TF of one term in document i is larger than zero, set it to one, otherwise set it to zero.

As we discussed before, there is no corresponding weights for unknown test words. In this section methods, we firstly find the most similar training word for unknown test word and replace this unknown word by corresponding training word to generate new binary TF feature vectors. Note that because word vectors files has its limitation, we also set a threshold (0.6) to select such similar words. If the biggest cosine similarity value is also smaller than 0.6 we will abandon this unknown word. The best thresholds slightly vary with datasets, but we fix it equal to 0.6 in our methods.

Once new test binary BoW vectors are calculated, we use formulas 3.5, 3.8, 3.9 introduced in Chapter 3 to compute supervised term weighting vectors: **LCR**, **OR**, **WFO**(λ). These term weighting vectors will be used as the basic reference vector to generate document feature vectors. For LCR weight scheme, each new weighted document feature vector is generated by multiplying **LCR** vector and binary BoW feature vector, which is defined by formula 4.12:

$$\mathbf{Docvec}^{(i)} = \mathbf{LCR} \circ \hat{\mathbf{f}}^{(i)} \quad (4.12)$$

where \circ means the elementwise product.

Similarly, for OR term weighting scheme, document feature vectors can be generated by formula 4.13:

$$\mathbf{Docvec}^{(i)} = \mathbf{OR} \circ \hat{\mathbf{f}}^{(i)} \quad (4.13)$$

Flow chart 4.9 summarizes the processes for generating document feature vectors by applying LCR or OR supervised term weighting scheme into BoW models with binary TF features:

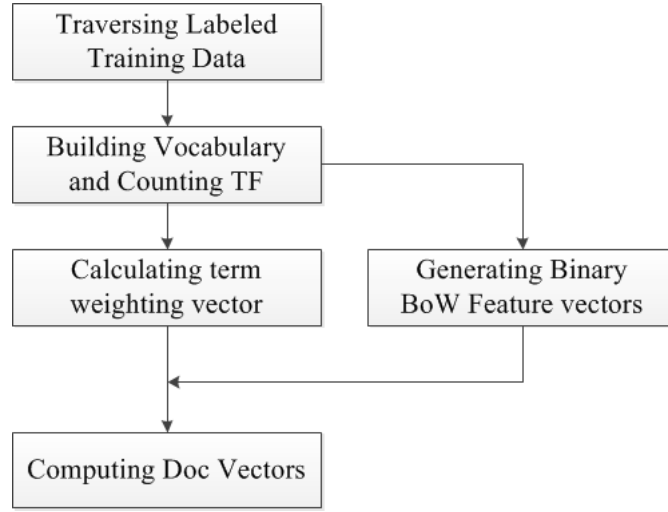


Figure 4.9: LCR/OR and Binary Vectors Method

And for WFO term weighting scheme, document feature vector can be generated by following formula 4.14:

$$\mathbf{Docvec}^{(i)} = \mathbf{WFO}(\lambda) \circ \hat{\mathbf{f}}^{(i)} \quad (4.14)$$

where λ is the adjustable parameter.

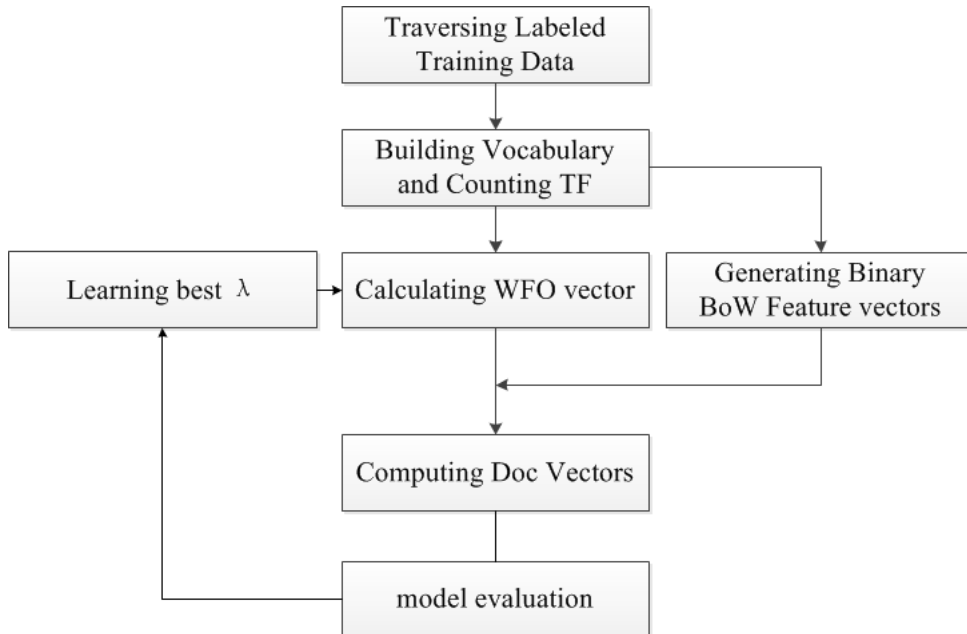


Figure 4.10: WFO and Binary Vectors Method

There is an adjustable parameter in WFO, the processes of using WFO scheme

method are a little different from LCR and OR. In WFO method, it will firstly learn the best parameter based on training documents and then use the learned WFO vector to generate document feature vectors by formula 4.14. So the method flow chart is modified as shown in figure 4.10:

Chapter 5

Experimental evaluation

5.1 Experiment Setup

5.1.1 Implementation

Text mining tasks normally include several basic processes: data collection and cleaning, feature learning, model training and evaluation. In our experiment, we integrated all these processes into one single module named Doc2vec. Flow Chat 5.1 (in next page) shows the processes within our experiment implementation. Firstly, we import documents with class labels and clean all the documents using uniform regular expression followed by lower-casing operation, which filters out rare characters excluding [A-Za-z0-9(,!?'“]. For several omit expresses, we recover the corresponding completed expresses. For example, transform “it’s” into “it is” and “I’ll” into “I will”. Note that we do not remove any stop words or perform stemming. After data cleaning, we segment the documents into word by the spaces among them. All documents are transformed into word sets whose element is single word. Then we initialize a Doc2vec object including documents (word sets) and the labels, training vocabulary (used for term weighting schemes) and set vocabulary (used for loading word vectors).

As some datasets are already separated into training part and test part while some are not. So for non-separated dataset, we use CV-10 technique to spilt the

dataset and manage to achieve credible results. More specifically, assign each document one random CV value (range 0 to 9) and in each round test, set those documents with a same CV as test data set and others are set as training data set. Once training and test parts are determined, we traverse those dataset to compute

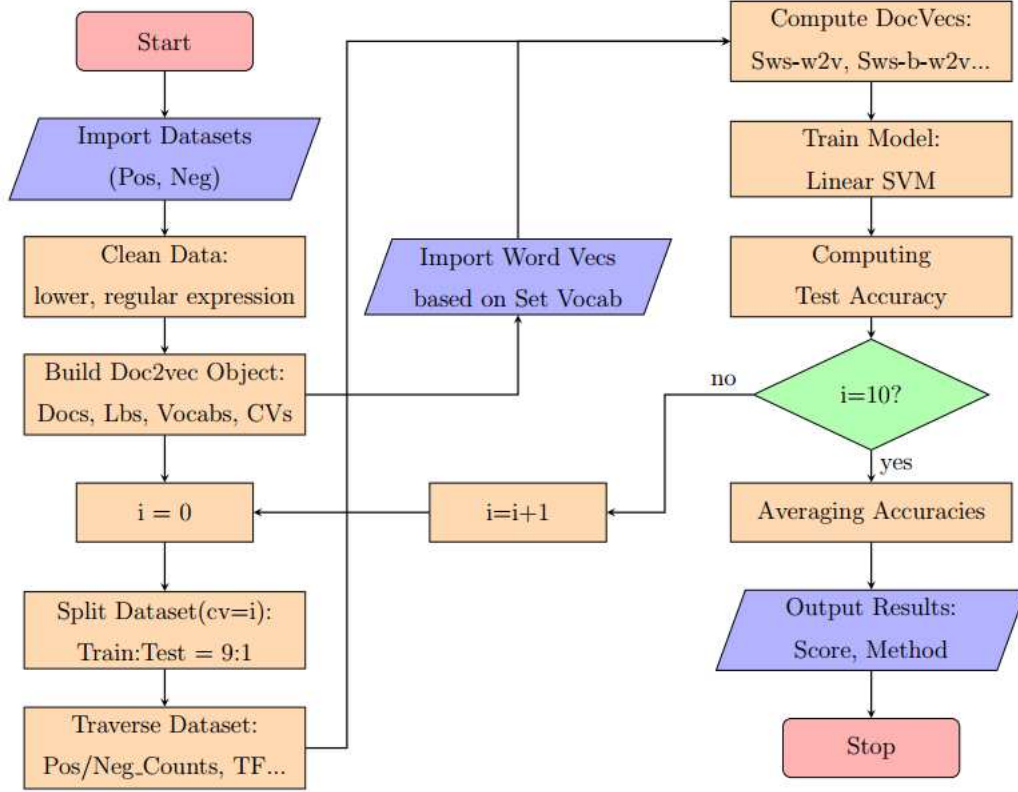


Figure 5.1: Experiment Implementation

other important statistical information such as TF for both datasets, positive and negative term counts for training data set and so on. Such information is used for computing document feature vectors by different methods discussed in previous chapters: unsupervised weighting schemes based on word vectors methods(Uws-w2v), supervised weighting schemes based on word vectors methods(Sws-w2v) and supervised weighting schemes based on binary vectors expanded by word vectors methods (Sws-b-w2v).

As we introduced before, in our methods we need to import pre-trained word vector files firstly. In order to save computer resources and speed up the query speed, we only import word vectors whose corresponding words are included in the

data set total vocabulary as shown in Flow Chat 5.1. And then document feature vectors are calculated by different proposed methods. The next process is training model on computed document feature vectors by linear SVM algorithm. And then evaluate model performances on test data set. For non-separated datasets, we repeat these processes and calculate the average accuracy while for pre-separated datasets, we just output the score of single test run. Actually, Flow Chat 5.1 summarizes the processes for sentiment analysis on non-separated datasets. For pre-separated datasets, flow chat can be simplified by removing the loop structure (10-CV).

5.1.2 Datasets and Word Vectors

We evaluate our methods on different benchmark datasets and compare our results with several published scores. Table 5.1 lists these datasets and the corresponding statistical information. Note that column length represents the average number of words in each document. As a result, we categorize the first three datasets into short length set group and the last two datasets into long length set group. Pos and Neg represent number of documents with positive and negative labels respectively. Vocab means the total vocabulary size of cleaned datasets. The last column CV means these datasets are non-separated and separated by 10-CV technique.

Table 5.1: Datasets Summary

Dataset	Length	Pos	Neg	Vocab	CV
PL-sen-05	21	5331	5331	9947	10
PL-subj-5k	24	5000	5000	11218	10
MPQA	3	3316	7308	20929	10
PL-2K	787	1000	1000	39727	10
IMDB	231	25K	25K	168152	None

Here are the details about these datasets:

- PL-sen-05¹: Short movie review data set and each review only has one sentence (Pang, et al., 2005)[43].
- PL-subj-5k: Short subjective movie review dataset and each review is either subjective or objective (Pang, et al., 2004) [44].
- MPQA²: Opinion polarity corpus dataset (Wiebe, et al., 2005)[45].
- PL-2K: The famous sentiment analysis benchmark dataset with 2000 full-length movie reviews (Pang, et al., 2004)[44].
- IMDB³: A large dataset with 50k full-length movie reviews from IMDB (Maas, et al., 2011)[46]

In our experiments, we use the large pre-trained word vectors file released by Google⁴. These word vectors are learned from part of Google News data set (about 100 billion words). Table 5.2 shows basic information about this file.

Table 5.2: Word Vectors File

Name	Dimension	Vocab	Size
GoogleNews-vectors-negative300.bin	300	3 Million	3.6G

5.2 Experiment Results and Analysis

5.2.1 Sws-b-w2v performs better on long-length datasets

In our experiments, we find Sws-b-w2v methods have better performances on longer-length documents. Table 5.3 shows the experiment results of our Sws-b-w2v methods as well as the ones of some baselines, where x-uni-w2v and x-bi-w2v mean that x method is base on 1-gram model and 2-grams model respectively.

¹<http://www.cs.cornell.edu/people/pabo/movie-review-dat>

²<http://www.cs.pitt.edu/mpqa/>

³<http://ai.stanford.edu/amaas/data/sentiment/>

⁴<http://code.google.com/p/word2vec/>

Cre-tfidf is a simple but novel supervised weighting scheme by adjusting TF in TF-IDF (Kim, et al., 2014) [23] and nbsvm is the excellent binary Naive Bayes SVM method which is the baseline for many sentiment analysis methods (Wang, et al., 2012) [24].

Table 5.3: Results of Sws-b-w2v methods against baselines

	Method	PL-sen-05	PL-subj-5k	MPQA	PL-2K	IMDB
our results	LCR-uni-w2v	78.4	91.3	85.6	88.2	88.29
	LCR-bi-w2v	<u>79.5</u>	92.0	85.7	89.8	91.22
	OR-uni-w2v	78.3	91.2	85.3	88.2	88.60
	OR-bi-w2v	79.4	91.9	85.5	89.8	<u>91.56</u>
	WFO-uni-w2v(0.1)	77.2	90.3	84.3	87.9	<i>89.48</i>
	WFO-bi-w2v	78.4	90.8	84.6	89.2	91.39
	WFO-uni-w2v(0.05)	77.9	91.1	85.0	88.1	89.18
	WFO-bi-w2v	79.3	91.5	85.4	89.5	91.54
	WFO-uni-w2v(0.01)	78.1	91.1	85.3	<i>88.4</i>	88.66
	WFO-uni-w2v	79.2	91.7	85.7	89.7	91.52
other results	Bag-of-words	77.1	91.0	86.3	88.1	88.6
	cre-tfidf-uni	77.5	91.8	-	88.7	88.8
	cre-tfidf-bi	78.6	92.8	-	89.7	91.3
	nbsvm-uni	78.1	92.4	85.3	87.8	88.29
	nbsvm-bi	79.4	<u>93.2</u>	<u>86.3</u>	89.5	91.22

Note, top 3 scores in each column are in bold and the best one is uniquely underlined. Those italic numbers mean they are not in top 3 but are the highest ones among results of methods based on unigram models

From Table 5.3, it can be seen that for longer-length datasets (PL-2K and IMDB), our methods outperform the state of the art methods. OR-bi-w2v and LCR-bi-w2v have the same best accuracy (89.8%) on PL-2k slightly better than the state of the art accuracy (89.7%) of cre-tfidf-bi method. For uni-gram models, WFO-uni-w2v has the best accuracy (88.4%) on PL-2K with adjustable parameter equal to 0.01. Moreover, our Sws-b-w2v methods consistently outperform the

listed other methods on IMDB datasets. OR-bi-w2v achieves the best accuracy at 91.56% and WFO-uni-w2v with $\lambda = 0.1$ has the highest accuracy (89.48%) in term of uni-gram models. On PL-sen-05 data set, the results of LCR-w2v and OR-w2v methods are slightly better than (close to) the results of nbsvm methods while the performances of WFO-w2v methods are lightly weaker which is still better than cre-tfidf methods.

As for other two short documents (snippets) datasets, the results of our Sws-b-w2v methods are close to the one of the state of the art methods. Our methods are slightly weaker on PL-subj-5k and MPQA. Particularly, for MPQA our methods fail to achieve satisfying results because there are only average 3 words within each MPQA dataset document and our supervised term weight schemes can not find good discriminative words from them. However, as we discussed before, word vectors remain intrinsic information among words and Uws-w2v or Sws-w2v methods may achieve better results on such short snippets datasets. Furthermore, comparison among our methods, it can be seen that LCR-w2v methods are better for short length datasets while OR-w2v methods perform better on full-length datasets and have the best overall performance. WFO-w2v methods have robust performance but it needs extra efforts to find the appropriate parameter for certain data set.

Based on the performance comparisons and results analysis, we can draw conclusions that our proposed methods applying supervised weighting schemes on binary BoW feature vectors have its advantages for sentiment analysis tasks and the idea that utilizes word vectors to link unknown test words with training can improve its performance further..

5.2.2 Word Vectors based methods perform better with supervised schemes

As we only take the word vectors in our Uws-w2v and Sws-w2v methods excluding the phrase vectors, we compare our word vectors based methods with published methods based on unigram models. Table 5.4 summarizes results of our Word

Vectors based methods (rows 2-3 are Uws-w2v methods and rows 4-8 are Sws-w2v methods) as well as the ones of baseline methods on three short documents (snippets) datasets.

Table 5.4: Results of Uws-w2v and Sws-w2v methods against baselines

Method	PL-sen-05	PL-subj-5k	MPQA
w2v-average	77.6	91.2	<u>88.5</u>
w2v-tfidf	77.3	90.0	86.3
w2v-LCR	77.8	91.4	87.5
w2v-OR	77.8	91.3	87.4
w2v-WFO(0.1)	78.0	91.0	85.3
w2v-WFO(0.05)	78.0	91.0	87.2
w2v-WFO(0.01)	77.7	91.0	87.4
Bag-of-words	77.1	91.0	86.3
cre-tfidf-uni	77.5	91.8	-
nbsvm-uni	<u>78.1</u>	<u>92.4</u>	85.3

It can be seen that our word vectors based methods consistently outperform Bag-of-words method and have satisfactory results close to the ones of the state of the art methods. As we directly use the pre-trained Word Vectors and have not relearned them on certain dataset, the performances on different datasets are affected to some extent by the Word Vectors. And, if some words in certain dataset are not included in the vectors file, it may cause some errors (we abandon such words).

Generally, Sws-w2v methods outperform Uws-w2v methods. W2v-WFO ($\lambda = 0.1$ or 0.05) has the best accuracy on PL-sen-05 dataset among our methods and w2v-LCR has the best accuracy on PL-subj-5k dataset among our methods. For MPQA, it has been shown in 5.3 that our supervised weighting schemes learned bad weights because of its too short document length, so Sws-w2v methods have weaker performance. However, as word vectors contain intrinsic information among words, we find the simplest w2v-average methods has the best

performance on MPQA and even outperforms the state of the art methods (both unigram model and bigram model based). Moreover, poorer performances of w2v-tfidf and Bag-of-words proof the idea that supervised term weighting schemes will improve sentiment analysis systems.

Chapter 6

Conclusion and perspectives

6.1 Summary

In this thesis, we introduced two unsupervised term weighting schemes and three supervised term weighting schemes based on bag of words model. We compared these three supervised weighting schemes on sentiment analysis tasks. Then we discussed the advanced word vector technology and introduced the cosine similarity methods to measure intrinsic relationship between words. We proposed two kinds of ideas to utilize word vectors in sentiment analysis tasks to improve analysis systems: one is based on multiplication operation and the other is used for expanding BoW models with binary features.

Our experiment results show supervised term weighting schemes and the intrinsic information among words can really improve the performance of sentiment analysis. The performance of document vectors learned are evaluated by sentiment analysis system, in which SVM is used as the basic classifier. Our proposed methods have achieved good performance, in which the performance on long-length movie review is even slightly better than the state of the art.

6.2 Future Work

Supervised term weighting schemes are all based on statistical information and have the limitations in some situations. For example, “funny” and “unfunny” have opposite meanings and are obviously discriminating words. But “not funny” also has high counts among negative emotional documents which will result in wrong predicts for the word “funny”. As word vectors can provide intrinsic information among words, for future work we will try to find a more sophisticated and robust way to incorporate word vector into designing supervised weighting schemes.

Bibliography

- [1] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.
- [2] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.
- [3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.
- [4] Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.
- [5] Sanjiv Ranjan Das and Mike Y Chen. Yahoo! for amazon: Sentiment parsing from small talk on the web. In *EFA 2001 Barcelona Meetings*, 2001.
- [6] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.
- [7] ZHAO Yan-Yan, QIN Bing, and LIU Ting. Sentiment analysis. *Journal of Software*, 21(8):1834–1848, 2010.
- [8] Hong Yu and Vasileios Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in*

- natural language processing*, pages 129–136. Association for Computational Linguistics, 2003.
- [9] TF Yao and SW Peng. A study of the classification approach for chinese subjective and objective texts. *Proc. of the NCIRCS*, 2007:117–123, 2007.
 - [10] A Sentimental Education. Sentiment analysis using subjectivity summarization based on minimum cuts. *Bo Pang and Lillian Lee*, 2004.
 - [11] Luciano Barbosa and Junlan Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 36–44. Association for Computational Linguistics, 2010.
 - [12] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.
 - [13] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49. ACM, 2004.
 - [14] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.
 - [15] Timothy G Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Improvements that don’t add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 601–610. ACM, 2009.
 - [16] Justin Martineau and Tim Finin. Delta tfidf: An improved feature space for sentiment analysis. In *ICWSM*, 2009.
 - [17] Georgios Paltoglou and Mike Thelwall. A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the 48th Annual*

- Meeting of the Association for Computational Linguistics*, pages 1386–1395. Association for Computational Linguistics, 2010.
- [18] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.
 - [19] Franca Debole and Fabrizio Sebastiani. Supervised term weighting for automated text categorization. In *Text mining and its applications*, pages 81–97. Springer, 2004.
 - [20] Pascal Soucy and Guy W Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *IJCAI*, volume 5, pages 1130–1135, 2005.
 - [21] Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. Supervised and traditional term weighting methods for automatic text categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):721–735, 2009.
 - [22] Zhi-Hong Deng, Kun-Hu Luo, and Hong-Liang Yu. A study of supervised term weighting scheme for sentiment analysis. *Expert Systems with Applications*, 41(7):3506–3513, 2014.
 - [23] Yoon Kim and Owen Zhang. Credibility adjusted term frequency: A supervised term weighting scheme for sentiment analysis and text classification. *arXiv preprint arXiv:1405.3518*, 2014.
 - [24] Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012.
 - [25] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.
 - [26] Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

- [27] Fei Huang and Alexander Yates. Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 495–503. Association for Computational Linguistics, 2009.
- [28] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, 2009.
- [29] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.
- [30] Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM, 2007.
- [31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [32] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.
- [33] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.
- [34] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics, 1996.

- [35] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.
- [36] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTER-SPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.
- [37] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [38] John C Platt et al. Using analytic qp and sparseness to speed training of support vector machines. *Advances in neural information processing systems*, pages 557–563, 1999.
- [39] CJ Van Rijsbergen, DAVID J Harper, and MARTIN F Porter. The selection of good search terms. *Information Processing & Management*, 17(2):77–91, 1981.
- [40] Shoushan Li, Rui Xia, Chengqing Zong, and Chu-Ren Huang. A framework of feature selection methods for text categorization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 692–700. Association for Computational Linguistics, 2009.
- [41] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. *arXiv preprint arXiv:1502.03520*, 2015.
- [42] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

- [43] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics, 2005.
- [44] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.
- [45] Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210, 2005.
- [46] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

Appendix A

Experiment environment:

Operation system: Ubuntu 12.04

Hardware: 4-CPU 2.67GHz, 4G RAM

Project Codes (Python): <https://github.com/linbojin/dv4sa>



Figure A.1: Project Screenshot

Result Screenshots:

```
['esos'] ('esos', 0.83135986328125)
['recordings'] ('recording', 0.6843936443328857)
['accelerated'] ('slowed', 0.6032029390335083)
['comedians'] ('comedian', 0.7487119436264038)
['voyage'] ('voyages', 0.8024001121520996)
['gorefests'] ('gore', 0.6289738416671753)
['connoisseur'] ('connoisseurs', 0.6636632084846497)
['partially'] ('partly', 0.6639516949653625)
['minty'] ('creamy', 0.6456297636032104)
['preteens'] ('teens', 0.6333256959915161)
['penetrates'] ('absorbs', 0.6428040862083435)
['tantalizing'] ('intriguing', 0.7066733837127686)
['aroused'] ('arouse', 0.7325844764709473)
['perceptively'] ('insightfully', 0.601125955581665)
['endearingly'] ('charmingly', 0.7552269697189331)
['adoration'] ('affection', 0.7005795240402222)
['connects'] ('connect', 0.7299025058746338)
['bared'] ('baring', 0.6799218654632568)
['confronted'] ('confronting', 0.6697782874107361)
['liberated'] ('liberating', 0.6759877800941467)
['holidays'] ('holiday', 0.8605460524559021)
['ensnared'] ('entangled', 0.6429120302200317)
['continental'] ('continent', 0.6160043478012085)
['divides'] ('divide', 0.7381653189659119)
['antidotes'] ('antidote', 0.603725254535675)
['frisky'] ('randy', 0.6193982362747192)
['amongst'] ('among', 0.7948091626167297)
['tiempos'] ('esas', 0.7826317548751831)
['engulfing'] ('engulfed', 0.7313540577888489)
['resent'] ('resentful', 0.6545130610466003)
['snicker'] ('giggle', 0.6516643166542053)
['knowingly'] ('willfully', 0.6525254845619202)
['lauren'] ('rebecca', 0.7202187776565552)
['pentecostal'] ('evangelical', 0.6329083442687988)
['dallas'] ('austin', 0.6741836667060852)
['services'] ('service', 0.6880399584770203)
['monica'] ('michele', 0.7114782333374023)
```

Figure A.2: Similarity Words with Distances

```

inventive : 3.04037184464
engrossing : 2.73676754923
lively : 2.70384750885
wonderfully : 2.70384750885
heartwarming : 2.63484595444
grown : 2.56072929943
vividly : 2.56072929943
captures : 2.48077338772
detailed : 2.48067790897
evocative : 2.48067790897
gem : 2.48067790897
tour : 2.48067790897
captivating : 2.48067790897
explores : 2.39365784928
spare : 2.39365784928
potent : 2.39365784928
unflinching : 2.39365784928
answers : 2.39365784928
russian : 2.39365784928
jealousy : 2.39365784928
richly : 2.39365784928
masterful : 2.39365784928
haynes : 2.39365784928
startling : 2.39365784928
culture : 2.37092005602
powerful : 2.37092005602
wonderful : 2.33129372147
warm : 2.33129372147
unique : 2.29841709695
refreshing : 2.29841709695

```

(a) Positive Words

```

named : -2.30698781831
carvey : -2.30698781831
incoherent : -2.30698781831
lousy : -2.40230671837
kung : -2.40230671837
meandering : -2.40230671837
soggy : -2.40230671837
choppy : -2.40230671837
ballistic : -2.40230671837
51 : -2.40230671837
conceived : -2.40230671837
generic : -2.44685444802
britney : -2.48932681569
product : -2.48932681569
waste : -2.48942278369
mediocre : -2.48942278369
routine : -2.48942278369
fatal : -2.56937824378
inept : -2.56937824378
plodding : -2.64349493642
pinocchio : -2.64349493642
bore : -2.71249652847
tiresome : -2.71249652847
benigni : -2.77704377024
pointless : -2.77704377024
disguise : -2.83767711277
boring : -2.83795629836
seagal : -2.8948442474
poorly : -2.8948442474
badly : -3.22340936204

```

(b) Negative Words

Figure A.3: Discriminating Words with Weights