NUS
National University
of Singapore

Founded 1905

Data Analysis for Emotion Identification in Text

ZHANG Zhengchen

(B. Eng, M. Eng)

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2013

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

2013-05-31

ZHANG Zhengchen

Acknowledgments

I would like to express my deep and sincere gratitude to my supervisor, Professor Shuzhi Sam Ge. Without his inspiration and guidance, it would not be possible for me to finish my journey in obtaining my Ph.D. In the past four years, Professor Ge taught me invaluable methodologies of doing research. He gave me many opportunities to attend academic conferences and even to organize conferences. It was my great honor to work with Professor Ge and his team members. I appreciate the culture of the laboratory built by Professor Ge that every one is self motivated and selfless. I am very grateful for Professor Ge's enthusiasm, vision and leadership in research that, taken together, make him a great mentor. Words are short to express my deep sense of gratitude towards Professor Ge.

I am deeply grateful to my co-supervisor, Professor Chang Chieh Hang, for his constant support and assistance during my Ph.D. study. His passion and sharpness influenced me greatly on the research work. I am indebted to the other committee members of my PhD program, Dr. Haizhou Li, Agency for Science, Technology and Research (A*STAR), Singapore and Professor Cheng Xiang, NUS for the assistance and advice that they provided through all levels of my research progress. I am sincerely grateful to all the supervisors and committee advisers who have encouraged and supported me during my PhD journey.

I wish to express my warm and sincere thanks to my senior, Dr. Hongsheng He, for his lead at the beginning of my research and discussion of the work in this thesis. It was a great honor for me to work with Dr.

Dongyan Huang, A*STAR, Singapore to participate in the INTERSPEECH 2011 Speaker State Challenge and win the Sleepiness Sub-Challenge Prize. I appreciate the generous help, encouragement and friendship from Yanan Li, Qun Zhang, Wei He, Shuang Zhang, and many other fellow students/colleagues in the group. All the excellent fellows made my PhD marathon more fun, interesting and fruitful.

I take this opportunity to sincerely acknowledge the National University of Singapore (NUS), for providing financial assistance which buttressed me to perform my work comfortably.

I owe my loving thanks to all my family members. Without their encouragement and understanding it would have been impossible for me to finish this work.

Contents

Abstract

This thesis investigates how to identify emotional sentences in an article using data analysis technologies. Two types of methods are proposed to solve the problem through classifying data and learning data association.

A straightforward method of identifying emotional sentences is to formulate it as a classification problem. It is an imbalanced pattern classification problem because the number of neutral sentences is much more than the number of emotional ones in an article. A classifier based on Linear Discriminant Analysis (LDA) is proposed for the classification on imbalanced data sets. Emotional words and special punctuations are taken as features for the classifier. Experiments conducted on a children's story corpus demonstrate that the proposed method could generate competitive results with state-of-the-art systems while consuming much less time. A Partial Least Squares (PLS) based classifier which has been applied to other imbalanced pattern classification problems like speaker state classification is employed to perform the emotional sentence identification task. Both the Un-weighted Accuracies (UA) are about 0.66 obtained by the LDA based and PLS based methods on the UIUC children story corpus. The classifier fusion is also investigated to further improve the system performance by combining different classifiers and features. The experimental results demonstrated that the fusion of the Extreme Learning Machine and the Asymmetric Simple Partial Least Squares (SIMPLS) based classifier could generate better performance than single classifiers.

Emotion identification in text is formulated as a ranking problem that

calculates the score of emotion which is hidden in every sentence. The sentences with higher emotion scores are predicted as emotional ones. With the associations between words, bigrams and sentences, a mutual reinforcement ranking algorithm is proposed to address the graph based ranking problem. Experimental results obtained on the UIUC children story corpus showed that the method was faster than Support Vector Machines (SVM), while the system performance was a little worse than SVM. The algorithm is applied on the document summarization problem by employing the associations between words, sentences, and sentence clusters. The experimental results obtained on DUC-2001 and DUC-2005 data sets showed the effectiveness of the proposed approach. The associations between objects should be updated if new objects are appended to a data set. The incremental learning of data association is discussed to make association based methods be able to adapt to new data. Experiments of word similarity estimation and link prediction in social network proved the effectiveness of the proposed method.

List of Tables

List of Figures

Chapter 1

Introduction

Emotion identification in text studies the emotion contained in a sentence or in an article. In the research filed of emotion recognition, six categories of emotion are often used to label the sentences: anger, disgust, fear, joy, sadness, and surprise [1, 2, 3].

In this work, we focus on the first step of identifying emotion in text: to detect emotional sentences in an article. Much research has been conducted in related fields. Classifying a sentence into a neutral or emotional class is a straightforward solution to this problem. It is an imbalanced pattern classification problem as the number of neutral sentences in an article is much larger than the number of emotional ones. Traditional algorithms may be affected by the imbalanced data set. Many technologies like over-sampling have been proposed to address the imbalanced pattern classification problem [4]. It will take more time to over-sample the data, which makes the whole learning process slow. In this work, some efficient classifiers are introduced to address the imbalanced pattern classification problem. Another way of solving the emotional sentence identification problem is to calculate the degree of emotion hidden in every sentence. The sentences with higher emotion scores are selected as the emotional ones. In this case, the classification problem becomes a ranking problem. To the best of our knowledge,

no study has been reported for ranking the emotional scores of sentences although much work has been done to solve the other ranking problems. In this thesis, a mutual reinforcement learning algorithm that utilizes the associations between terms, bigrams and sentences is presented to rank the sentences. The method is also applied on document summarization tasks. The data association such as bigram-term affinity should be updated with the growth of the data set. We investigate the incremental learning of data association to make the association based methods be able to adapt to new data sets.

In this chapter, the background and motivation of this research is first introduced. The related work is described to show the state-of-the-art research of emotion identification in text, imbalanced pattern classification and data association. The contribution of this thesis is summarized, and the thesis structure is illustrated at last.

1.1 Background

The research of emotion identification in text has drawn considerable amount of interest in the field recently. There are many important applications such as analyzing customer feedback of products [5] and building intelligent dialog agents [6]. In the Social Robotics Laboratory[1] at National University of Singapore (NUS), we have developed a social robot named Adam as shown in Fig. 1.1 which is expected to be able to tell stories with emotional speech and gestures. It is necessary to understand emotions in a story for Adam to tell stories with emotional speech. In this work, we focus on the first

[1] http://robotics.nus.edu.sg/

Figure 1.1: The appearance of Adam.

step of identifying emotions in text: detecting the emotional sentences in an article. To identify emotional sentences in a story, this work focuses on data classification and data association technologies.

1.2 Related work

1.2.1 Emotion identification in text

The difficulty of identifying emotions in pure text is caused by the variety and complexity of languages. Statistical methods are popular in the emotion

identification field, and many annotated corpora have been presented to meet the requirements of these methods. The UIUC children's stories corpus [1] consists of 176 stories by Grimm's, Andersen, and Potter. Every sentence in the corpus was annotated with one of the seven emotions described above by two annotators. An emotion annotation task was also reported in [7] that the emotion category, emotion intensity and the words/phrases that indicate emotion in text were annotated on a corpus of blog posts. There are some other corpus like MPQA [8] and the news headline corpus reported in [9], but it is difficult to find a gold standard corpus. The emotion annotated corpora are not same as the general purpose corpora such as PennTreebank [10] because the inter-annotator agreement on labeling a sentence as emotion or non-emotion is lower.

There are many machine learning algorithms available in the field of emotion identification in text. In [11], the authors reported the experiments on a preliminary data set of 22 fairy tales using supervised machine learning with the SNoW learning architecture for classification of emotional versus non-emotional contents. Experiments were also conducted in [7] to classify the emotional and non-emotional sentences by employing the SVM classifier. The features employed in these two papers included emotional words, Part-of-Speech (POS) of words, special punctuations, etc. Both of the work only studied the two classes classification problem, while not addressed the multiclass classification for all the emotions. In [2], the authors presented a method of classifying all the emotions. To study the influence of word features on classification accuracy, the authors only took emotional words as the features. A mutual information feature extraction algorithm was proposed to select strong emotional words. SVM was taken as the classifier

to determine the emotion of each sentence. The influence of word features on emotion classification was studied, while the classification results were affected severely by the imbalanced data set in which most of the sentences were annotated as neutral sentences, which indicated that emotion recognition in text was an imbalanced pattern classification problem. A hierarchical classification method for the multi-emotion classification problem was proposed in [3]. The authors first classified sentences into emotional and non-emotional classes, then they classified the emotional sentences into positive and negative sentences. Finally every sentence was classified into a specific emotion class such as happiness, fear, etc. The experimental results demonstrated that the method was able to alleviate the influence to system performance caused by an imbalanced data set.

In the 4th International Workshop on Semantic Evaluations (SemEval-2007), one of the tasks was to annotate news headlines using a predefined list of emotions [12]. The system proposed in [13] employed synonym expansion and matched lemmatized unigrams in the test headlines against a corpus of hand annotated headlines. A rule based method was proposed in [14] to detect the sentiment of news headlines. The authors first evaluated the emotion and valence of individual words using rules and some knowledge resources like WordNet-Affect. Then the word which was the root of the dependency graph of a headline was selected as the head word, and the weight of this word was set to be higher than other words.

Many theories in psychology provide ideas for computer scientists to detect emotion in text. Based on the appraisal theories, the authors in [15] built a knowledge base which stores affective reaction to real-life contexts. In the knowledge base, a situation is presented as a chain of actions, and each action

is a triple: agent, action, and object. The emotion of a situation depends on the relationship between actions taken place on the agents and objects. This method can understand emotion hidden in the language although some of the work has to be done manually like building the core of the knowledge base and connecting the actions in a chain. In [16], the authors predict the positive or negative senses of a sentence using semantic dependency and contextual valence analysis. After obtaining the dependencies in a sentence, a set of rules were applied to calculated the emotional valence of each dependency. The emotion of a sentence is a combination of the valence and sign of the dependencies. The OCC model [17], which is a famous model in the appraisal theories, is presumably the most accepted appraisal emotion model by computer scientists because it provides a finite set of clear criterion for identifying emotions. These criteria were applied in [18] to sense the affect in text. The authors defined a set of variables and mapped text components to specific values of the variables. The rules of the OCC model were employed to predict the emotion of a sentence based on the value of the variables. The OCC model has also been formalized in a logical framework [19] and applied in generating emotions for embodied characters [6].

1.2.2 Imbalanced pattern classification

A straightforward method of identifying emotional sentences is to classify a sentence into neutral or emotional class. It is an imbalanced pattern classification problem because there are much more neutral sentences than emotional ones in an article. Imbalanced pattern classification has drawn considerable attention in the field of Machine Learning in recent years. If the

number of samples belonging to one of the classes is much smaller than others in an imbalanced data set, the performance of some learning algorithms decreases significantly [4, 20], especially for the minority class. People are more interested in the rare cases in many real world tasks like medical diseases diagnosis [21] and text processing [3, 22].

Some algorithms have been proposed to address the imbalanced classification problem. A review of the state-of-the-art technologies was conducted in [4] where three types of methods were introduced: sampling methods [23, 24, 25, 26, 27], cost sensitive methods [28, 29, 30, 31, 32], and kernel based methods [33, 34, 35]. Sampling methods either add new samples to the minority class (oversampling) or remove samples from the majority class (undersampling) to balance the data set. Hence, the standard learning algorithm can be applied to the modified data set. Such methods will change the original distribution of the data set, and they do not improve the classifiers. Nevertheless, the sampling technique is able to improve system performance on most imbalanced data sets. In general, cost sensitive methods aim to minimize the overall misclassifying cost on the training data set, where the misclassifying cost is a penalty of classifying samples from one class to another. There is no cost for correct classification, and the cost of misclassifying minority samples is set to be higher than the majority ones in cost sensitive methods. Many standard methods like AdaBoost [29], Neural Networks [30], and SVM [31, 32] have been adapted for the imbalanced learning. The basic idea of kernel methods is to map the features of samples from a linear nonseparable space into a higher dimensional space where the linear separation can be conducted. A kernel boundary alignment (KBA) algorithm was proposed in [33] that the kernel matrix was generated by a conformal transformed ker-

nel function according to the class-imbalance ratio. The SVM boundary was enlarged by the kernel matrix, and the classification accuracy was improved. Methods integrating kernel methods and sampling methods are also proved to be efficient for imbalanced learning problems [34, 35].

Some methods for feature extraction and dimension reduction have been adapted to address classification problems [36, 37, 38]. A partial least squares (PLS) based classifier was proposed for unbalanced pattern classification in [36]. The borderline of a PLS classifier was moved towards to the center of minority class to increase the accuracy for majority class without decreasing the minority class accuracy. The classifier was proved to be affected little by the class distribution in classification, and it could generate good classification accuracy for the minority class. The method also consumed less computation time than algorithms like SVM and Adaboost.

1.2.3 Data association

In this work, emotion identification in text is also formulated as a ranking problem that calculates the score of emotion for each sentence. We assume that every sentence contains some emotion, and the degree of the emotion is determined by the bigrams and words in a sentence. If the emotion score of a sentence is high enough, it is selected as an emotional sentence. Hence, one need to rank the emotion scores of all sentences. A mutual-reinforcement learning method is proposed to solve the ranking problem using the associations between sentences, bigrams, and terms.

Association is one of the fundamental intelligence of human beings that plays an important role in perception, recognition and emotion building. In

statistics, an association is any relationship between two measured quantities that renders them statistically dependent [39]. Association between objects has been applied to information retrieval for a long time. In [40, 41, 42], word co-occurence was employed for document retrieval. Association rule mining is a typical application of data association which aims to find the associations between items from a set of transactions [43, 44, 45]. Recently, the fast development of social networking service web like facebook provides a new platform for researchers to study the relationships and activities of users online. One of the important applications is to predict links for users which connect people who may know each other [46, 47, 48].

Computing semantic similarities between words is a direct application of distance measurement between objects. It has been widely used in word sense disambiguation [49], document retrieval [50], and hyper link following behavior prediction [51]. There are two main directions of computing word similarities: thesaurus based methods and corpus based methods. Thesauruses like WordNet [52] store relationship between words like synonymy and hypernymy. Several methods have been proposed to calculate the word similarity utilizing such relationships [53, 54]. Methods using large corpus like web pages [55] and Wikipedia [56] provide another solution to this problem. In [56], the authors used machine learning techniques to represent the meaning of any text as a weighted vector of Wikipedia-based concepts. Conventional metrics like cosine similarity were used to assess the relatedness of texts by comparing the corresponding vectors.

Association analysis in data mining is to find interesting relationships hidden in a large data set [57]. The uncovered relationships are normally represented in the form of association rules or sets of frequent items. Associ-

ation rule mining aims to find associations among items from a set of transactions that every transaction contains a set of items [45]. The most popular application of association rule mining is to do affinity analysis on products for a store. It has also been applied to evaluate page views associated in a session for a web site. An association rule is of the format: $LHS \rightarrow RHS$, where LHS is the left hand side and RHS is the right hand side. They are two item sets with no common items. The task of association rule mining is to find such rules that appear frequently in a data set. Many algorithms have been proposed like Apriori [58], Charm [44], FP-growth [43], Closet [59], Magnum Opus [60], etc. Normally an association rule mining method generates frequent item sets first and then construct rules based on these item sets [45]. Some efficient algorithms have been reported for the frequent item sets generation [43, 61]. The second step is relatively straightforward. The association rule mining problem describes a basic type of association between objects: co-occurrence. The association rule mining can only find association existing in the training data set. In this thesis, the technology of predicting hidden associations will be discussed to discover possible associations that are not in the training sets. The association rule mining discovers frequent item sets, no matter how many items are there in a set. In this thesis, we will only discuss the association between two objects. Normally, the association rule mining only finds item sets with support and confidence bigger than some thresholds, while does not emphasize the vale of support or confidence of each association. The association here will calculate the values which is called association degree.

Link prediction will encourage communications between users in a social network, which has attracted much attentions in the information retrieval

research field. Measuring distances between nodes in a network is a fundamental step of predicting links. The common neighbors [46] method counts the number of neighbors shared by two nodes. Two nodes are more likely to become friends if the overlap of their neighborhoods is large. Adamic and Adar [47] claimed that rarer neighbors are more important than the common neighbors. Hence, common neighbors of low degrees are given higher weights in an Adamic/Adar score. The preferential attachment method is based on the idea that the probability of having a new neighbor is proportional to the number of the current neighbors for a node. Moreover, the probability of two users becoming friends is proportional to the product of the number of their current friends [48]. A method of taking weights of links into account to measure the distance between nodes was proposed in [62]. Some researchers formulate the link prediction as a classification problem which predicts a new link is true or false [63]. Features like the number of neighbors, distances between nodes were employed to train a SVM, which was used to predict new links between nodes.

1.2.4 Incremental learning of data association

Data association describes the connection between objects. The connections should be updated with the growth of the data set. For example, the bigram-term associations are employed to calculate the emotion score of a sentence in this work. The associations are calculated using the semantic similarities between words, and such similarities are trained on a labeled corpus. New annotated sentences may be added to the corpus, and the associations should be updated to adapt to the new sentences. Incremental learning of data

association is investigated in this work.

In the association rule mining field, techniques for maintaining discovered association rules in an updated database have been proposed in [64, 65]. An efficient algorithm was proposed in [64] which updates the rules found in an original DB with an increment db. The authors noticed some interesting phenomenons such as by scanning the increment db only, many item sets can be pruned away before the update against DB. Utilizing these features, the rules appeared in the new database and those no longer have enough supports will be discovered without running algorithms like Apriori again. In [65], a more general algorithm was proposed which also considered the deletion of item sets from the original database. Such methods can be a reference for solving the incremental learning problems that measure association degrees by counting co-occurrence times of objects. For other association degree measurements like cosine similarity, these methods are not applicable.

In the machine learning field, several methods have been proposed to train Support Vector Machines (SVM) [66, 67] and Neural Networks (NN) [68] incrementally. As the SVM optimization problem is a linearly constrained quadratic programming problem, the authors of [67] used a "warm-start" algorithm which took an existing solution as the starting point to find a new solution. The method took advantage of the natural incremental properties of the standard active set approach to linearly constrained optimization problems. It was able to quickly retrain a support vector machine after adding a small number of new training vectors to the existing training set. Another way of solving incremental training of SVM is to update the margin vector coefficients and the bias to preserve the Karush-Kuhn-Tucker (KKT) Conditions on both new and old data [66]. Inspired by AdaBoost, the authors

of [68] proposed an incremental learning algorithm for supervised neural networks which combine several *weak* classifiers obtained on different data sets using Littlestone's majority-voting scheme [69]. All these methods are supervised learning methods, and the training set needs to be labeled manually. The requirement is not always satisfied in the association learning.

1.3 Contributions

The aim of this study is to investigate efficient algorithms of classifying data in an imbalanced data set and learning data associations for emotion identification in text. The main contributions of this thesis are highlighted as follows:

(i) an LDA based classifier is proposed to address the imbalanced pattern classification problem, and the proposed method is applied on the emotional sentence detection problem;

(ii) the ASimPLS method which has been proved to be efficient for other imbalanced pattern classification problems is employed to solve the emotional sentence identification problem;

(iii) fusion of different classifiers is investigated to improve the classification performance;

(iv) the emotional sentence detecting task is formulated as a graph ranking problem, and a mutual reinforcement ranking method utilizing the association between terms, bigrams, and sentences is proposed to address the graph based ranking problem; and

(v) incremental learning for data association is proposed which makes an association model be able to adapt to new data.

1.4 Thesis Structure

The thesis structure is shown in Fig. 1.2. The emotional sentences identi-

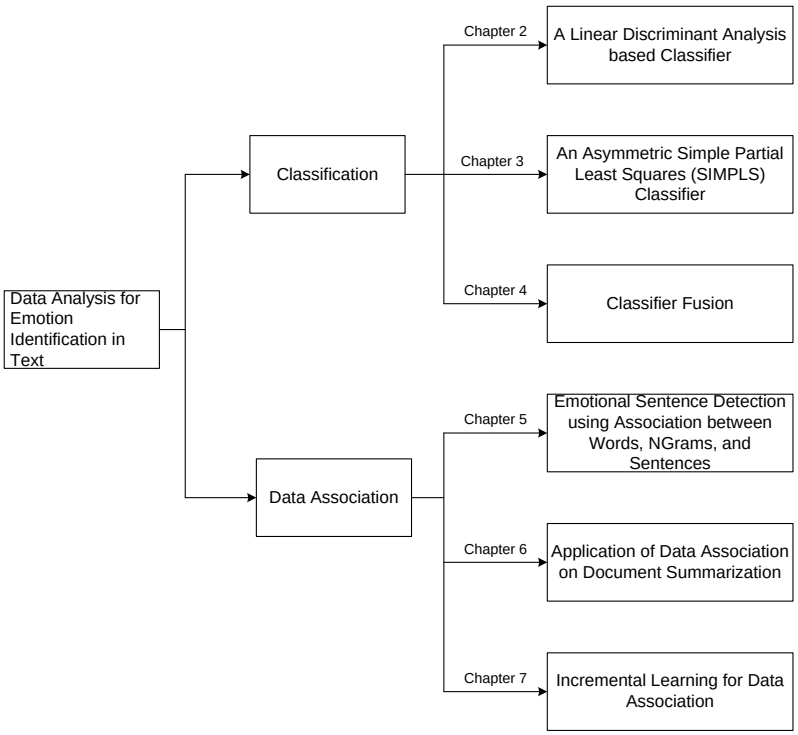


Figure 1.2: Thesis structure.

fication problem is first taken as a classification problem. In Chapter 2, a LDA based classifier is proposed to solve the imbalanced pattern classification problem. An ASimPLS classifier is employed in Chapter 3, which has been successfully applied on other imbalanced pattern classification problems like speaker state classification. To combine different classifiers and features, a classifier fusion system is proposed in Chapter 4. The emotion identification problem is further formulated as a ranking problem that ranks the emotion

scores hidden in the sentences. A mutual reinforcement learning method is proposed in Chapter 5 to solve the ranking problem by utilizing the associations between terms, bigrams, and sentences. The proposed method is applied on document summarization task in Chapter 6. The data association should keep updating with the growth of the training set. The problem is addressed in Chapter 7 to study how to update the data association when new data is appended.

Chapter 2

A Linear Discriminant Analysis based Classifier for Imbalanced Pattern Classification

2.1 Introduction

An intuitive solution to the emotion identification problem is to treat it as a classification problem. Many researchers have reported that the classification of emotions in text is an imbalanced pattern classification problem because there are much more neutral sentences than the emotional sentences in a corpus [2, 3].

In this chapter, a Linear Discriminant Analysis (LDA) based classifier, LDA-Imbalance, is proposed for the imbalanced classification problems. The classifier keeps the advantage of low time complexity like PLS, and improves the performance further by considering more information of the training data. LDA has been widely used in dimension reduction and classification. There is debate about the influence to LDA caused by unbalanced data set [70, 71]. We can demonstrate that the imbalanced data set can affect LDA, but we also agree with [71] that a re-balanced data set cannot guarantee a

better performance than LDA. The proposed classifier for imbalanced pattern classification is described in detail in the following section.

The original LDA method aims to find a projection vector that can maximize the ratio of the between-class scatter matrix to the within-class scatter matrix. We redefine the within-class scatter matrix to increase the influence of minority class for an imbalanced pattern classification problem. To find a projection vector with better discriminant information, our algorithm maximizes the ratio of between-class scatter matrix and within-class scatter matrix, and minimizes the ratio of the variance of one class to the within-class scatter matrix simultaneously. After obtaining the projection matrix, two methods are introduced to classify testing samples into a class using the projection matrix. The algorithm is applied to multiclass classification problems using one-against-rest strategy. The experimental results obtained by the algorithm on several data sets demonstrate that it could obtain competitive results with many existing classifiers on imbalanced data sets. The contributions of this chapter are highlighted as follows:

(i) a LDA-Imbalance classifier is proposed to address the imbalanced pattern classification problem;

(ii) an asymmetric method of classifying testing samples using the projection matrix obtained by LDA-Imbalance is introduced; and

(iii) the LDA-Imbalance method, which is designed for two classes classification problems, is extended to solve multiclass classification problems using one-against-rest strategy.

The rest of this chapter is organized as follows: The proposed classifier is presented in detail in Section 2.2. Section 2.3 describes the experiments on

Table 2.1: Nomenclature.

$\mathbb{R}$	the field of real numbers;
$\mathbb{R}^{N\times M}$	the set of $N \times M$-dimensional real matrices;
$\mathbf{X}$	the feature matrix of a set of samples;
$\mathbf{y}$	a label vector of a set of samples;
$\|\mathbf{x}\|$	the Euclidean norm of a vector $\mathbf{x}$;
$\|\mathbf{X}\|_F$	the Frobenius norm of a matrix $\mathbf{X}$;
$\mathrm{Tr}(\mathbf{X})$	the trace of a matrix $\mathbf{X}$;
$\mathbf{\Sigma_X}$	the covariance matrix of a matrix $\mathbf{X}$;
$\mathbf{S_X}$	the estimation of $\mathbf{\Sigma_X}$;
$\bar{\mathbf{S}}$	the normalized covariance matrix, i.e. $\mathbf{S}/\mathrm{Tr}(\mathbf{S})$;

a synthetic data set and several UCI data sets. The experimental results of applying the method on emotional sentences identification are reported in Section 2.4. We summarize our work in Section 2.5. Key notations used in this chapter are listed in Table 2.1.

2.2 LDA-Imbalance: a LDA based classifier for imbalanced data sets

In this section, we will introduce a LDA based classifier designed for imbalanced pattern classification problems. The classifier first finds a projection matrix which achieves specific optimal objectives, and then predicts the labels of testing samples using the matrix. The classifier is also applied to multiclass classification problems using one-against-rest strategy.

2.2.1 Finding projection matrix

Let $\mathbf{X} \in \mathbb{R}^{N\times M}$ and $\mathbf{y} \in \mathbb{R}^{N\times 1}$ denote the features and labels of the training data respectively, where N is the number of training data, and M is the

dimension of features. In the label vector $\mathbf{y}$, $\mathbf{y}_i = 1$ if a data i belongs to the positive class in a two class classification problem. Otherwise, $\mathbf{y}_i = -1$. The Fisher criterion function applied in LDA is [72]

$$J = \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}} \tag{2.1}$$

where

$$\mathbf{S}_b = \sum_{i=1}^{2} n_i (\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^T \tag{2.2}$$

is the between-class scatter matrix and

$$\mathbf{S}_w = \sum_{i=1}^{2} \sum_{j=1}^{n_i} (\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)(\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)^T \tag{2.3}$$

is the within-class scatter matrix. The parameter n_i is the number of observations in class i; $\bar{\mathbf{x}}_i \in \mathbb{R}^{M \times 1}$ is the mean vector of class i; $\bar{\mathbf{x}} \in \mathbb{R}^{M \times 1}$ is the mean vector of all the observations; and $\mathbf{x}_{ij} \in \mathbb{R}^{M \times 1}$ is the jth observations in class i. The weight $\mathbf{w} \in \mathbb{R}^{M \times 1}$ in (2.1) can be obtained by solving the eigenvalue decomposition problem [36]

$$\mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{w} = \lambda \mathbf{w} \tag{2.4}$$

It has been proved that LDA can obtain statistically optimal solution only when the distributions of observations in different classes satisfy the homoscedastic Gaussian (HOG) model [73], which means that the observations belonging to different classes obey Gaussian distributions with distinct mean vectors but with the same covariance matrix for all the classes. This assumption is seldom satisfied in the real world problems. The estimation

of means and variance matrices are usually not identical with the true ones because of small number of observations. Besides, LDA will also be affected by the class distribution in classification problems on imbalanced data sets. For a two-class classification problem, let positive class denote the class with minority sample number and negative class denote the class with majority sample number. Suppose both of the classes have more than one sample, the within-class scatter matrix is computed as

$$\begin{aligned} \mathbf{S}_w &= \sum_{i=1}^{2}\sum_{j=1}^{n_i}(\mathbf{x}_{ij}-\bar{\mathbf{x}}_i)(\mathbf{x}_{ij}-\bar{\mathbf{x}}_i)^T \\ &= (n_1-1)\mathbf{S}_+ + (n_2-1)\mathbf{S}_- \end{aligned} \tag{2.5}$$

where $n_i > 1, i = 1, 2$ are numbers of samples in the positive and negative classes, and the covariance matrices of positive and negative classes are denoted by $\mathbf{S}_+$ and $\mathbf{S}_-$. We can see that if $n_2 \gg n_1$ in a training data set, $(n_2-1)\mathbf{S}_-$ will "dominate" the computation of $\mathbf{S}_w$, which will affect the classification results as $\mathbf{S}_w$ can represent $(n_2-1)\mathbf{S}_-$ only, while $(n_1-1)\mathbf{S}_+$ is neglected. To alleviate the influence caused by the large number of observations in one class, we add a weighting scheme to compute the within-class scatter matrix $\mathbf{S}_w$,

$$\begin{aligned} \mathbf{S}'_w &= \sum_{i=1}^{2}\frac{1}{n_i-1}\sum_{j=1}^{n_i}(\mathbf{x}_{ij}-\bar{\mathbf{x}}_i)(\mathbf{x}_{ij}-\bar{\mathbf{x}}_i)^T \\ &= \mathbf{S}_+ + \mathbf{S}_- \end{aligned} \tag{2.6}$$

One can see that $\mathbf{S}'_w$ has no relationship with the class distribution n_1 and n_2. Hence, $\mathbf{S}'_w$ is able to describe the covariance of an imbalanced data set more accurately.

To further discriminate the two classes, we set up another optimal objective that aims to minimize the covariance matrix of one class.

$$\max_{\mathbf{W}}(\frac{\mathbf{W}^T(\mathbf{S}_b - \alpha\mathbf{S}_1)\mathbf{W}}{\mathbf{W}^T\mathbf{S}'_w\mathbf{W}}) \tag{2.7}$$

We can rewrite (2.7) as

$$\max_{\mathbf{W}}(\frac{\mathbf{W}^T(\mathbf{S}_b + \alpha\mathbf{S}_1)\mathbf{W}}{\mathbf{W}^T\mathbf{S}'_w\mathbf{W}}) \tag{2.8}$$

where the parameter α for balancing the weights of the two optimizing objectives is set to be negative. The covariance matrix of class 1 is denoted by $\mathbf{S}_1$. Class 1 can be either positive or negative, which means $\mathbf{S}_1$ equals to $\mathbf{S}_+$ or to $\mathbf{S}_-$. Similarly, we use $\mathbf{S}_2$ to represent the covariance matrix of another class. As α is a negative number, one has to minimize $\mathbf{S}_1$ to maximize the whole object function.

We normalize the matrices by $\bar{\mathbf{S}}_b = \mathbf{S}_b/\mathrm{Tr}(\mathbf{S}_b)$, $\bar{\mathbf{S}}_1 = \mathbf{S}_1/\mathrm{Tr}(\mathbf{S}_1)$ and $\bar{\mathbf{S}}_2 = \mathbf{S}_2/\mathrm{Tr}(\mathbf{S}_2)$ to eliminate magnitude variations existing between different matrices. Equation (2.8) can be written as

$$\arg\max_{\mathbf{W}}(\frac{\mathbf{W}^T(\bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1)\mathbf{W}}{\mathbf{W}^T(\bar{\mathbf{S}}_1 + \bar{\mathbf{S}}_2)\mathbf{W}}) \tag{2.9}$$

To find the solution of (2.9), let

$$J(\mathbf{W}) = \frac{\mathbf{W}^T(\bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1)\mathbf{W}}{\mathbf{W}^T(\bar{\mathbf{S}}_1 + \bar{\mathbf{S}}_2)\mathbf{W}} \tag{2.10}$$

we have

$$J(\mathbf{W})\mathbf{W}^T(\bar{\mathbf{S}}_1 + \bar{\mathbf{S}}_2)\mathbf{W} = \mathbf{W}^T(\bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1)\mathbf{W} \tag{2.11}$$

Differentiating both sides with respect to $\mathbf{W}$,

$$\begin{aligned}&\frac{dJ}{d\mathbf{W}}\mathbf{W}^T(\bar{\mathbf{S}}_1+\bar{\mathbf{S}}_2)\mathbf{W}+2J(\mathbf{W})(\bar{\mathbf{S}}_1+\bar{\mathbf{S}}_2)\mathbf{W}\\ =&2(\bar{\mathbf{S}}_b+\alpha\bar{\mathbf{S}}_1)\mathbf{W}\end{aligned}$$

$$\frac{dJ}{d\mathbf{W}}=\frac{2(\bar{\mathbf{S}}_b+\alpha\bar{\mathbf{S}}_1)\mathbf{W}-2J(\mathbf{W})(\bar{\mathbf{S}}_1+\bar{\mathbf{S}}_2)\mathbf{W}}{\mathbf{W}^T(\bar{\mathbf{S}}_1+\bar{\mathbf{S}}_2)\mathbf{W}}$$

Let $\frac{dJ}{d\mathbf{W}}=0$,

$$(\bar{\mathbf{S}}_b+\alpha\bar{\mathbf{S}}_1)\tilde{\mathbf{W}}=J(\tilde{\mathbf{W}})(\bar{\mathbf{S}}_1+\bar{\mathbf{S}}_2)\tilde{\mathbf{W}} \tag{2.12}$$

where $\tilde{\mathbf{W}}$ is the solution matrix. Let a constant

$$\lambda=J(\tilde{\mathbf{W}})=\frac{\tilde{\mathbf{W}}^T(\bar{\mathbf{S}}_b+\alpha\bar{\mathbf{S}}_1)\tilde{\mathbf{W}}}{\tilde{\mathbf{W}}^T(\bar{\mathbf{S}}_1+\bar{\mathbf{S}}_2)\tilde{\mathbf{W}}} \tag{2.13}$$

it can be seen that the projection $\mathbf{W}$ can be obtained by solving the generalized eigenvalue decomposition problem

$$(\bar{\mathbf{S}}_b+\alpha\bar{\mathbf{S}}_1)\mathbf{W}=\lambda(\bar{\mathbf{S}}_1+\bar{\mathbf{S}}_2)\mathbf{W} \tag{2.14}$$

2.2.2 Algorithm properties

Some properties of the proposed algorithm are discussed in this section. As shown above, we normalize the matrices $\mathbf{S}_b$, $\mathbf{S}_1$, and $\mathbf{S}_2$ to eliminate magnitude variations existing between different samples. It is worth noting that

the formula

$$\begin{aligned}\bar{\mathbf{S}}_1 + \bar{\mathbf{S}}_2 &= \bar{\mathbf{S}}_+ + \bar{\mathbf{S}}_- \\ &= \frac{\mathbf{S}_+}{\mathrm{Tr}(\mathbf{S}_+)} + \frac{\mathbf{S}_-}{\mathrm{Tr}(\mathbf{S}_-)} \\ &= \frac{1}{\mathrm{Tr}(\mathbf{S}_+)}(\mathbf{S}_+ + \frac{\mathrm{Tr}(\mathbf{S}_+)}{\mathrm{Tr}(\mathbf{S}_-)}\mathbf{S}_-)\end{aligned} \tag{2.15}$$

is not equivalent with the normalization of $\mathbf{S}'_w$:

$$\bar{\mathbf{S}}'_w = \frac{\mathbf{S}'_w}{\mathrm{Tr}(\mathbf{S}'_w)} = \frac{\mathbf{S}_+ + \mathbf{S}_-}{\mathrm{Tr}(\mathbf{S}_+ + \mathbf{S}_-)} \tag{2.16}$$

Through the normalization of (2.15), the contribution of $\mathbf{S}_+$ and $\mathbf{S}_-$ to the within-class scatter matrix is further balanced by the parameter $\frac{\mathrm{Tr}(\mathbf{S}_+)}{\mathrm{Tr}(\mathbf{S}_-)}$.

The parameter α in our method is used to balance the two optimal objectives. When $\alpha = 0$, our method is a modified LDA method. We can prove that the method becomes a modified Common Spatial Patterns (CSP) method when $\alpha = -\infty$. When $\alpha = -\infty$, we can ignore the first term of the objective function, and the solution of (2.8) is same with the solution of

$$\max_{\mathbf{W}}(\frac{\mathbf{W}^T(\alpha\mathbf{S}_1)\mathbf{W}}{\mathbf{W}^T\mathbf{S}'_w\mathbf{W}}) = \max_{\mathbf{W}}(\alpha\frac{\mathbf{W}^T\mathbf{S}_1\mathbf{W}}{\mathbf{W}^T(\mathbf{S}_1+\mathbf{S}_2)\mathbf{W}}) \tag{2.17}$$

Remark 1. *The solutions of* $\max_{\mathbf{W}}(\frac{\mathbf{W}^T\mathbf{S}_1\mathbf{W}}{\mathbf{W}^T(\mathbf{S}_1+\mathbf{S}_2)\mathbf{W}})$ *are same as the solutions of* $\max_{\mathbf{W}}(\frac{\mathbf{W}^T\mathbf{S}_1\mathbf{W}}{\mathbf{W}^T\mathbf{S}_2\mathbf{W}})$*, which is the objective function of CSP [74] .*

Rewrite

$$\frac{\mathbf{W}^T\mathbf{S}_1\mathbf{W}}{\mathbf{W}^T(\mathbf{S}_1+\mathbf{S}_2)\mathbf{W}} = \frac{1}{1+\frac{\mathbf{W}^T\mathbf{S}_2\mathbf{W}}{\mathbf{W}^T\mathbf{S}_1\mathbf{W}}}, \tag{2.18}$$

it is obvious that those two optimization problems are equivalent to each other.

It is well known that only one projection vector can be obtained using LDA, because the rank of $\mathbf{S}_b$ is 1 for two-class classification problems. More vectors can be obtained by our method as $\text{rank}(\bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1) \leq \text{rank}(\bar{\mathbf{S}}_b) + \text{rank}(\bar{\mathbf{S}}_1) \leq 1+M$, where $\bar{\mathbf{S}}_b \in \mathbb{R}^{M\times M}$ and $\bar{\mathbf{S}}_1 \in \mathbb{R}^{M\times M}$. Hence the maximum rank of $(\bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1)$ is M, and we can get more projection vectors than LDA.

Theorem 1. *Given symmetric matrices $\bar{\mathbf{S}}_1$, $\bar{\mathbf{S}}_2$, and $\bar{\mathbf{S}}_b$ belonging to $\mathbb{R}^{M\times M}$ and a real number α, if rank$(\bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1) = M$ and rank$(\bar{\mathbf{S}}_1 + \bar{\mathbf{S}}_2) = M$, $\mathbf{W}\mathbf{W}^T = (\bar{\mathbf{S}}_1 + \bar{\mathbf{S}}_2)^{-1}$.*

Proof. Let $\mathbf{A} = \bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1$, and $\mathbf{B} = \bar{\mathbf{S}}_1 + \bar{\mathbf{S}}_2$. The projection matrix $\mathbf{W}$ can be obtained by solving the generalized eigenvalue problem:

$$\mathbf{A}\mathbf{w} = \lambda\mathbf{B}\mathbf{w} \tag{2.19}$$

As $\bar{\mathbf{S}}_b$, $\bar{\mathbf{S}}_1$ and $\bar{\mathbf{S}}_2$ are symmetric matrices, both $\mathbf{A}$ and $\mathbf{B}$ are also symmetric matrices. Let $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ and $\mathbf{B} = \mathbf{U}\mathbf{U}^T$ using the Cholesky factorization. We can rewrite (2.19) as

$$\mathbf{U}^{-1}\mathbf{L}\mathbf{L}^T\mathbf{U}^{-T}\mathbf{U}^T\mathbf{w} = \lambda\mathbf{U}^T\mathbf{w} \tag{2.20}$$

Let $\mathbf{v} = \mathbf{U}^T\mathbf{w}$, we have

$$\mathbf{U}^{-1}\mathbf{L}\mathbf{L}^T\mathbf{U}^{-T}\mathbf{v} = \lambda\mathbf{v} \tag{2.21}$$

It can be seen that $\mathbf{U}^{-1}\mathbf{L}\mathbf{L}^T\mathbf{U}^{-T}$ is still a symmetric matrix. Hence $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ under the assumption that $\mathbf{A}$ and $\mathbf{B}$ are full rank matrices, where the column vectors of $\mathbf{V}$ are solutions of (2.21). Because $\mathbf{v} = \mathbf{U}^T\mathbf{w}$, one can deduce that

$$(\mathbf{U}^T\mathbf{W})^T(\mathbf{U}^T\mathbf{W}) = \mathbf{W}^T\mathbf{U}\mathbf{U}^T\mathbf{W} = \mathbf{I} \tag{2.22}$$

Multiply pseudo inverses of $\mathbf{W}^T$ and $\mathbf{W}$ respectively, we have

$$\begin{aligned} \mathbf{U}\mathbf{U}^T &= \mathbf{W}^{T+}\mathbf{W}^{+} \\ &= (\mathbf{W}\mathbf{W}^T)^{-1}\mathbf{W}\mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1} \\ &= (\mathbf{W}\mathbf{W}^T)^{-1} \end{aligned} \tag{2.23}$$

and

$$\mathbf{W}\mathbf{W}^T = (\mathbf{U}\mathbf{U}^T)^{-1} = \mathbf{B}^{-1} = (\bar{\mathbf{S}}_1 + \bar{\mathbf{S}}_2)^{-1} \tag{2.24}$$

□

2.2.3 Classification using the projection matrix

For two classes classification problems, the label of a testing sample $\mathbf{x} \in \mathbb{R}^{M\times 1}$ is predicted by

$$y = \text{sign}(f(\mathbf{W}, \mathbf{x})) \tag{2.25}$$

where $f(\mathbf{W}, \mathbf{x})$ is a function of the projection matrix $\mathbf{W}$ and a sample $\mathbf{x}$. If $f(\mathbf{W}, \mathbf{x}) \geq 0$, $y = 1$, otherwise $y = -1$.

2.2.3.1 Symmetric method for two classes classification

The first method predicts the label of a testing sample $\mathbf{x}$ by comparing distances between $\mathbf{x}$ and the centers of positive and negative classes [75], i.e.

$$f(\mathbf{W}, \mathbf{x}) = \|\mathbf{W}^T\mathbf{x} - \mathbf{W}^T\bar{\mathbf{x}}_2\| - \|\mathbf{W}^T\mathbf{x} - \mathbf{W}^T\bar{\mathbf{x}}_1\| \tag{2.26}$$

where $\mathbf{W} \in \mathbb{R}^{M \times k}$ is the projection matrix, and $k \leq M$ is an integer. For a vector $\mathbf{p} = \mathbf{W}^T\mathbf{x} - \mathbf{W}^T\bar{\mathbf{x}}_1 = \mathbf{W}^T(\mathbf{x} - \bar{\mathbf{x}}_1)$,

$$\|\mathbf{p}\| = \sqrt{\mathbf{p}^T\mathbf{p}} = \sqrt{(\mathbf{x} - \bar{\mathbf{x}}_1)^T\mathbf{W}\mathbf{W}^T(\mathbf{x} - \bar{\mathbf{x}}_1)} \tag{2.27}$$

where $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ are mean vectors of positive and negative training samples. According to (2.27) and Theorem 1, one can deduce that if $k = M$, the classification results will be independent of $\bar{\mathbf{S}}_b$ and $\bar{\mathbf{S}}_1$ in (2.14). The borderline of this method is composed by the points lying in the middle of two centers. We call this method Symmetric Method.

2.2.3.2 Asymmetric method for two classes classification

The classifying function of Asymmetric Method is defined as

$$f(\mathbf{W}, \mathbf{x}) = \mathbf{x}^T\mathbf{W}_x\mathbf{W}_y^+ - b \tag{2.28}$$

where $\mathbf{W}_x$ is same as $\mathbf{W}$, $\mathbf{W}_y$ is the projection vector for labels whose elements are obtained by (2.29), and $\mathbf{W}_y^+$ is the pseudo inverse of $\mathbf{W}_y$. The bias b is used to adjust the borderline to improve system performance. The elements of the projection vector for labels are calculated by [76]

$$\mathbf{w}_y = \frac{1}{\lambda}\mathbf{S}_\mathbf{y}^{-1}\mathbf{S}_{\mathbf{yX}}\mathbf{w}_x = \frac{1}{\lambda}\mathbf{S}_\mathbf{y}^{-1}\mathbf{S}_{\mathbf{Xy}}^T\mathbf{w}_x \tag{2.29}$$

where λ is the eigenvalue corresponding to the eigenvectors obtained by (2.14) and $\mathbf{w}_x$ is a column vector of $\mathbf{W}_x$. For a normalized feature matrix $\mathbf{X}$ and a

label vector $\mathbf{y}$,

$$\begin{aligned} \mathbf{S_X} &= \tfrac{1}{N-1}\mathbf{X}^T\mathbf{X} \\ \mathbf{S_y} &= \tfrac{1}{N-1}\mathbf{y}^T\mathbf{y} \\ \mathbf{S_{Xy}} &= \tfrac{1}{N-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{2.30}$$

are the estimation of covariance matrices of $\mathbf{X}$, $\mathbf{y}$, and the covariance matrix of $\mathbf{X}$ and $\mathbf{y}$.

The method is inspired by Canonical Correlation Analysis (CCA) which aims to find two vectors $\mathbf{w}_x \in \mathbb{R}^{M\times 1}$ and $\mathbf{w}_y \in \mathbb{R}^{M'\times 1}$ that maximize $\mathrm{corr}(\mathbf{X}\mathbf{w}_x, \mathbf{Y}\mathbf{w}_y)$ where $\mathbf{Y} \in \mathbb{R}^{N\times M'}$. For two vectors $\mathbf{x} \in \mathbb{R}^{M\times 1}$ and $\mathbf{y} \in \mathbb{R}^{1\times M'}$, if $\mathbf{x}^T\mathbf{w}_x = \mathbf{y}\mathbf{w}_y$, we have

$$\mathbf{y} = \mathbf{x}^T\mathbf{w}_x\mathbf{w}_y^+ \tag{2.31}$$

where $\mathbf{w}_y^+$ is the pseudo inverse of $\mathbf{w}_y$. For two classes classification problems, $M' = 1$ as $\mathbf{Y}$ is a vector of samples' labels. Hence, $\mathbf{w}_y$ is a scalar. For a testing sample, if $\mathbf{x}^T\mathbf{w}_x\mathbf{w}_y^+ \geq 0$ in (2.31), the label $y = 1$. Otherwise $y = -1$. A method for predicting labels of testing samples can be conducted by

$$f'(\mathbf{W}, \mathbf{x}) = \mathbf{x}^T\mathbf{W}_x\mathbf{W}_y^+ \tag{2.32}$$

which is the first addend of (2.28).

It is necessary to introduce a bias b for the classifying function (2.32) as the classifier using $f'(\mathbf{W}, \mathbf{x})$ will lead to a high classification accuracy of minority class while a low accuracy of majority class. A two dimensional binary sample data set is shown in Fig. 2.1, where the majority class samples are signed by circles, and the minority ones are expressed by stars. The x-axis of

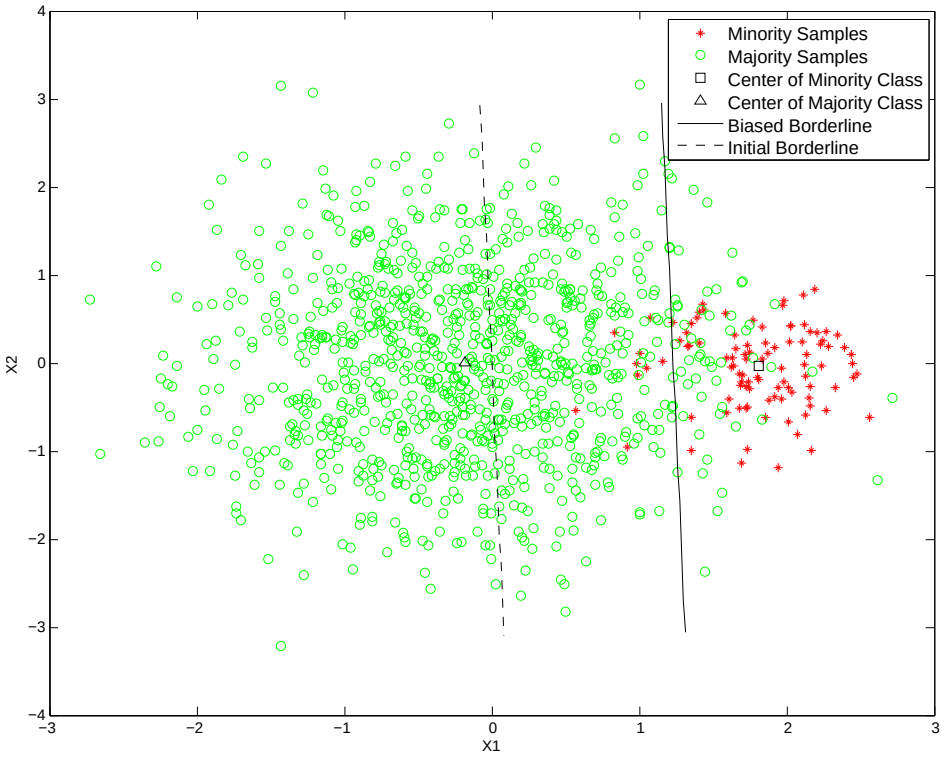


Figure 2.1: A two dimensional artificial data set where the majority class data is denoted by circle and the minority class is denoted by star.

the figure is the first dimension of samples' coordinate, and the y-axis denotes the second dimension. The dashed line L denotes the borderline of classification $\mathbf{x}^T\mathbf{W}_x\mathbf{W}_y^+ = 0$. One can see that L goes through the point $O(0,0)$ which is apart from the center of the minority class. According to (2.32), points located on the left side of L are classified as negative class, and those located on the right side are classified as positive class. It can be seen that this classification method can label most of the minority class data correctly, but it will misclassify many samples belonging to the majority class. To solve this problem, an intuitive idea is to move the the borderline in Fig. 2.1 towards the center of minority class. Similar to the method in [36] applied to a PLS classifier, a bias is calculated to decide how much shall the borderline

be moved. The centers of minority and majority classes can be expressed by $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$. We can estimate two circles with radius $\mathbf{r}_1 = \text{std}(\mathbf{X}_1)$ and $\mathbf{r}_2 = \text{std}(\mathbf{X}_2)$ where $\mathbf{X}_1$ and $\mathbf{X}_2$ are feature matrices of positive and negative samples respectively. The new line L' goes through the point P that the ratio of distance between the point and the positive center to the distance between the point and the negative center is equal with the ratio of the positive class radii to the negative class radii. Hence, the offset b can be estimated as

$$b = (\bar{\mathbf{x}}_1 - (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2) \cdot \mathbf{r}_1./(\mathbf{r}_1 + \mathbf{r}_2))^T \mathbf{W}_x \mathbf{W}_y^+ \tag{2.33}$$

where $\mathbf{v}_1./\mathbf{v}_2$ is the dot division of two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, which is the element by element division. We can see that the solid line in Fig. 2.1 which is the biased borderline is closer to the minority class than the dashed line, which may increase the accuracy of classifying the majority class without losing the accuracy of classifying minority class. The function of $\mathbf{W}$ and a testing sample $\mathbf{x}$ is

$$f(\mathbf{W}, \mathbf{x}) = \mathbf{x}^T \mathbf{W}_x \mathbf{W}_y^+ - b \tag{2.34}$$

It can be seen that the line goes through point $(\bar{\mathbf{x}}_1 - (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2) \cdot \mathbf{r}_1./(\mathbf{r}_1 + \mathbf{r}_2))$.

2.2.3.3 Multiclass classification

One-against-rest strategy is employed to extend our method to solve multi-class classification problems as our method aims to minimize the covariance matrix of only one class and it is not able to be applied to multiclass problems directly. For a k classes problem, the label of a testing sample $\mathbf{x}$ is predicted

by

$$y = \arg\max_i P_i(\mathbf{x}) \tag{2.35}$$

where $i = 1, \ldots, k$, and P_i is the probability of the sample belonging to class i. We generate k label sets for the training samples by labeling the samples belonging to class i as 1, and all the other samples as -1. A classifier is run for k times, and a probability belonging to class i can be obtained for each sample.

Two methods for two classes classification were introduced in previous sections. For Symmetric Method, the probability of a sample $\mathbf{x}$ belonging to class i is calculated by

$$P_i(\mathbf{x}) = \begin{cases} \frac{d_2}{d_1+d_2} & , d_1 + d_2 \neq 0 \\ 0.5 & , d_1 + d_2 = 0 \end{cases} \tag{2.36}$$

where $d_1 = \|\mathbf{W}^T\mathbf{x} - \mathbf{W}^T\bar{\mathbf{x}}_i\|$ and $d_2 = \|\mathbf{W}^T\mathbf{x} - \mathbf{W}^T\bar{\mathbf{x}}_j\|$, $\bar{\mathbf{x}}_i$ is the mean vector of training samples belonging to class i, and $\bar{\mathbf{x}}_j$ is the mean vectors of those not belonging to class i. For the Asymmetric Method,

$$P_i(\mathbf{x}) = \mathbf{x}^T\mathbf{W}_{xi}\mathbf{W}_{yi}^{+} - b_i \tag{2.37}$$

where $\mathbf{W}_{xi}$, $\mathbf{W}_{yi}$ and b_i are obtained using the label sets of class i.

2.3 Experimental evaluation

In this section, we show the benefits of our method using a synthetic data set first. Further, the experimental results obtained on several UCI bench-

mark data sets [77] are reported. The feature data is normalized by mean variance normalization method [78] at the beginning of all the algorithms. True Positive Rate (TPR), True Negative Rate (TNR), unweighted accuracy (UA), weighted accuracy (WA), precision, recall, F1, G-Mean and AUC are selected to measure the system performance. The code of AUC employed in this chapter is from Statistical Pattern Recognition Toolbox written by Vojtech Franc and Vaclav Hlavac[1]. The relative definitions of TPR and TNR are shown below:

$$\begin{aligned}
&\text{True Positive Rate (TPR)} = \tfrac{\text{TP}}{\text{TP+FN}} \\
&\text{True Negative Rate (TNR)} = \tfrac{\text{TN}}{\text{TN+FP}} \\
&\text{UA} = \tfrac{\text{TPR+TNR}}{2} \\
&\text{WA} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}
\end{aligned} \tag{2.38}$$

where TP, FN, TN, and FP are defined in Table 2.2. The metrics including precision, recall, F1, G-Mean and AUC are also employed to evaluate the methods.

$$\begin{aligned}
&\text{precision} = \tfrac{\text{TP}}{\text{TP+FP}} \\
&\text{recall} = \tfrac{\text{TP}}{\text{TP+FN}} \\
&\text{F1} = 2\tfrac{\text{precision}\cdot\text{recall}}{\text{precision+recall}} \\
&\text{sensitivity} = \tfrac{\text{TP}}{\text{TP+FN}} \\
&\text{specificity} = \tfrac{\text{TN}}{\text{TN+FP}} \\
&\text{G-Mean} = \sqrt{\text{sensitivity} \cdot \text{specificity}}
\end{aligned} \tag{2.39}$$

[1] http://cmp.felk.cvut.cz/cmp/software/stprtool/

Table 2.2: Definition of TP, FN, FP, and TN.

	Predicted Positive	Predicted Negative
Actual Positive	TP (True Positive)	FN (False Negative)
Actual Negative	FP (False Positive)	TN (True Negative)

2.3.1 A synthetic data set

We generate a data set which contains 1000 negative samples and 100 positive samples. Both the positive and negative samples satisfy Gaussian distribution, and $\mu_{pos} = (2, 4)$, $\mu_{neg} = (0, 0)$, $\sum_{pos} = \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix}$, $\sum_{neg} = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$. A 10-fold cross validation is conducted, and we compare the classification results of the following methods: SVM, LDA, LDA-Imbalance, and PLS. For SVM, the radial basis function (rbf) kernel function is applied, and the other settings are default settings of libsvm-3.1[2]. In this experiment, we use the Asymmetric Method introduced in Section 2.2.3.2 to predict the labels for LDA, LDA-Imbalance, and PLS, as the borderline obtained by this method is easier to be identified than those obtained by the Symmetric Method. Here we also evaluate the PLS method because it has been shown that LDA has close relationship with PLS in [79]. PLS can generate a borderline for classification problems disregarding the covariance matrices of training samples. Hence, PLS would not be affected by the imbalanced data sets. There are many different algorithms of PLS, and we use the method in [79].

The classification results are shown in Table 2.3, where the highest value of each row is highlighted in bold faces. It can be seen that TNR obtained by SVM is the highest of all the methods, but its TPR is only 36.5%, which is the lowest among the methods. The results demonstrate that SVM failed in

[2]http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Table 2.3: Classification results on a synthetic data set.

	SVMs	LDA	LDA-Imbalance	PLS
TPR	0.365	0.846	0.853	**0.862**
TNR	**0.991**	0.843	0.892	0.822
UA	0.678	0.844	**0.873**	0.842
WA	**0.933**	0.843	0.888	0.825
Precision	**0.810**	0.350	0.439	0.328
Recall	0.365	0.846	0.853	**0.862**
F1	0.483	0.488	**0.572**	0.468
G-Mean	0.588	0.842	**0.870**	0.840
AUC	**0.984**	0.965	0.974	0.965
Time	0.021	0.004	0.004	**0.002**

identifying the points in minority class because of the imbalance of the data set. LDA is able to obtain better results than SVM for positive samples. The TPR obtained by PLS is higher than LDA, while the TNR decreased 2.1%. The LDA-Imbalance method is able to achieve a better TNR and a better TPR than LDA at the same time. Processing time for SVM was more than the other methods, and PLS uses the least time.

We show the data distribution and the borderlines obtained by the algorithms in Fig. 2.2. In the figure, the stars represent training and testing positive samples, and circles denote negative ones. The solid line, the dashed line, and the dotted line represent the borderlines obtained by LDA-Imbalance, LDA, and PLS methods respectively. From the figure, one can see that the borderlines obtained by LDA and PLS are close to the direction of maximum variance of the majority class, while the one obtained by LDA-Imbalance method is able to represent the variance direction of the minority class better. All the three borderlines go through a common point which is the bias discussed in Section 2.2.3.2.

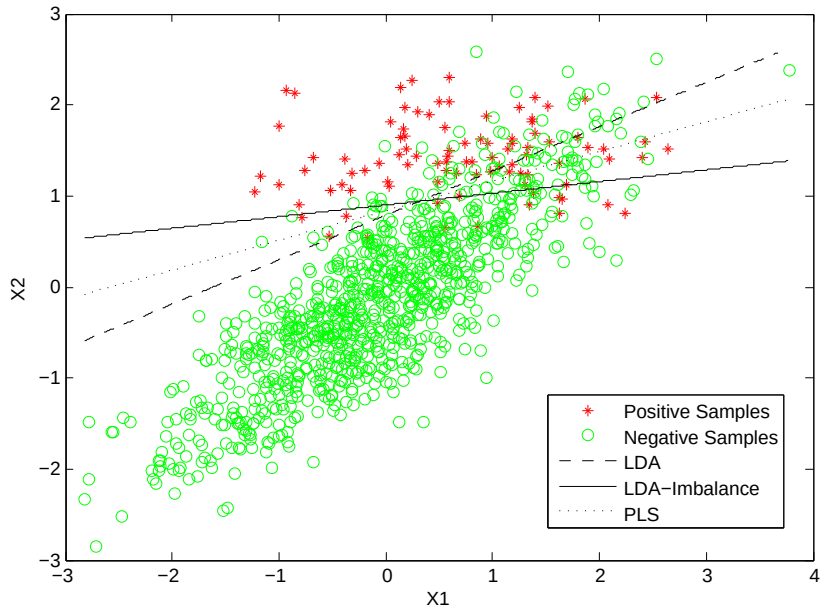


Figure 2.2: Classification on a synthetic data set.

2.3.2 Evaluation using UCI data sets

To further demonstrate the efficiency of the proposed method, we conduct more experiments on UCI data sets which have been widely used in testing of machine learning algorithms [27, 36, 80]. The proposed algorithm in this chapter is evaluated by both two classes classification and multi-class classification problems.

2.3.2.1 Two classes classification

Eight UCI data sets used in our experiments are downloaded from the UCI website[3], and the details of the data sets are shown in Table 2.4. All these data sets are imbalanced data sets, and the highest ratio of majority sample number to the minority sample number is 4141/36 which is about 115/1 in the Abalone data set. The following methods are compared in this ex-

[3]http://archive.ics.uci.edu/ml/

Table 2.4: Detail information of data sets used for two classes classification problem.

Data Set Name	Data Number	Positive/Negative	Feature Dimension
Abalone	4177	36/4141	8
Breast cancer(original)	699	241/458	8
CMC	1473	333/1140	9
Haberman	306	81/225	3
Hypothyroid	3163	151/3012	25
Pima	768	268/500	8
SPECT	267	55/212	22
Yeast	1484	51/1433	8

periment: SVM, SVM with Synthetic Minority Over-sampling TEchnique (SMOTE) [23] technology, SVM with a cost sensitive model (SVMCost) [81], APLS [36], LDA and LDA-Imbalance. Same as in last section, the RBF kernel function is applied for SVM here. The minority class is oversampled by SMOTE to the same number of majority samples in the training set. The code of SVMCost method is in the SVM-Light package[4]. The MATLAB code of APLS method is same with that used in [36], and a maximum iteration number is added to ensure the loop ends. Both the Symmetric Method and Asymmetric Method are employed in the experiments.

Ten-fold cross validation experiments are conducted on all the eight data sets, and the results are shown in Table 2.5. In the table, LDA and LDA-Imbalance denote the results obtained by Symmetric Method, and LDA-A and LDA-Imbalance-A denote the results obtained by the Asymmetric Method. All the criteria are average values of the results obtained in the experiments. In the table, the highest value of each evaluation criterion is highlighted. The methods proposed in this chapter: LDA-A, LDA-Imbalance, and LDA-Imbalance-A could obtain competitive results on all data sets with

[4]http://svmlight.joachims.org/

existing methods. For most of the data sets, SVM with SMOTE technology is able to generate much better results than SVM only. However, it consumes more time than other methods on most data sets. SVMCost could obtain very good results on all the data sets. Similar to SVMSmote, the long processing time is a disadvantage of SVMCost. APLS method obtained best AUC value on 2 of the 8 data sets, which means on these two data sets, the information of covariance matrices is a disturbance to the classification. LDA method is able to generate competitive results with APLS and SMOTE. The LDA-Imbalance method obtained better UA values than the LDA method on 4 of the 8 data sets, and obtained competitive AUC values on 2 data sets, which proves the benefit of the proposed weighted LDA method. The LDA-A and LDA-Imbalance-A methods won 4 champions of AUC values and 4 champions of F1 values on the 8 data sets, which demonstrates that the borderline is adjusted by considering the projection matrix $\mathbf{W_y}$. From the table, one can also see that the LDA and LDA-A methods consume much less time than other methods. It is worth noting that the parameter α is set to be -0.5 for LDA-Imbalance and LDA-Imbalance-A methods. Better results could be obtained by adjusting the parameter value.

To give a deeper look at the methods, the average TPR and TNR obtained by the methods are also listed in Table 2.5. One can see that SVM obtains 100% TNR and 0% TPR on Abalone and Yeast data sets, which means the positive samples are taken as noise and are neglected in the learning phase. The results obtained by SVM with SMOTE technology are much more balanced than those obtained by SVM. APLS can obtain balanced TPR and TNR on all data sets except Abalone. The TPR obtained by LDA-Imbalance is bigger than those got by LDA method on 6 of the 8 data sets,

Table 2.5: Comparison of system performance of all methods.

	Methods	TPR	TNR	WA	UA	Precision	Recall	F1	G-Mean	AUC	Time
	SVM	0.000	**1.000**	**0.991**	0.500	0.000	0.000	0.000	0.000	0.684	0.102
	SVMSmote	0.838	0.575	0.577	0.707	0.017	0.838	0.033	0.679	0.780	1.032
	SVMCost	0.865	0.890	0.890	0.877	0.063	0.865	0.116	0.867	0.918	1.619
	APLS	**0.942**	0.674	0.676	0.808	0.025	**0.942**	0.048	0.795	0.857	0.015
Abalone	LDA	0.826	0.921	0.920	0.873	**0.081**	0.826	**0.144**	0.856	0.918	0.017
	LDA-I	0.899	0.894	0.894	**0.897**	0.068	0.899	0.124	**0.892**	0.926	0.020
	LDA-A	0.861	0.912	0.912	0.886	0.076	0.861	0.138	0.879	**0.927**	0.014
	LDA-I-A	0.882	0.898	0.898	0.890	0.070	0.882	0.127	0.881	0.924	0.014
	SVM	0.403	**0.941**	**0.846**	0.672	**0.587**	0.403	0.459	0.605	0.705	0.015
	SVMSmote	0.818	0.771	0.780	0.795	0.424	0.818	0.551	0.790	0.799	0.063
	SVMCost	**0.869**	0.791	0.804	**0.830**	0.463	**0.869**	0.597	**0.828**	**0.832**	0.306
	APLS	0.827	0.800	0.804	0.814	0.461	0.827	0.586	0.812	0.814	0.003
BCO	LDA	0.824	0.810	0.811	0.817	0.472	0.824	0.594	0.815	0.821	0.003
	LDA-I	0.838	0.809	0.813	0.824	0.475	0.838	**0.600**	0.822	0.827	0.003
	LDA-A	0.841	0.798	0.805	0.820	0.463	0.841	0.591	0.818	0.824	0.003
	LDA-I-A	0.858	0.799	0.808	0.828	0.469	0.858	**0.600**	0.826	0.830	0.003
	SVM	0.058	**0.985**	**0.775**	0.521	**0.528**	0.058	0.102	0.218	0.487	0.107
	SVMSmote	**0.709**	0.608	0.631	0.659	0.346	**0.709**	0.463	0.655	**0.649**	0.301
	SVMCost	0.683	0.636	0.646	0.659	0.354	0.683	0.463	0.658	0.643	0.308
	APLS	0.681	0.641	0.650	0.661	0.356	0.681	0.466	0.660	0.641	0.005
CMC	LDA	0.696	0.624	0.640	0.660	0.351	0.696	0.464	0.658	0.643	0.006
	LDA-I	0.707	0.616	0.636	0.661	0.349	0.707	0.465	0.659	0.646	0.006
	LDA-A	0.683	0.638	0.647	0.660	0.355	0.683	0.465	0.659	0.643	0.006
	LDA-I-A	0.695	0.630	0.644	**0.662**	0.354	0.695	**0.467**	**0.661**	0.646	0.006
	SVM	0.215	**0.928**	0.736	0.571	0.531	0.215	0.287	0.407	0.532	0.006
	SVMSmote	**0.607**	0.665	0.646	0.636	0.396	**0.607**	0.469	0.629	0.598	0.022
	SVMCost	0.351	0.894	**0.748**	0.623	**0.544**	0.351	0.414	0.544	0.579	0.141
	APLS	0.517	0.813	0.732	**0.665**	0.498	0.517	**0.495**	0.642	0.633	0.001
Hab	LDA	0.496	0.831	0.740	0.664	0.512	0.496	0.492	0.632	0.633	0.001
	LDA-I	0.490	0.807	0.720	0.648	0.474	0.490	0.471	0.617	0.619	0.001
	LDA-A	0.517	0.813	0.732	**0.665**	0.498	0.517	**0.495**	**0.643**	**0.634**	0.001
	LDA-I-A	0.523	0.780	0.708	0.652	0.461	0.523	0.479	0.633	0.625	0.001
	SVM	0.679	**0.995**	**0.979**	0.837	**0.865**	0.679	**0.750**	0.818	0.995	0.099
	SVMSmote	0.917	0.610	0.625	0.764	0.108	0.917	0.191	0.747	0.964	0.454
	SVMCost	0.959	0.959	0.959	**0.959**	0.542	0.959	0.684	**0.959**	0.995	0.392
	APLS	**0.983**	0.933	0.936	0.958	0.428	**0.983**	0.588	0.958	**0.996**	0.054
Hyp	LDA	0.720	0.987	0.973	0.853	0.732	0.720	0.711	0.839	0.993	0.025
	LDA-I	0.945	0.933	0.933	0.939	0.413	0.945	0.568	0.938	0.993	0.026
	LDA-A	0.911	0.941	0.940	0.926	0.522	0.911	0.649	0.902	0.994	0.024
	LDA-I-A	0.851	0.954	0.949	0.902	0.564	0.851	0.662	0.882	0.993	0.024
	SVM	0.549	0.869	0.754	0.709	0.688	0.549	0.605	0.688	0.709	0.027
	SVMSmote	**0.809**	0.687	0.728	**0.748**	0.580	**0.809**	**0.671**	0.744	0.736	0.084
	SVMCost	0.555	**0.880**	**0.764**	0.718	**0.710**	0.555	0.618	0.697	0.717	0.195
	APLS	0.728	0.765	0.749	0.746	0.623	0.728	0.666	0.745	**0.741**	0.003
Pima	LDA	0.713	0.778	0.752	0.745	0.631	0.713	0.664	0.743	0.739	0.003
	LDA-I	0.715	0.776	0.752	0.745	0.630	0.715	0.664	0.743	0.739	0.003
	LDA-A	0.732	0.764	0.750	**0.748**	0.623	0.732	0.668	**0.746**	**0.741**	0.003
	LDA-I-A	0.731	0.762	0.748	0.747	0.621	0.731	0.666	0.745	0.740	0.003
	SVM	0.456	**0.923**	**0.823**	0.689	**0.595**	0.456	0.484	0.616	0.728	0.004
	SVMSmote	0.802	0.725	0.737	**0.763**	0.427	0.802	**0.537**	0.751	**0.756**	0.018
	SVMCost	0.743	0.769	0.758	0.756	0.447	0.743	0.535	0.746	0.751	0.078
	APLS	**0.812**	0.714	0.727	**0.763**	0.419	**0.812**	0.531	**0.755**	0.752	0.006
SPECT	LDA	0.797	0.712	0.725	0.755	0.414	0.797	0.524	0.747	0.748	0.002
	LDA-I	0.780	0.700	0.713	0.740	0.400	0.780	0.510	0.729	0.737	0.003
	LDA-A	0.740	0.762	0.754	0.751	0.441	0.740	0.533	0.740	0.750	0.002
	LDA-I-A	0.735	0.769	0.759	0.752	0.452	0.735	**0.537**	0.740	0.751	0.003
	SVM	0.000	**1.000**	**0.966**	0.500	0.000	0.000	0.000	0.000	0.536	0.023
	SVMSmote	**0.830**	0.714	0.718	0.772	0.094	**0.830**	0.166	0.764	0.763	0.155
	SVMCost	0.773	0.869	0.865	**0.821**	**0.171**	0.773	**0.273**	0.812	**0.821**	0.585
	APLS	0.808	0.820	0.820	0.814	0.140	0.808	0.233	0.808	0.818	0.005
Yeast	LDA	0.771	0.860	0.856	0.815	0.163	0.771	0.262	0.807	0.819	0.006
	LDA-I	0.775	0.853	0.850	0.814	0.157	0.775	0.254	0.806	0.813	0.006
	LDA-A	0.784	0.847	0.845	0.816	0.154	0.784	0.251	0.808	0.819	0.006
	LDA-I-A	0.794	0.846	0.844	0.820	0.153	0.794	0.250	**0.813**	**0.821**	0.006

which proves that the LDA-Imbalance can alleviate the influence on TPR caused by imbalance.

In the proposed LDA-Imbalance method, we can either minimize the covariance matrix of the positive class or minimize the covariance of the negative one. To compare the different optimal objects, we run the algorithm twice by minimizing both the positive covariance and the negative covariance. As discussed in Section 2.2, the dimension of the projection matrix $\mathbf{W}$ obtained by our method is possible to be $M \times M$. To demonstrate the influence to the classification results caused by the dimension of $\mathbf{W}$, we show the AUC values obtained using the first k columns of $\mathbf{W}$ method where $k = 1, \ldots, M$ in Figs. 2.3. In the figure, the horizontal axis denotes the number k, and the vertical axis presents the average AUC values obtained by LDA-Imbalance-A. The line with square points denotes the performance obtained by minimizing the covariance matrix of negative samples, and the line with '+' points denotes the performance obtained by minimizing the positive covariance. One can see that the best AUC values of almost all the data sets are obtained by using the first five dimensions of $\mathbf{W}$, which indicates that the most important information of the projection matrix is stored in the first several columns. One can see that the performances achieved by minimizing the two covariance matrices are same at the maximum dimension on all the data sets except Hypothyroid, which has testified Theorem 1. For the Hypothyroid data set, we have $\text{rank}(\bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1) = 18$ while $(\bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1) \in \mathbb{R}^{25\times 25}$, where $(\bar{\mathbf{S}}_b + \alpha\bar{\mathbf{S}}_1)$ is defined in (2.14).

We can see that AUC values obtained by minimizing positive and negative covariance matrices are very different. The average norms of the positive and negative covariance matrices of different data sets are shown in Table 2.6.

Table 2.6: Average norms of positive and negative covariance matrices of training samples in different data sets.

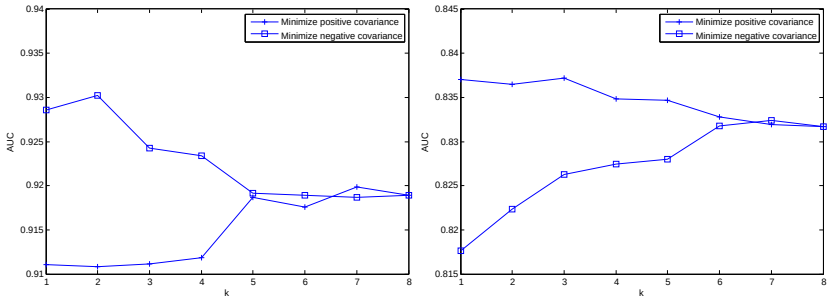
	Abalone	B. C.(Original)	CMC	Haberman
$\| \bar{\mathbf{S}}_+ \|_F$	0.781	0.519	0.399	0.605
$\| \bar{\mathbf{S}}_- \|_F$	0.782	0.520	0.399	0.605
	Hypothyroid	Pima	SPECT	Yeast
$\| \bar{\mathbf{S}}_+ \|_F$	0.523	0.400	0.356	0.485
$\| \bar{\mathbf{S}}_- \|_F$	0.522	0.400	0.355	0.484

However, the data shows that the values of positive and negative norms are almost same. We will conduct further experiments to investigate which matrix should be minimized.

There is only one parameter α in our method, which is a balance of the two optimal objects in (2.14). When $\alpha = 0$, our method is a normalized LDA method; when $\alpha \to \infty$, our method becomes a simplified CSP method. To demonstrate the influence of α, we show the system performance with

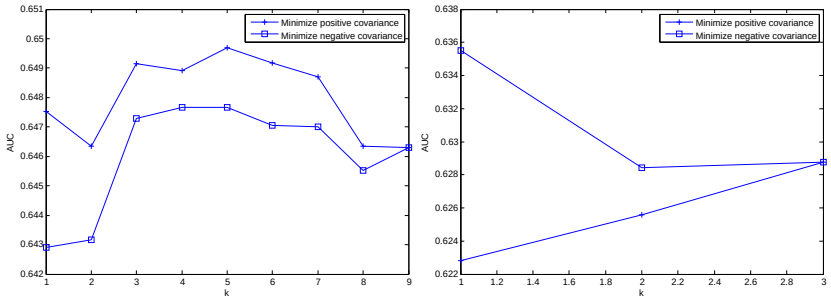
$$\begin{aligned} \alpha = [&-10, -8, -6, -4, -2, -1, -0.8, -0.6, -0.4, \\ &-0.2, 0, 0.2, 0.4, 0.6, 0.8, 1, 2, 4, 6, 8, 10] \end{aligned} \tag{2.40}$$

on all the eight data sets in Figure 2.4. Here we use the first dimension of the projection matrices $\mathbf{W}_x$ and $\mathbf{W}_y$. For every data set, we minimize the covariance of the class which can generate the better performance according to Fig. 2.3. One can see from Fig. 2.4 that the performance of LDA-Imbalance and LDA-Imbalance-A methods have similar trends. The system performance increases when $\alpha < 0$, and it reaches to the top when $\alpha \geq -6$. The AUC value becomes lower when α is too big or too small. For the Yeast data set, the $AUC_{\alpha=10} \approx AUC_{\alpha=-2}$. We did another experiments with more α values, and the results are shown in Fig. 2.4i. We can see that the AUC value



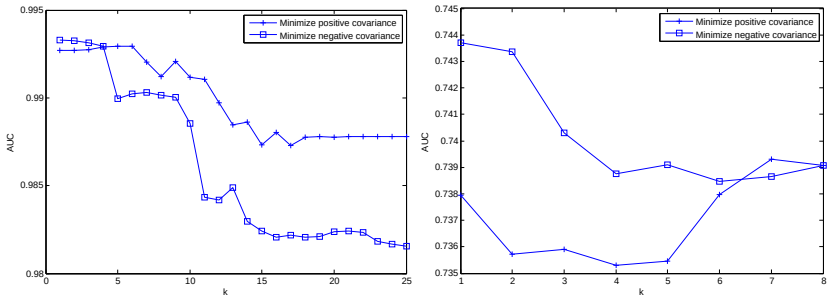
(a) Abalone

(b) BCO



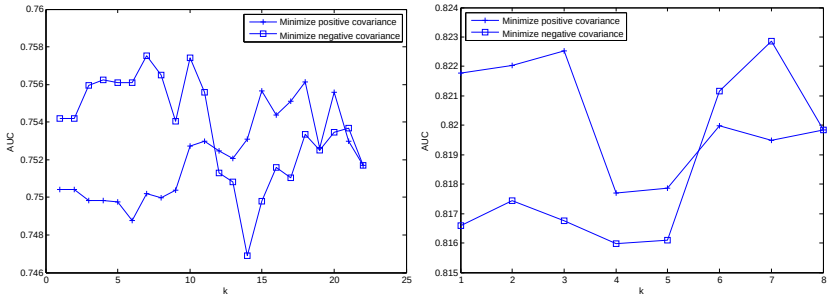
(c) Cmc

(d) Haberman



(e) Hypothyroid

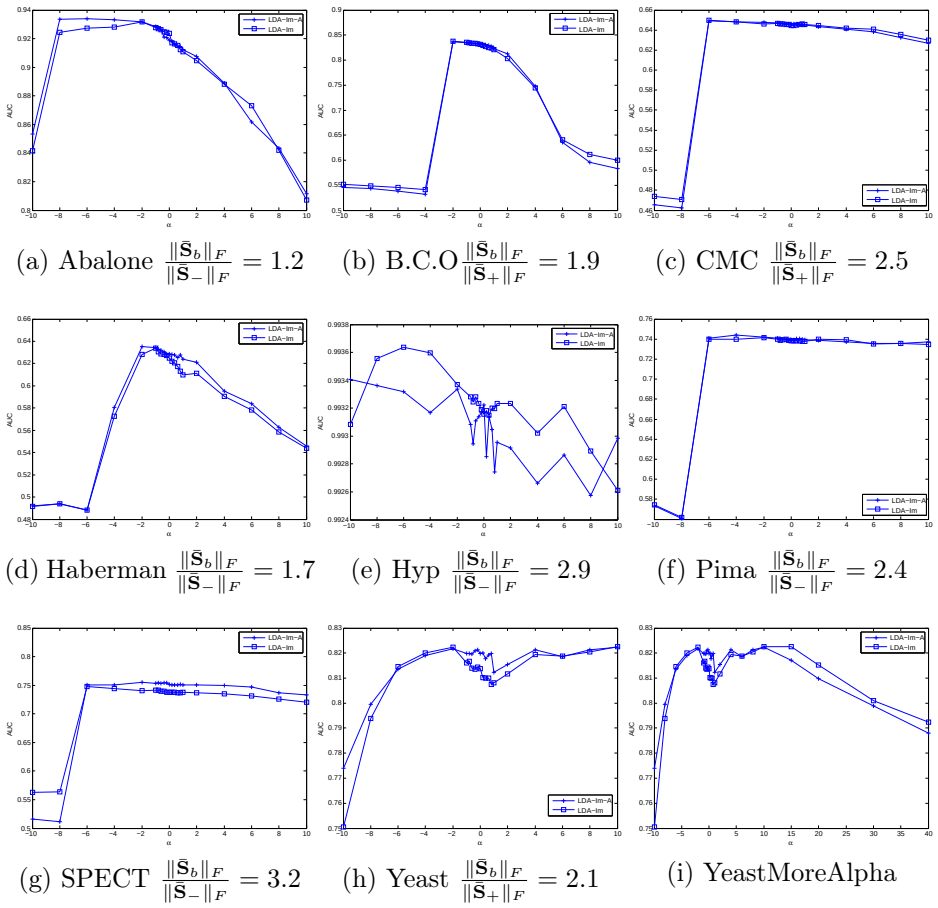
(f) Pima



(g) SPECT

(h) Yeast

Figure 2.3: Average AUC values obtained by minimizing positive and negative covariance matrices using first k column vectors of $\mathbf{W}$ in (2.25).



(a) Abalone $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_-\|_F} = 1.2$ (b) B.C.O $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_+\|_F} = 1.9$ (c) CMC $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_+\|_F} = 2.5$

(d) Haberman $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_-\|_F} = 1.7$ (e) Hyp $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_-\|_F} = 2.9$ (f) Pima $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_-\|_F} = 2.4$

(g) SPECT $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_-\|_F} = 3.2$ (h) Yeast $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_+\|_F} = 2.1$ (i) YeastMoreAlpha

Figure 2.4: System performance obtained with different values of α.

decreases when $\alpha > 15$. To find an estimation of α, we calculated the average $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_1\|_F}$ where $\bar{\mathbf{S}}_1$ is $\bar{\mathbf{S}}_+$ or $\bar{\mathbf{S}}_-$ for all the data sets. The ratios are shown in the titles of the sub-figures in Fig. 2.4. One can see that when $\alpha = -\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_1\|_F}$, the system could obtain a relative good AUC value, which means that $\frac{\|\bar{\mathbf{S}}_b\|_F}{\|\bar{\mathbf{S}}_1\|_F}$ could be a reference to estimate the value of α.

2.3.2.2 Multiclass classification

Four data sets of UCI are selected to evaluate the performance of our methods in multiclass classification problems. The details of the data sets are

Table 2.7: Data sets used for multiclass classification.

Data Set Name	Training Data Number	Testing Data Number	Class Number	Feature Dimension
DNA	2000	1186	3	180
Landsat	4435	2000	6	36
Optidigits	3823	1797	10	60
Theyroid	3772	3428	3	21

shown in Table 2.7. Six methods are compared in this experiment: APLS, LDA, EWLDR [80], LDA-Imbalance (LDA-I), LDA-A and LDA-Imbalance-A (LDA-I-A). Similar to our method, EWLDR proposed in [80] gave different weights to within-class scatter matrices of different classes to improve the performance of the basic LDA method. The weights are given according to the distance between classes, while we set the weight value according to the number of samples. An evolution strategy was also proposed in [80] to obtain best weights, which is useful for our future research. It is worth noting that we implement LDA method using one-against-rest strategy, and the results obtained are different from those in [80]. We also put the results in [80] here for the convenience of comparing. The APLS, LDA-I, LDA-A, and LDA-I-A also use the one-against-rest strategy. The results are shown in Table 2.8. It can be seen that LDA-I method obtains best results on Optidigits and Thyroid data sets, and LDA-Imbalance-A method performs best on DNA data set. For DNA and Optidigits data sets, the results are obtained with the parameter $\alpha = -2$ by minimizing the negative covariance matrix of training samples. The parameter α is set to be -0.5 for Landsat data set. For the Thyroid data sets, the results are obtained by minimizing the positive covariance matrix when $\alpha = -100$. We can see that the value of alpha is quite big, which indicates that the optimization object of LDA conflicts with CSP in this data set.

Table 2.8: Results of multiclass classification problems using different methods.

Data Sets	APLS	EWLDR	LDA [80]	LDA	LDA-I	LDA-A	LDA-I-A
DNA	0.9220	0.9418*	0.941	0.9310	0.9380	**0.9440**	**0.9440**
Landsat	0.7730	**0.8315**	0.8265	0.7955	0.8295*	0.7555	0.7810
Optdigits	0.9050	0.9410*	0.9388	0.9215	**0.9527**	0.0935	0.9238
Thyroid	0.7970	0.9422*	0.9387	0.9029	**0.9431**	0.8235	0.8442

2.3.2.3 Discussion

The method proposed in this chapter is a linear classifier. It could not solve some problems like XOR problem [82]. For example, if there are two minority clusters located at both sides of the majority class cluster in Fig. 2.1, the line will go through the center of the whole data set. Some methods like kernel methods could be applied to solve this problem.

Experiments of LDA-Imbalance method with SMOTE technology are also conducted. However, the results are much worse than the results without SMOTE technology. The sampling technology changed the distribution of the original training data set. In this case, the borderline obtained is suitable for the over sampled training set, while not suitable for the testing set. Hence, SMOTE technology is not able to improve the performance of the proposed method.

2.4 Application on emotion identification in text

In this section, we report the application of the proposed classifier on the text based emotion identification problem. In the research field of emotion recognition, six categories of emotion are often used to label the sentences: anger, disgust, fear, joy, sadness, and surprise [3, 2, 11]. Including neutral, there are

seven types of emotion that can be expressed in text. In our experiments, a sentence is classified into neutral or emotional classes. The corpus utilized in this work is the UIUC children's stories corpus [1], which consists of 176 stories by Grimm's, Andersen, and Potter. Every sentence in the corpus is annotated with one of the seven emotions by two annotators. The surprise class is subdivided into positive surprise and negative surprise in the corpus, and we treat these two classes as one surprise class in this chapter. There are disagreements between annotations, and here we take the first annotator's labels as the standard ones.

Features extracted from the sentences include special punctuations, number of positive and negative words, and emotional words. Some examples of the features are shown in Table 2.9. The special punctuation feature is extracted directly from the sentences. Each punctuation is represented by a boolean value. The positive and negative words are obtained online[5]. The TAGPos and TAGNeg word lists on the web are taken as the features, and there is a total of 1636 positive words and 2006 negative words. If we use a boolean value to describe every word in the list, we will obtain a high dimensional and sparse feature matrix with more than 3000 dimensions. To reduce the dimension of the feature matrix, we only take the number of positive words and the number of negative words in a sentence as the features. A list of 300 emotional words is collected using WordNet Affect [83, 84] and Internet. The number of emotional words is much less than the number of positive and negative words, and these emotional words often represent the emotion of a sentence directly. Hence, we set a boolean value for each emotional word in the feature vectors. The positive, negative and emotional words that do

[5]http://www.wjh.harvard.edu/~inquirer/homecat.htm

Table 2.9: The feature set used in the emotion identification experiments.

Feature	Example
Special punctuation	",!,?
Positive word counts	0,1,2,3,4,5,other
Negative word counts	0,1,2,3,4,5,other
WordNet emotion words	anger, fear, happiness

not appear in the UIUC corpus are removed from the list before we generate features for the sentences. A 149 dimension feature vector is generated for each sentence at last.

To compare with the existing methods, a 5-fold cross validation experiment is conducted. In our experiment, we only identify a sentence is emotional or not. The number of neutral and emotional sentences in the corpus is shown in Table 2.10. One can see that in both training and testing sets, the numbers of neutral sentences are about twice as big as the numbers of emotional ones. The methods compared in this section are same as those in Section 2.3.2.1. Similar to Table 2.5, LDA and LDA-I in Table 2.11 denote the results obtained by Symmetric Method, and LDA-A and LDA-I-A denote the results obtained by the Asymmetric Method. We list the results of different criteria in Table 2.11 for every method. In the table, the highest accuracy is highlighted in bold font. One can see that SVM method obtained highest accuracy for neutral sentences but the lowest accuracy for emotional sentences. The SVM with SMOTE technology could generate balanced accuracy for both classes. The methods except SVM could generate competitive UA values, and the highest UA is obtained by APLS and LDA-I method. The LDA-I-A method obtained highest accuracy for emotional sentences, and best F1 and AUC values. One can also find that LDA based methods could obtain competitive results with much less time consuming comparing

Table 2.10: Number of neutral and emotional sentences in UIUC Children's Story corpus.

	Sentence Number	Neutral/Emotional
Training Set	12241	8048/4193
Testing Set	3061	2091/970

Table 2.11: System performance of detecting emotional sentences.

	SVMs	SMOTE	APLS	LDA	LDA-I	LDA-A	LDA-I-A
Emotional	0.352	0.713	0.708	0.667	0.689	0.687	**0.715**
Neutral	**0.895**	0.599	0.611	0.633	0.630	0.610	0.604
WA	**0.712**	0.637	0.644	0.644	0.650	0.636	0.641
UA	0.624	0.656	**0.660**	0.650	**0.660**	0.649	0.659
Precise	**0.632**	0.475	0.482	0.481	0.490	0.473	0.480
Recall	0.352	0.713	0.708	0.667	0.689	0.687	**0.715**
F1	0.452	0.570	**0.573**	0.555	0.570	0.557	**0.573**
GMean	0.562	0.653	**0.657**	0.645	0.656	0.643	0.656
AUC	0.609	0.643	**0.647**	0.638	**0.647**	0.636	**0.647**
Time(S)	90.8	237.0	143.9	3.8	4.1	**3.7**	**3.7**

to other methods.

2.5 Summary

In this chapter, a LDA based classification method has been proposed to address the imbalanced pattern classification problem. We have demonstrated the efficiency of the method in both two classes and multiclass classification problems on imbalanced data sets through theoretical justifications and empirical studies. The method has been applied to identify emotional sentences in an article. The experimental results showed that the LDA based methods could obtain competitive results comparing to existing methods while consumes much less time.

Chapter 3

An Asymmetric Simple Partial Least Squares (SIMPLS) based Classifier

3.1 Introduction

In this chapter, a classifier based on Asymmetric Simple Partial Least Squares (SIMPLS) is introduced to solve the emotional sentence identification problem which has been formulated as an imbalanced pattern classification problem in last chapter. The researchers in I^2R, A*Star, Singapore found that the SimPLS based classifier works well for imbalanced pattern classification problems like voice conversion and speaker state classification [38]. The speaker state classification problem aims to predict a speaker's state is intoxication/sleepiness or not [85]. In these applications, SIMPLS gave extremely better prediction accuracy to the class with the smaller data number. An asymmetric SIMPLS (ASIMPLS) classifier was introduced to enhance the performance of SIMPLS to the class with the larger data number in [38]. Hence, the method is applied to emotional sentence identification problem in this chapter. The method is expected to generate good accuracy for emo-

tional sentences (True Positive Rate) which is difficult for some traditional methods.

Partial Least Squares (PLS) is a well known dimension reduction method that has been recently adapted for high dimensional classification problems. An asymmetric PLS classifier was proposed to solve the problem of unbalanced pattern classification in the biomedical field [36]. In [36], an algorithm of adjusting the boundary in the score vector space of the training data was proposed to balance the True Positive Rate (TPR) and True Negative Rate (TNR). SIMPLS is an alternative approach to PLS and it is widely used in practice due to its fast and avoidance of matrix inverse calculation [86]. A SIMPLS based classifier has been proposed to solve the speaker state classification problem in [38]. An algorithm was proposed to adapt the model obtained by SIMPLS to different distribution of development data sets, which is assumed to reflect the distribution of real-world data set. The asymmetric SIMPLS classifier has been evaluated with the SLC corpus for INTERSPEECH 2011 Speaker State Challenge [85, 38] which aims to detect the sleepiness of a speaker. Experimental results show that the asymmetric SIMPLS classifier features the properties of fast, a high accuracy of the minority class, and less susceptible to the distribution of different classes.

In this chapter, we investigate the effect of SIMPLS in the text based emotion identification tasks. This chapter is organized as follows: Section 3.2 presents theoretical study on the effect of asymmetric SIMPLS for unbalanced data classification. In Section 3.3, experiments on the UIUC children story corpus are reported, including feature sets, performance comparison of different classifiers, and the performance analysis. Conclusions and future work are discussed in Section 3.4.

3.2 Asymmetric SIMPLS Classifier

Given a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ and a label matrix $\mathbf{y} \in \mathbb{R}^{N \times 1}$, SIMPLS aims to find a predictive linear model [86]

$$\widehat{\mathbf{Y}} = \mathbf{XB} \tag{3.1}$$

where N is the number of samples and M is the dimension of features. Here $\mathbf{y}$ is one dimensional in binary classification problems. The solution is obtained by extracting the successive orthogonal factors of $\mathbf{X}$ and $\mathbf{Y}$,

$$\mathbf{t}_a = \mathbf{X}_0 \mathbf{r}_a \tag{3.2}$$

and

$$\mathbf{u}_a = \mathbf{Y}_0 \mathbf{q}_a, a = 1, 2, ..., A \tag{3.3}$$

with the following four restrictions, where $\mathbf{X}_0 = \mathbf{X} - \text{mean}(\mathbf{X})$, $\mathbf{Y}_0 = \mathbf{Y} - \text{mean}(\mathbf{Y})$, and $A \leq M$.

(i) The covariance of $\mathbf{u}_a$ and $\mathbf{t}_a$ is maximized: $\max(\mathbf{u}_a' \mathbf{t}_a) = \max(\mathbf{q}_a' (\mathbf{Y}_0' \mathbf{X}_0) \mathbf{r}_a)$;

(ii) $\mathbf{r}_a$ is normalized: $\mathbf{r}_a' \mathbf{r}_a = 1$;

(iii) $\mathbf{q}_a$ is normalized: $\mathbf{q}_a' \mathbf{q}_a = 1$; and

(iv) The scores are orthogonal to each other: $\mathbf{t}_b' \mathbf{t}_a = 0$ for $a > b$.

The algorithm of extracting scores and loadings of $\mathbf{X}$ and $\mathbf{Y}$ is shown in Algorithm 1 [86, 38].

In the algorithm, the score vector $\mathbf{t}_a$ is normalized by $\mathbf{t}_a = \mathbf{t}_a / \sqrt{\mathbf{t}_a' * \mathbf{t}_a}$. According to the restriction (iv), we have $\mathbf{T}'\mathbf{T} = \mathbf{I}$ where $\mathbf{T} = [\mathbf{t}_1, ..., \mathbf{t}_A]$. To

Algorithm 1 Asymmetric SIMPLS Training

Input: Feature set $\mathbf{X}$, Label $\mathbf{y}$, and Number of components A
Variables: Projection matrix $\mathbf{R}$,
score vectors $\mathbf{T}$ and $\mathbf{U}$, loading $\mathbf{P}$ and $\mathbf{Q}$
$\mathbf{R} = []; \mathbf{V} = []; \mathbf{Q} = []; \mathbf{T} = []; \mathbf{U} = [];$
$\mathbf{y} = [y_1, y_2, ..., y_N]'; \mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]'$
$\mathbf{y}_0 = \mathbf{y} - \text{mean}(\mathbf{y})$; $\mathbf{X}_0 = \mathbf{X} - \text{mean}(\mathbf{X})$;
$\mathbf{S} = \mathbf{X}_0'\mathbf{y}_0$
for $i = 1$ to A **do**
  $\mathbf{q}_i$ = dominant eigenvectors of $\mathbf{S}'\mathbf{S}$
  $\mathbf{r}_i = \mathbf{S} * \mathbf{q}_i$
  $\mathbf{t}_i = \mathbf{X}_0 * \mathbf{r}_i$
  $\text{normt}_i = \text{SQRT}(\mathbf{t}_i'\mathbf{t}_i)$
  $\mathbf{t}_i = \mathbf{t}_i/\text{normt}_i$
  $\mathbf{r}_i = \mathbf{r}_i/\text{normt}_i$
  $\mathbf{p}_i = \mathbf{X}_0' * \mathbf{t}_i$
  $\mathbf{q}_i = \mathbf{y}_0^T * \mathbf{t}_i$
  $\mathbf{u}_i = \mathbf{y}_0 * \mathbf{q}_i$
  $\mathbf{v}_i = \mathbf{p}_i$
  if $i > 1$ **then**
    $\mathbf{v}_i = \mathbf{v}_i - \mathbf{V} * (\mathbf{V}' * \mathbf{p}_i)$
    $\mathbf{u}_i = \mathbf{u}_i - \mathbf{T} * (\mathbf{T}' * \mathbf{u}_i)$
  end if
  $\mathbf{v}_i = \mathbf{v}_i/\mathbf{SQRT}(\mathbf{v}_i' * \mathbf{v}_i)$
  $\mathbf{S} = \mathbf{S} - \mathbf{v}_i * (\mathbf{v}_i' * \mathbf{S})$
  $\mathbf{r}_i, \mathbf{t}_i, \mathbf{p}_i, \mathbf{q}_i, \mathbf{u}_i$, and $\mathbf{v}_i$ into
  $\mathbf{R}, \mathbf{T}, \mathbf{P}, \mathbf{Q}, \mathbf{U}$, and $\mathbf{V}$, respectively.
end for
$\mathbf{B} = \mathbf{R} * \mathbf{Q}'$

predict the label of the samples,

$$\widehat{\mathbf{Y}}_0 = \mathbf{T}\mathbf{T}'\mathbf{Y}_0 = \mathbf{X}_0\mathbf{R}\mathbf{R}'\mathbf{X}_0'\mathbf{Y}_0 = \mathbf{X}_0\mathbf{R}\mathbf{R}'\mathbf{S}_0 \tag{3.4}$$

Hence, $\mathbf{B}$ in (3.1) can be written as:

$$\mathbf{B} = \mathbf{R}(\mathbf{R}'\mathbf{S}_0) = \mathbf{R}(\mathbf{T}'\mathbf{Y}_0) = \mathbf{R}\mathbf{Q}' \tag{3.5}$$

The labels of new samples can be predicted by

$$\widehat{\mathbf{Y}}^* = \mathbf{X}_0^*\mathbf{B} \tag{3.6}$$

where $\mathbf{X}_0^* = \mathbf{X}^* - \text{mean}(\mathbf{X}^*)$ and $\mathbf{X}^*$ is the new feature matrix.

Inspired by [36], we would like to look deeper into the label prediction. PLS will find the score and loading vectors in (3.2) and (3.3). The inner relation between $\mathbf{X}$ and $\mathbf{Y}$ can be estimated by the regression coefficient $\mathbf{b}$ via the latent variables [86]:

$$\widehat{\mathbf{u}}_a = \mathbf{b}_a\mathbf{t}_a \tag{3.7}$$

$$\mathbf{b}_a = \mathbf{u}_a'\mathbf{t}_a/(\mathbf{t}_a'\mathbf{t}_a) \tag{3.8}$$

Hence, the algorithm of predicting labels for new samples is summarized in Algorithm 2. From Algorithm 2, we have

$$\hat{\mathbf{Y}} = \text{sign}(\sum_{i=1}^{A}\mathbf{b}_i\mathbf{t}_i\mathbf{q}_i') = \text{sign}(\sum_{i=1}^{A}\mathbf{m}_i\mathbf{t}_i) = \text{sign}(\mathbf{m}\cdot\mathbf{t}) \tag{3.9}$$

One can see that the label $\widehat{\mathbf{Y}}$ is a function of the score vectors $\mathbf{t}$. The algorithm generated a borderline which is a hyper plane in the score space

Algorithm 2 Predicting labels for new samples using PLS.

Input: New feature matrix $\mathbf{X}$; projection matrix $\mathbf{R}$, regression coefficients $\mathbf{b}$, loading $\mathbf{P}$ and $\mathbf{Q}$ obtained by Algorithm 1 on the training set.
Output: Predicted Label $\hat{Y}$ of $\mathbf{X}$

$\mathbf{X}_0 = \mathbf{X} - \text{mean}(\mathbf{X})$
for $i = 1$ to A **do**
  $\mathbf{t}_i = \mathbf{X}_{i-1}\mathbf{r}_i$;
  $\mathbf{X}_i = \mathbf{X}_{i-1} - \mathbf{t}_i\mathbf{p}_i'$;
end for
$\hat{\mathbf{Y}} = \text{sign}(\sum_{i=1}^{A} b_i\mathbf{t}_i\mathbf{q}_i')$

of feature matrix $\mathbf{X}$, which goes through the original point. An example of the borderline obtained on a synthetic data set is shown in Fig. 3.1. In the figure, the X-axis and Y-axis denote the first and second dimension of the score vector $\mathbf{t}$. The borderline draw in dash-dot line goes through the original point $(0, 0)$, which is away from the center of the minority class. If we classify the points using this borderline, many samples belonging to the majority class will be miss-classified. A method was proposed in [36] to adjust the borderline to enhance the accuracy of majority samples. The idea is to move the borderline towards to the center of minority class by estimating the radii of the two classes.

$$bias = \text{mean}(\mathbf{t}_p) - (\text{mean}(\mathbf{t}_p) - \text{mean}(\mathbf{t}_n)) * \text{std}(\mathbf{t}_p)/(\text{std}(\mathbf{t}_p) + \text{std}(\mathbf{t}_n)) \quad (3.10)$$

where std is the standard deviation function. The score vector of minority class is $\mathbf{t}_p$, and the score vector of majority class is $\mathbf{t}_n$. Equation (3.9) can be written as

$$\hat{\mathbf{Y}} = \text{sign}(\mathbf{m} \cdot \mathbf{t}) - \mathbf{m}_1 bias \quad (3.11)$$

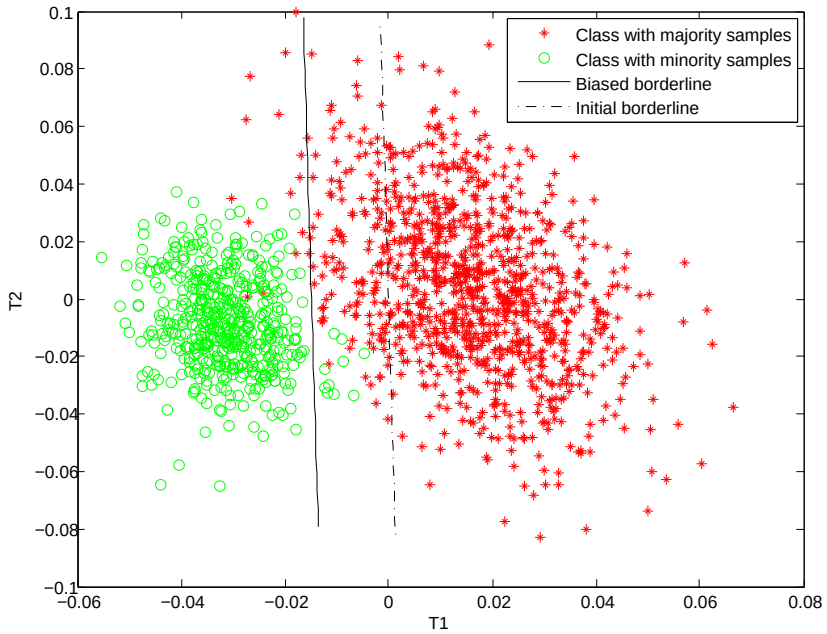


Figure 3.1: Asymmetric SIMPLS classifier is illustrated on a synthetic dataset.

The adjusted borderline is also shown in Fig. 3.1 drawn by the real line. One can see that the line is more close to the center of minority class.

3.3 Experimental Results

The method is evaluated using the UIUC children's stories corpus [1], which consists of 176 stories by Grimm's, Andersen, and Potter. The features are same as those in Chapter 2, which include special punctuations, number of positive and negative words, and emotional words. A 149 dimension feature vector is generated for each sentence. SVM, Extreme Learning Machine (ELM) [87], and APLS [36] methods are employed to compare with our algorithm. Criterions including True Positive Rate (TPR), True Negative Rate (TNR), Weighted Accuracy (WA), Unweighted Accuracy (UA),

Table 3.1: System performance of different methods.

	SVM	SVM-S	ELM	ELM-S	APLS	ASimPLS
Emotional(%)	0.352	0.713	0.204	0.581	0.708	**0.736**
Neutral(%)	0.895	0.599	**0.911**	0.605	0.611	0.582
WA(%)	**0.712**	0.637	0.672	0.596	0.644	0.634
UA(%)	0.624	0.656	0.558	0.593	**0.660**	0.659
Precise	**0.632**	0.475	0.561	0.459	0.482	0.473
Recall	0.352	0.713	0.204	0.581	0.708	**0.736**
F1	0.452	0.570	0.279	0.486	0.573	**0.575**
GMean	0.562	0.653	0.410	0.547	**0.657**	0.654
AUC	0.609	0.643	0.541	0.580	**0.647**	0.646
Time(S)	90.8	237.0	**0.3**	55.5	143.9	4.4

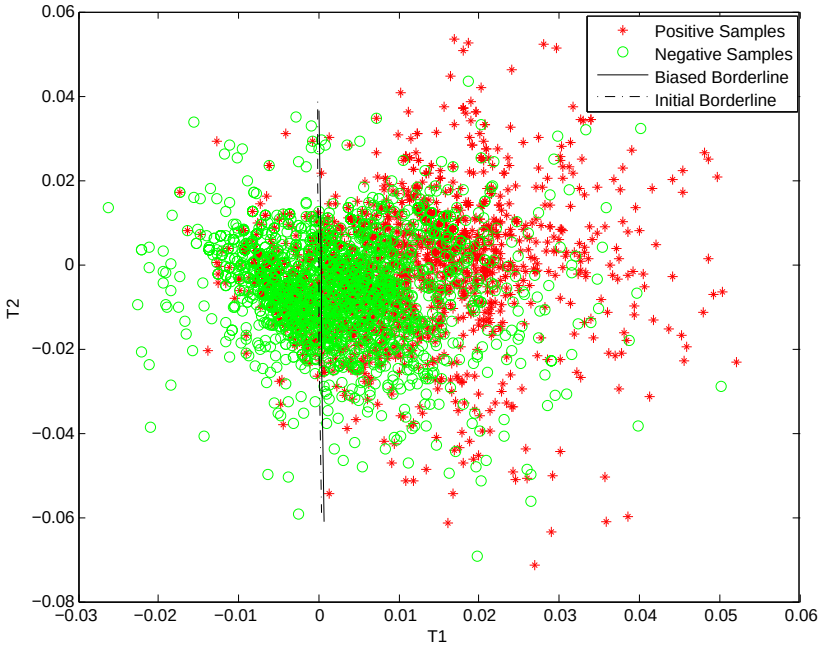
and F-Measure are employed to evaluate the methods. A five-fold cross validation is conducted, and the results are shown in Table 3.1. One can find that SVM and ELM obtained high accuracies for neutral sentences and low accuracies for emotional sentences, which is the effect of the imbalanced data set as we discussed in previous chapters. SVM-S and ELM-S denote the two methods with SMOTE [23] technologies which over-samples the minority samples in the training set before the data is sent to the classifiers. The two methods achieved a better balance between TPR and TNR. APLS [36] method obtained a balanced result without using any preprocessing technologies. ASimPLS method obtained a competitive result with APLS method using less time. ELM method is the fastest one among all the algorithms, and it will be slower than ASimPLS if SMOTE technology is applied to generate new samples.

To show the borderline of the classifier clearly, we illustrate the two dimension score vector space of the data set in Fig. 3.2. In the figure, the vectors of first and second dimension of the score matrix are X-axis and Y-axis respectively. The dash dot line denotes the borderline without using the

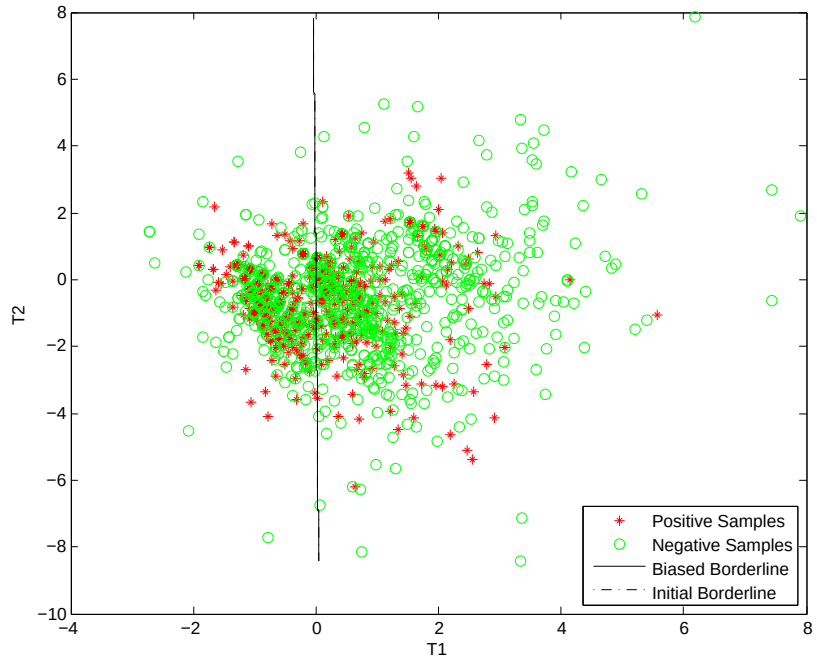
bias. The real line denotes the borderline after adjusting. One can see that the centers of positive and negative samples are very close. Hence, the bias calculated by (3.10) is quite small. The biased borderline is closer to the center of positive samples, which will increase the TNR.

3.3.1 Highly agree corpus

In the UIUC corpus, the two annotators may annotate different labels for a same sentence. We select the sentences labeled with same emotions by the two annotators to evaluate our algorithm. Ideally, the results should be better than those obtained using the sentences annotated by the first annotator because more people agree with the new labels. The results obtained by different methods on the highly agree corpus in a five-fold cross validation is shown in Table 3.2. As discussed in last chapter, the highly agree corpus is more imbalanced than the whole corpus. However, one can find that SVM method outperforms all the other methods, which is not the expected result. Both APLS and ASimPLS method obtained a very high TPR and a very low TNR. We plot the first two dimensions of the score vector space of the feature set in Fig. 3.3. In the figure, we can find that the distribution of testing samples is not same as the training samples, which may be the reason for the failing of ASimPLS. To prove this, a 2-fold cross validation experiment was conducted, which increased the number of samples in the testing data set. The results are shown in Table 3.3. We can see that ASimPLS method could generate competitive results with SVM. The score vector space of the data set is shown in Fig. 3.4. Compared to Fig. 3.3, the distribution of the testing set is more similar to the training set. The SVM method obtained



(a) Training set



(b) Testing set

Figure 3.2: The score vector space of the feature set.

Table 3.2: Comparison of system performance of different methods on the highly agree data set.

	SVM	SVM-S	ELM	ELM-S	APLS	ASimPLS
Emotional	0.408	**0.972**	0.318	0.762	0.853	0.960
Neutral	**0.917**	0.033	0.765	0.282	0.249	0.087
WA	**0.772**	0.301	0.635	0.418	0.421	0.336
UA	**0.662**	0.502	0.541	0.522	0.551	0.524
Precise	**0.662**	0.286	0.479	0.333	0.365	0.297
Recall	0.408	**0.972**	0.318	0.762	0.853	0.960
F1	**0.505**	0.442	0.294	0.427	0.468	0.453
GMean	**0.611**	0.176	0.390	0.336	0.244	0.223
AUC	**0.643**	0.472	0.517	0.496	0.525	0.495
Time(S)	41.4	140.9	**0.2**	12.7	18.0	1.9

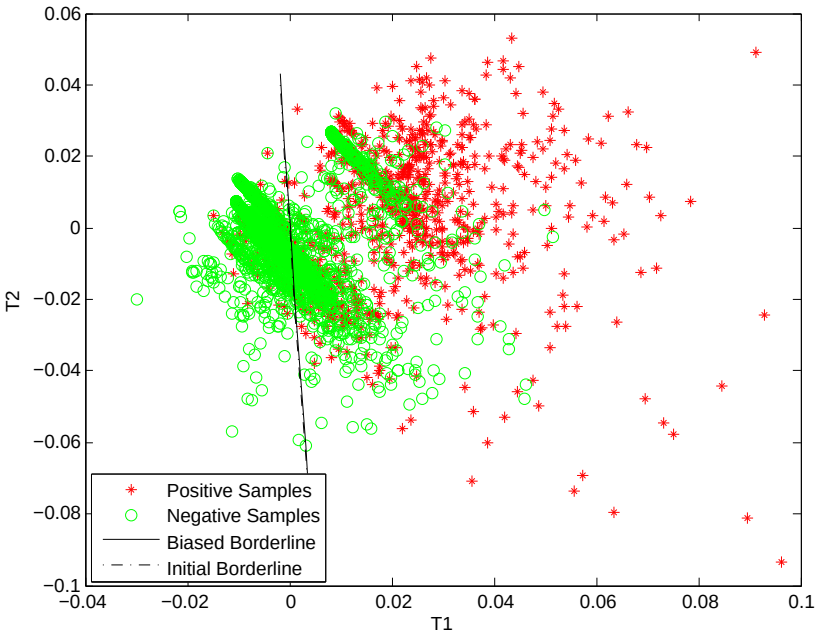
Table 3.3: Results obtained in the 2-fold cross validation experiment.

	SVM	SVM-S	ELM	ELM-S	APLS	ASimPLS
Emotional	0.408	0.457	0.344	0.808	**0.992**	0.587
Neutral	**0.916**	0.879	0.841	0.249	0.007	0.736
WA	**0.771**	0.759	0.698	0.409	0.288	0.694
UA	0.662	**0.668**	0.592	0.529	0.499	0.662
Precise	**0.659**	0.602	0.509	0.333	0.285	0.471
Recall	0.408	0.457	0.344	0.808	**0.992**	0.587
F1	0.504	0.520	0.378	0.445	0.443	**0.523**
GMean	0.611	0.634	0.514	0.334	0.083	**0.657**
AUC	0.643	**0.650**	0.571	0.503	0.470	0.644
Time(S)	9.9	33.3	**0.2**	4.7	6.0	3.1

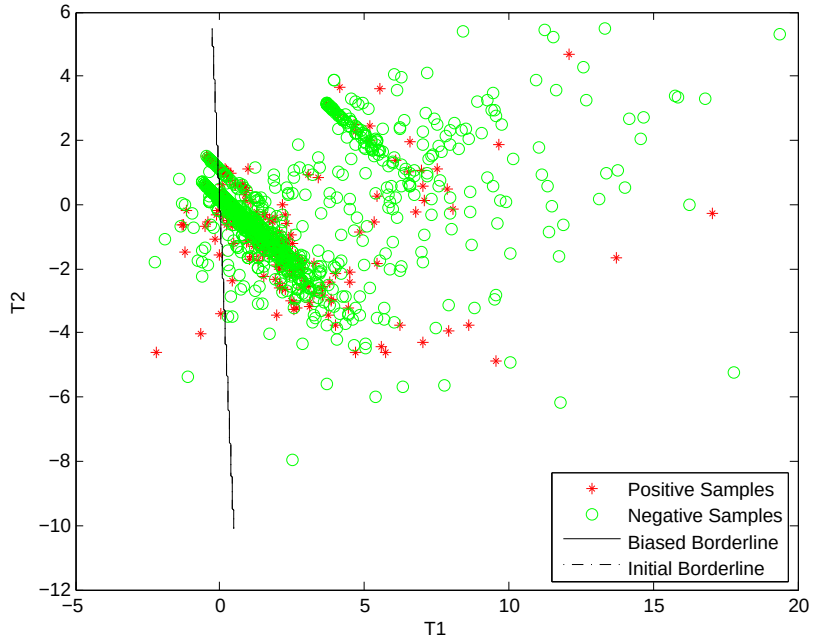
almost same results as in the 5-fold experiments, which demonstrates that it is very stable.

3.3.2 Number of components

When we build the PLS model, a very important issue is the choice of the optimal number of components k. In this experiment, we set the value of k from 1 to 100, and run the system to evaluate the influence of k. The F1 values obtained on the whole corpus and the highly agree corpus are shown

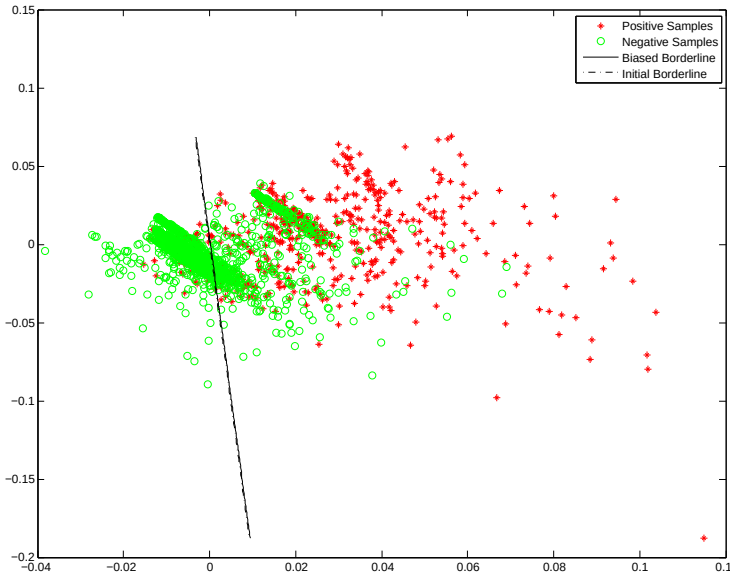


(a) Training

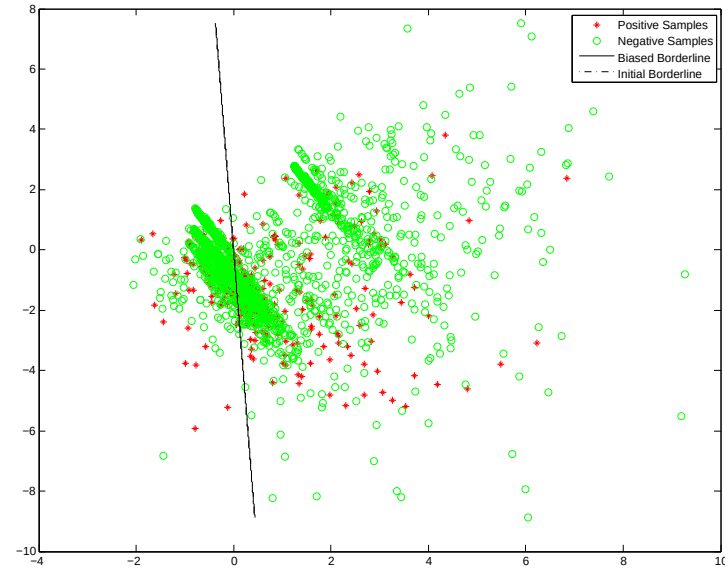


(b) Testing

Figure 3.3: The score vector space of the feature set of highly agree corpus.

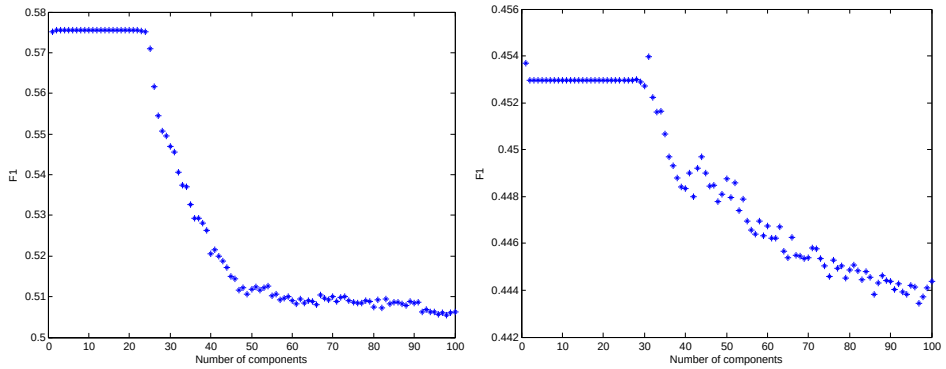


(a) Training



(b) Testing

Figure 3.4: The score vector space of the feature set of highly agree corpus in the 2-fold cross validation experiments.



(a) F1 values on the whole data set. (b) F2 values on the highly agree data set.

Figure 3.5: The system performance with different number of components.

in Fig. 3.5. One can see from the figure that the F1 value achieved highest when the number of components is 2, and became stable until k is about 30 for the whole data set. When $k > 30$, the system performance decreased heavily, which means that the information contained in the high dimensional components became noise. Similar trend can be found on the highly agree corpus.

3.4 Summary

In this chapter, we have employed a SIMPLS based method to identify emotional sentences in an article. Experiments on a children story corpus have been conducted, and the results demonstrated that the method could obtain competitive results with existing methods using less time. The method has been proved to be able to generate good TPR as expected.

Chapter 4

Classifier Fusion for Emotion Identification

4.1 Introduction

In this chapter, we investigate the methods of fusing different classifiers to identify emotional sentences in text. The assumption is that combining classifiers with different properties will lead to a better system. A new classifier called Extreme Learning Machine (ELM) [87] is employed to solve the emotion identification problem. ELM has been proved to be efficient in many classification problems like EEG signal processing [88], cancer diagnosis [89] and protein sequence classification [90] etc. We employ ELM to predict a sentence is emotional or not in this work. Other classifiers like LDA-Imbalance and ASimPLS are also used in this chapter. The features extracted form sentences include UniGram, Subjective words, and special punctuations etc. Besides these traditional features, we calculate the emotion value of each words based on some seed emotional words. The emotion values of these words are used to calculate the emotion value of a sentence, and the values of sentences are taken as features for classification. We divide the feature sets into three subsets, and we train a model on each subset using ELM and

ASimPLS. Then the six models are combined using fusion methods. Two fusion methods are employed in this chapter: a weighted summation method and the FoCal method. The contributions of this chapter are highlighted as follows:

(i) a fusion system is proposed to identify emotional sentences in an article;

(ii) the ELM method is employed to solve the text based emotion identification problem; and

(iii) a feature that represents the emotion values of a word is presented, and the feature is used to calculate the emotion of a sentence.

The rest of the chapter is organized as follows: The proposed fusion system for predicting emotional sentences is presented in Section 4.2. Our method is evaluated on a children story corpus and the results are reported in Section 4.3. We conclude our work in Section 4.4.

4.2 A fusion system for emotional sentence identification

Figure 4.1 illustrates the structure of the proposed fusion system which is composed of six sub-systems with three different feature sets and two classifiers. Synthetic Minority Over-sampling Technique (SMOTE) [23] technology is applied before the features are sent to ELM because the data set is highly imbalanced. ELM is the classifier that will be evaluated in this work. ASimPLS [38] is selected because the results in last chapter shew that it could generate good results for emotion identification in text. The fusion of the

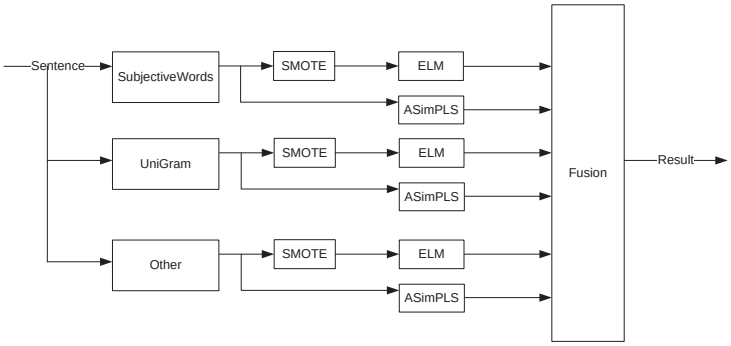


Figure 4.1: Structure of the proposed fusion system.

six classifiers are carried out using two different methods: the FoCal fusion method and a weighted summation method.

4.2.1 Features

Features extracted from the sentences include special punctuations, number of positive and negative words, emotional words, and UniGram. Some examples of the features are shown in Table 4.1. The special punctuation feature is extracted directly from the sentences. Each punctuation is represented by a boolean value. The positive and negative words are obtained online[1]. We only take the number of positive words and the number of negative words in a sentence as the features. Reported by [3], a set of subjective words could generate good results in emotion identification. We take the same set of subjective words[2] as our features. There are more than $8,000$ words in the list, and we select those appear in the corpus more than 10 times as features. Only 373 subjective words left after selection, and we set a boolean value for each word in the feature vectors. UniGram is also taken as a feature of

[1]http://www.wjh.harvard.edu/~inquirer/homecat.htm
[2]http://www.cs.pitt.edu/mpqa/subj_lexicon.html

Table 4.1: Some examples of the features used in the classification system.

Feature	Example
Special punctuation	",!,?
Positive word counts	0,1,2,3,4,5,other
Negative word counts	0,1,2,3,4,5,other
WordNet emotion words	anger, fear, happiness
Emotion of words	terrible [5,1,3,29,0,2]
Subjective words	anxious, clever

a sentence. Only the words appear more than 25 times in the corpus are selected.

Furthermore, about 300 emotional words are collected using WordNet Affect [83, 84] and Internet. These emotional words often represent the emotion of a sentence directly. We also set a boolean value for each word in the feature. We believe that words that are not in the list also can represent emotions, and they are important features for identifying an emotional sentence. Taking the 300 emotional words as seed words, we calculate the emotion of all the words existing in the corpus using WordNet. The emotion of a word is represented by a vector $\overrightarrow{\mathbf{r}} = [r_1, ..., r_6]$ where each element of the vector is an integer denoting the magnitude of an emotion. The meaning of each element is [*Surprise Anger Disgust Fear Joy Sad*]. The emotion of a word is calculated in two directions. The first direction is that if a given word appears in the synset of a *happy* word, the value of *happy* emotion of the given word will plus one. Similarly, if a *happy* word exists in the synset of a given word, the value of *happy* emotion of this word also pluses one. The emotion of a sentence is calculated by summing the emotion of its words.

4.2.2 Classifiers

Two classifiers are employed in the fusion system. The first one is ASimPLS [38] which is a classifier that has been used for speaker state classification tasks. ELM [87, 88] is a learning method derived from Single Layer Feedforward Network (SLFN). It has been proved that to train a SLFN, one may fix the weights between input neurons and hidden neurons, and only adjust the weights between hidden neurons and output neurons [91]. Inspired by this, ELM randomly chooses and fixes the weights between input neurons and hidden neurons, and then determines the weights between hidden neurons and output neurons of the SLFN.

Given a training set $\{(\mathbf{x}_k, \mathbf{t}_k)|\mathbf{x}_k \in \mathbb{R}^{n\times 1}, \mathbf{t}_k \in \mathbb{R}^{m\times 1}, k = 1, \ldots, N\}$, a SLFN with $\widetilde{N}$ hidden neurons and activation function $g(x)$ is modeled as

$$\sum_{i=1}^{\widetilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_k + b_i) = \mathbf{o}_k \tag{4.1}$$

where $w_i \in \mathbb{R}^{n\times 1}$ is the vector of weights between the input neurons and the ith hidden neuron, and $\beta_i \in \mathbb{R}^{m\times 1}$ is the weight vector of the ith hidden neuron and the output neurons. The vector $\mathbf{o}_k$ is the output, and b_i is the threshold of the ith hidden neuron. The equations for the N samples can be written in matrix format:

$$\mathbf{H}\beta = \mathbf{O} \tag{4.2}$$

where

$$H = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\widetilde{N}} \cdot \mathbf{x}_1 + b_{\widetilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_{\widetilde{N}} \cdot \mathbf{x}_N + b_{\widetilde{N}}) \end{bmatrix} \tag{4.3}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\widetilde{N}}^T \end{bmatrix} \tag{4.4}$$

and

$$\mathbf{O} = \begin{bmatrix} \mathbf{o}_1^T \\ \vdots \\ \mathbf{o}_N^T \end{bmatrix} \tag{4.5}$$

ELM randomly assigns the input weights $\mathbf{w}_i$ and the threshold b_i according to some continues probability density function, and calculate the output weights β_i by solving

$$\widehat{\beta} = \arg\min_{\beta} \parallel \mathbf{H}\beta - \mathbf{T} \parallel^2 \tag{4.6}$$

where

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} \tag{4.7}$$

The solution is given by [88]

$$\widehat{\beta} = \mathbf{H}^{+}\mathbf{T} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T} \tag{4.8}$$

4.2.3 Fusion of Classifiers

Two fusion methods are employed: FoCal fusion and weighted summation fusion. Three data sets: training, development, and testing data sets are needed for fusion of classifiers. A set of scores are obtained for samples in the development data set, and we train a set of weights for the classifiers to

make the weighted summation of the scores obtained on the development set achieve the best accuracy. For a new sample, a set of scores are obtained by the single classifiers. The final score of this sample is the weighted summation of all the scores. The sample is then classified into a class according to the final score.

4.2.3.1 FoCal fusion

We conduct linear logistic regression fusion using FoCal toolkit [92] which is a MATLAB toolkit for evaluation, fusion and calibration of statistical pattern recognizers. To train the fusion parameters, a set of supervised training scores are obtained first, and an objective function is defined in terms of these scores. The fusion weights are obtained by optimizing this objective. As shown in (4.9), the predicted score of a new sample x is a weighted summation of the scores obtained by different classifiers.

$$\hat{y}(x) = \sum_{i=1}^{k} \alpha_i y_i(x) + \boldsymbol{\beta} \tag{4.9}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_k]$ is the vector contains the weight of each classifier, and k is the number of classifiers. The value $y_i(x)$ is the log-likelihood of a sample x belonging to positive or negative class obtained by classifier i. The vector $\boldsymbol{\beta}$ is a bias to adjust the summation. For two classes classification problems, $\boldsymbol{\beta}$ has two dimensions. We solve the following optimal problems to obtain $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

$$\arg\min(-\sum_{x=1}^{N} w_x \log_2 P_x) \tag{4.10}$$

where $w_x = \frac{1}{N_{pos}}$ if x belongs to the positive class, and N_{pos} is the number of positive samples in the data set. Similarly, $w_x = \frac{1}{N_{neg}}$ if x belongs to the negative class. N is the number of all training samples. The probability

$$P_x = \frac{\exp(\hat{y}_{c(x)}(x))}{\exp(\hat{y}_{pos}(x)) + \exp(\hat{y}_{neg}(x))} \tag{4.11}$$

where $c(x) = pos$ or $c(x) = neg$ is the class that x belongs to, and $\hat{y}_{c(x)}(x)$ is obtained by (4.9). Many methods can be used to find the solution of this optimal problem. We used a conjugate gradient algorithm[3] to find the solution.

4.2.3.2 Weighted summation

A weighted summation fusion method is also used to generate combination results:

$$\hat{y}(x) = \sum_{i=1}^{k} w_i y_i(x) \tag{4.12}$$

where x is an instance of the development set and $y_i(x)$, $i = 1, \ldots, k$ are scores generated by the subsystems, and k is the number of subsystems. The weight $w_i \in \{0, 0.1, 0.2, \ldots, 1\}$. We tried all the combination of w_i for all classifiers to obtain the best F1 value on the development data set. The group of weights which obtains the best performance is selected to fuse the results obtained on the testing set.

[3]The details are available at http://www.dsp.sun.ac.za/ nbrummer/focal/index.htm

Table 4.2: The number of neutral and emotional sentences in the corpus.

	Neutral	Anger	Disgust	Fear	Joy	Sadness	Surprise
Sentence Number	6016	261	199	354	825	422	342

4.3 Experimental results

4.3.1 Data set

We utilize the UIUC children's stories corpus [11], which consists of 176 stories from Grimm, Andersen, and Potter. In the corpus, every sentence is annotated with one of the eight labels: neutral, angry, disgust, fearful, happy, sad, positive surprised, and negative surprised. In this chapter, we take the positive and negative surprised as one class. Two annotators finished all the labeling, and there are disagreement between the annotators. In our experiments, we only select the sentences that are annotated with same labels by the two annotators. The numbers of different emotional sentences are shown in Table 4.2. It can be seen from the table that the data sets are highly imbalanced where the number of neutral sentences is much bigger than the summation of all the other sentences. For ELM method, the SMOTE [23] technology was applied before we trained the models to solve this imbalanced problem.

4.3.2 Performance of ELM

A five-fold cross validation is conducted on different feature sets. The evaluation criteria include unweighted accuracy, weighted accuracy, precise, recall, and F1. The relative definitions are given in Equations (2.38) and (2.39).

Table 4.3 shows the experimental results of cross validation of ELM, ELM

with SMOTE, LDA-Imbalance proposed in Chapter 2 and ASimPLS in Chapter 3. The time consuming reported in the table includes the time of training models and generating results for testing samples. One can see that ELM takes less than one second to finish running with all the features, which is the fastest among the algorithms. However, the system performance is affected by the imbalanced data set. ELM with SMOTE (ELM-S) technology could improve the performance. For ELM-S method, the best performance is achieved using the Other feature set, which means that simply putting all features together may even decrease the system performance. Classifier fusion may be a method to utilize all the features.

Table 4.3: Comparison of experimental results obtained by different classifiers on different feature sets.

Method	TPR	TNR	% UA	% WA	Precise	Recall	F1	Time(s)
All Features								
ELM	0.002	**1.0**	0.501	**0.715**	0.157	0.002	0.003	0.72
ELM-S	0.581	0.489	0.535	0.515	0.313	0.581	0.405	65.2
LDA-I-A	0.643	0.362	0.501	0.442	0.334	0.640	0.296	154.5
ASimPLS	0.794	0.624	**0.709**	0.672	0.459	0.794	**0.581**	13.0
Subjective Words								
ELM	0.002	0.999	0.500	0.714	0.264	0.002	0.004	0.2
ELM-S	0.546	0.525	0.535	0.530	0.318	0.546	0.398	19.4
LDA-I-A	0.600	0.718	0.659	0.684	0.466	0.600	0.520	9.6
ASimPLS	0.596	0.720	0.658	0.684	0.464	0.596	0.519	3.7
UniGram								
ELM	0.001	0.999	0.500	0.714	0.120	0.000	0.001	0.426
ELM-S	0.538	0.509	0.524	0.518	0.305	0.538	0.389	37.6
LDA-I-A	0.648	0.720	0.684	0.699	0.480	0.648	0.551	35.1
ASimPLS	0.706	0.686	0.696	0.692	0.473	0.706	0.567	7.0
Other								
ELM	0.245	0.835	0.540	0.666	0.502	0.245	0.258	**0.1**
ELM-S	0.687	0.419	0.553	0.494	0.371	0.687	0.436	9.6
LDA-I-A	0.410	0.600	0.505	0.547	**0.537**	0.410	0.199	1.6
ASimPLS	**0.895**	0.236	0.566	0.423	0.324	**0.895**	0.472	1.6

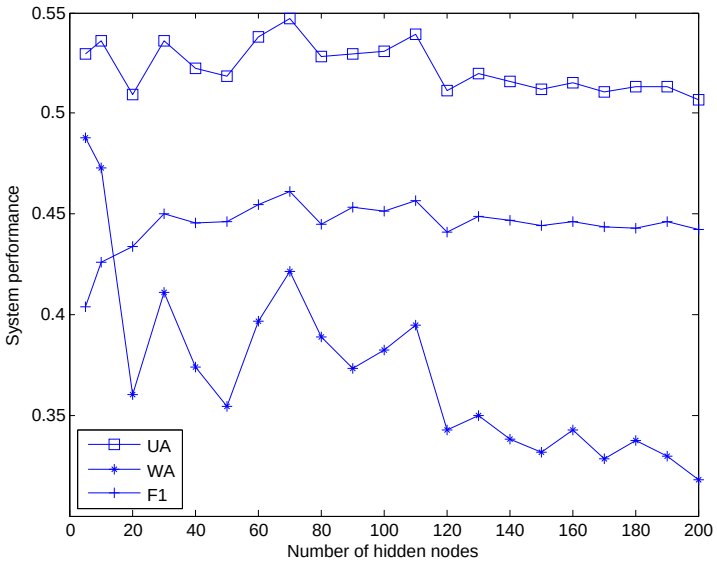


Figure 4.2: Performance of ELM-SMOTE with difference number of hidden nodes using the All Features set.

To investigate the influence to the performance caused by the number of hidden nodes, we conduct a set of five-fold cross validation experiments using ELM-SMOTE method. The results are shown in Fig. 4.2. One can see that the UA value and the F1 value did not change much when the number of hidden nodes grew. The value of WA decreases with the growing of hidden nodes number, which means that the accuracy of minority samples increases, while the accuracy of majority samples decreases. Hence, the UA and F1 values keep stable.

4.3.3 Performance of classifier fusion

To test a fusion method, we need to have training, development, and testing data sets. A five fold cross validation experiment was conducted in Section 4.3.2. We select the first three sentence groups of the first round as

Table 4.4: System performance of different fusion methods.

Classifier	TPR	TNR	% UA	% WA	Precise	Recall	F1
Train vs. Develop							
ELM-SW	0.589	0.421	0.505	0.470	0.293	0.589	0.391
ELM-UGram	0.542	0.526	0.534	0.531	0.318	0.542	0.401
ELM-Other	0.846	0.141	0.494	0.345	0.286	0.846	0.428
ASimPLS-SW	0.606	0.718	0.662	0.686	0.467	0.606	0.527
ASimPLS-UGram	0.696	0.692	0.694	0.693	0.479	0.696	0.567
ASimPLS-Other	**0.852**	0.377	0.614	0.514	0.357	**0.852**	0.503
Simple-Fusion	0.760	0.702	**0.731**	0.719	0.509	0.760	**0.610**
Focal-Fusion	0.663	**0.783**	0.723	**0.748**	**0.554**	0.663	0.604
Train+Develop vs. Test							
ELM-SW	0.483	0.614	0.549	0.576	0.340	0.483	0.399
ELM-UGram	0.505	0.519	0.512	0.515	0.302	0.505	0.378
ELM-Other	0.413	**0.851**	0.632	0.723	0.533	0.413	0.466
ASimPLS-SW	0.548	0.744	0.646	0.687	0.469	0.548	0.505
ASimPLS-UGram	0.705	0.676	0.690	0.684	0.472	0.705	0.565
ASimPLS-Other	**0.857**	0.305	0.581	0.466	0.337	**0.857**	0.484
Simple-Fusion	0.690	0.725	0.708	0.715	0.508	0.690	0.585
Focal-Fusion	0.670	0.767	**0.719**	**0.739**	**0.542**	0.670	**0.599**
Single classifiers with all features							
ELM-All	0.434	0.639	0.536	0.579	0.331	0.434	0.375
ASimPLS-All	0.770	0.642	0.706	0.679	0.470	0.770	0.583

the training set. The fourth group is taken as the development set, and the fifth group is set to be the testing set. The results obtained by single classifiers and the fusion system are listed in Table 4.4. The first half of the table shows the results obtained on the development data set. One can see that ASimPLS method with Other feature set obtained the best accuracy of predicting emotional sentences on both development and testing set. ELM method obtained best TNR value on the testing set. The Simple fusion and Focal fusion methods could obtain better UA, WA and F1 values than all the single classifiers on the development set. The parameters obtained on the development set are applied to the classification of testing set. The second

half of the table shows the results obtained on the testing set. We can find that both of the two fusion methods could generate better WA, Recall, and F1 values than the single classifiers. The last part shows the system performance of ELM and ASimPLS methods using all the features. One can find that the fusion methods also outperform single algorithms with all features.

4.4 Summary

In this chapter, ELM has been applied to predict emotional sentences in a story. The evaluation on a children story corpus showed that ELM method was very fast although it was affected by the imbalanced data set. A fusion system has been proposed to combine ELM and ASimPLS with three difference feature sets. The experimental results demonstrated that the fusion method could generate better performance than single classifiers.

Chapter 5

Emotional Sentence Identification using Data Association

5.1 Introduction

In this chapter, we propose a method to automatically identify emotional sentences in an article using the association relationship between words, bigrams, and sentences. We assume that emotional sentences have close relationship with emotional words and emotional bigrams. Emotion detection in text is formulated as a graph based ranking problem. A score is given to each sentence in an article, and the sentences with higher scores are taken as emotional ones. A mutual reinforcement learning method is proposed to solve the ranking problem. A set of emotional words is collected manually, and a set of emotional bigrams is extracted from the training corpus. The relationships between words, bigrams, and the testing sentences are calculated first, and an iterative method is introduced to calculate the scores of the sentences. We evaluate our method on the UIUC children's tales corpus, and the algorithm is proved to be efficient for detecting emotional sentences in text. The contributions of this chapter are highlighted as follows:

(i) the emotional sentences identification problem is formulated as a rank-

ing problem;

(ii) a mutual reinforcement method is presented to solve the graph based ranking problem; and

(iii) the mutual reinforcement method is applied to identify emotional sentences in an article, which employs the association between words, bigrams, and sentences.

The rest of the chapter is organized as follows: Section 5.2 describes the proposed algorithm of identifying emotional sentences in an article using data association. Our method is evaluated on the UIUC children's tales corpora and the results are reported in Section 5.3. We conclude our work in Section 5.4.

5.2 Emotional sentences detection in an article

In this chapter, we assume that a sentence is more emotional if it contains more emotional words and bigrams. Similarly, a word/bigram is more emotional if it appears in many emotional sentences and it is similar to many emotional bigrams/words. Based on the assumptions, an iteration method is presented to calculate the emotion scores of the sentences, words, and bigrams.

5.2.1 Mutual-reinforcement ranking

In this work, we first collect a set of emotional words and select a set of emotional bigrams from the training set. All the words, bigrams and sentences are given an equal initial emotional score. The mutual influence between

words, bigrams and testing sentences are then calculated to generate the final score of each sentence.

A set of emotional words are collected using WordNet Affect [83, 84] and Internet. To reduce the computing time, we only select the words that appear in the corpus to calculate the scores. A set of emotional bigrams are selected from the training sentences, and the method is shown in Algorithm 3. If a bigram contains an emotional word, it is considered as an emotional bigram. Otherwise, if a bigram appears more frequently in emotional sentences than in neutral sentences, it is also taken as an emotional bigram.

Algorithm 3 Select emotional bigrams from training data.

Input: a bigram $B = \{w_1, w_2\}$ and an emotional word set S
Output: 1 if B is an emotional bigram; 0 otherwise.
if $w_1 \in S$ or $w_2 \in S$ **then**
    return 1;
else
    $P_e = \frac{N_e}{S_e}$ where N_e is the times that B appears in emotional sentences, and S_e is the number of emotional sentences in the training set.
    $P_n = \frac{N_n}{S_n}$ where N_n is the times that B appears in neutral sentences, and S_n is the number of neutral sentences in the training set.
    if $P_e > P_n$ **then**
        return 1;
    else
        return 0;
    end if
end if

As shown in Fig. 5.1, an undirected graph is constructed for a given corpus, where the term vertexes, bigram vertexes and sentence vertexes are connected by the weighted edges in the graph. The weight of the edge con-

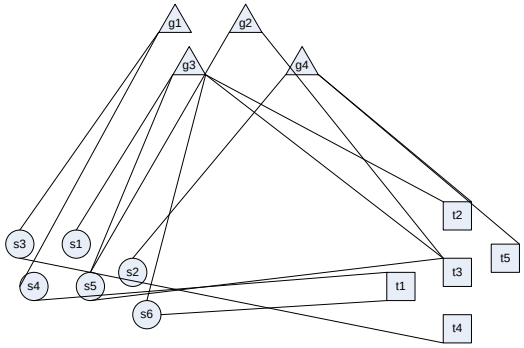


Figure 5.1: An undirected graph constructed for a document.

necting term t_l and sentence s_i is defined as

$$w_{st}(i,l) = \begin{cases} 1 & \text{if } t_l \text{ appears in } s_i \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

The weight of the edge connecting a sentence s_i and a bigram g_j is calculated by

$$w_{sg}(i,j) = \begin{cases} 1 & \text{if } g_j \text{ appears in } s_i \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

The weight of the edge connecting term t_l and a bigram g_j is

$$w_{gt}(j,l) = \begin{cases} 1 & \text{if } t_l \text{ appears in } g_j \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

Based on the assumptions raised in this chapter, the emotional score of a sentence is higher if it has close relationship with emotional words and bigrams.

We calculate the rank of each node in the graph with:

$$\begin{cases} r(s_i) = \alpha_1 \sum_{j=1}^{k} w_{sg}(i,j) r(g_j) + \beta_1 \sum_{l=1}^{m} w_{st}(i,l) r(t_l) \\ r(g_j) = \alpha_2 \sum_{i=1}^{n} w_{sg}(i,j) r(s_i) + \gamma_1 \sum_{l=1}^{m} w_{gt}(j,l) r(t_l) \\ r(t_l) = \beta_2 \sum_{i=1}^{n} w_{st}(i,l) r(s_i) + \gamma_2 \sum_{j=1}^{k} w_{gt}(j,l) r(g_j) \end{cases} \tag{5.4}$$

where $r(s_i)$ is the score of sentence s_i which will determine the rank of s_i, $r(g_j)$ is the rank of bigram g_j, and $r(t_l)$ is the rank of term t_l. The integer m, n and k are the number of terms, sentences and bigrams respectively. The weights $w_{st}(i,l)$, $w_{sg}(i,j)$ and $w_{gt}(j,l)$ are calculated by (5.1), (5.2) and (5.3) respectively. The parameters α_i, β_i and γ_i, $i = 1, 2$ balance the influence of different mutual effects to the ranks of all items.

The formula (5.4) can be written in the matrix form:

$$\begin{cases} \vec{r(s)} = \alpha_1 \mathbf{W}_{sg} \vec{r(g)} + \beta_1 \mathbf{W}_{st} \vec{r(t)} \\ \vec{r(g)} = \alpha_2 \mathbf{W}'_{sg} \vec{r(s)} + \gamma_1 \mathbf{W}_{gt} \vec{r(t)} \\ \vec{r(t)} = \beta_2 \mathbf{W}'_{st} \vec{r(s)} + \gamma_2 \mathbf{W}'_{gt} \vec{r(g)} \end{cases} \tag{5.5}$$

where $\vec{r(s)} = [r(s_1), \ldots, r(s_n)]^T$, $\vec{r(g)} = [r(g_1), \ldots, r(g_k)]^T$, $\vec{r(t)} = [r(t_1), \ldots, r(t_m)]^T$, $\mathbf{W}_{sg}(i,j) = w_{sg}(i,j)$, $\mathbf{W}_{st}(i,l) = w_{st}(i,l)$ and $\mathbf{W}_{gt}(j,l) = w_{gt}(j,l)$. The ranks are normalized after each iteration through:

$$\begin{cases} \vec{r(s)} = \vec{r(s)} / \|\vec{r(s)}\| \\ \vec{r(g)} = \vec{r(g)} / \|\vec{r(g)}\| \\ \vec{r(t)} = \vec{r(t)} / \|\vec{r(t)}\| \end{cases} \tag{5.6}$$

where $\|\vec{x}\| = \sqrt{\sum x_i^2}, \vec{x} = [x_i]^T, i = 1, 2, \cdots$. The ranks of terms, sentences and bigrams are initialized equally to be 1, and the algorithm is converged

when the difference between the two successive ranks below a given threshold (here we use $1e-4$).

5.2.2 Convergence analysis

As one of the fundamental problems of the iteration algorithm, convergence of the presented algorithm is discussed in this section. Our method can be written in the following format

$$\begin{cases} r(\vec{g}) = \alpha_1 \mathbf{W}_{gg} r(\vec{g}) + \beta_1 \mathbf{W}'_{sg} r(\vec{s}) + \gamma_1 \mathbf{W}_{gt} r(\vec{t}) \\ r(\vec{s}) = \alpha_2 \mathbf{W}_{sg} r(\vec{g}) + \beta_2 \mathbf{W}_{ss} r(\vec{s}) + \gamma_2 \mathbf{W}_{st} r(\vec{t}) \\ r(\vec{t}) = \alpha_3 \mathbf{W}'_{gt} r(\vec{g}) + \beta_3 \mathbf{W}'_{st} r(\vec{s}) + \gamma_3 \mathbf{W}_{tt} r(\vec{t}) \end{cases} \tag{5.7}$$

where $\mathbf{W}_{gg}$, $\mathbf{W}_{ss}$, and $\mathbf{W}_{tt}$ are zero matrixes. The vectors $\vec{r_g}$, $\vec{r_s}$, and $\vec{r_t}$ denote the ranking scores of bigrams, sentences, and terms. The feature $\mathbf{W}_{gg}$ is the bigram-bigram affinity matrix, $\mathbf{W}_{sg}$ is the sentence-bigram affinity matrix, $\mathbf{W}_{gt}$ is the bigram-term affinity matrix, and so on. α_i, β_i, and $\gamma_i, i = 1, 2, 3$ are weights to balance the relative weights among bigrams, sentences, and terms. It is worth noting that α_i, β_i, and γ_i here have no relationship with those in (5.4) and (5.5). In [93], it is proved that if the matrix

$$\mathbf{M} = \begin{bmatrix} \alpha_1 \mathbf{W}_{gg} & \beta_1 \mathbf{W}'_{sg} & \gamma_1 \mathbf{W}_{gt} \\ \alpha_2 \mathbf{W}_{sg} & \beta_2 \mathbf{W}_{ss} & \gamma_2 \mathbf{W}_{st} \\ \alpha_3 \mathbf{W}'_{gt} & \beta_3 \mathbf{W}'_{st} & \gamma_3 \mathbf{W}_{tt} \end{bmatrix} \tag{5.8}$$

is stochastic and irreducible, the algorithm will converge at a unique ranking.

The following revises are taken to make $\mathbf{M}$ stochastic and irreducible. (i) Let $\mathbf{X}$ denote any of the following three matrixes $\mathbf{W}_{gg}$, $\mathbf{W}_{ss}$, and $\mathbf{W}_{tt}$. We

revise $\mathbf{X}$ to $\bar{\mathbf{X}} = [1/k]_{k\times 1}[1]_{1\times k}$, where k is the order of $\mathbf{X}$. (ii) The matrixes $\mathbf{W}_{sg}$, $\mathbf{W}'_{sg}$, $\mathbf{W}_{gt}$, $\mathbf{W}'_{gt}$, $\mathbf{W}_{st}$ and $\mathbf{W}'_{st}$ are normalized by columns, and denoted by $\bar{\mathbf{W}}_{sg}$, $\bar{\mathbf{W}}'_{sg}$, $\bar{\mathbf{W}}_{gt}$, $\bar{\mathbf{W}}'_{gt}$, $\bar{\mathbf{W}}_{st}$ and $\bar{\mathbf{W}}'_{st}$. (iii) We force $\sum_{i=1}^{3}\alpha_i = \sum_{i=1}^{3}\beta_i = \sum_{i=1}^{3}\gamma_i = 1$. To alleviate the effect of matrix $\bar{\mathbf{X}}$, we set α_1, β_2 and γ_3 be a small number.

It can be proved that $\bar{\mathbf{M}} = \begin{bmatrix} \alpha_1\bar{\mathbf{W}}_{gg} & \beta_1\bar{\mathbf{W}}'_{sg} & \gamma_1\bar{\mathbf{W}}_{gt} \\ \alpha_2\bar{\mathbf{W}}_{sg} & \beta_2\bar{\mathbf{W}}_{ss} & \gamma_2\bar{\mathbf{W}}_{st} \\ \alpha_3\bar{\mathbf{W}}'_{gt} & \beta_3\bar{\mathbf{W}}'_{st} & \gamma_3\bar{\mathbf{W}}_{tt} \end{bmatrix}$ is stochastic and irreducible. We use $\bar{\mathbf{M}}$ to replace $\mathbf{M}$, and our method will converge to a unique ranking vector. In our experiments, the algorithm converges without adding the third restriction.

5.3 Experimental results

Our experiments of detecting emotional sentences are conducted on the UIUC Children's Story corpus [11] which consists of 176 stories by Grimm's, Andersen, and Potter. Every sentence in the corpus is annotated with one of the seven emotions: anger, disgust, fear, joy, sadness, surprise, and neutral by two annotators. Here we will only consider two classes: neutral sentences and non-neutral sentences. Five-fold cross validation was conducted to evaluate our method. The original form of the words in every sentence are obtained using the Stanford Parser [94]. We also obtain the bigrams of every sentence. We manually collected 3898 subjective words and a set of stop words. The emotional words appearing in the corpus are selected to calculate the associations. The stop words are removed from the word list. The emotional bigrams learned from the training data set are used to learn the associations.

True Positive Rate (TPR), True Negative Rate (TNR), unweighted accuracy (UA), weighted accuracy (WA), and F-Measures are employed to measure the system performances.

The system performance is shown in Table 5.1. SVM and Extreme Learning Machine (ELM) [87] are employed to compare with our algorithm. We use the set of emotional words and bigrams for calculating the associations as the features of SVM and ELM. If an emotional word or an emotional bigram appears in a sentence, we add a new dimension to the feature and the value is set to be 1. Otherwise, the value is set to be 0. The features obtained are about $5,000$ dimensions. No other features are employed for SVM and ELM as we would like to compare the system performance of different algorithms with the same feature set. As discussed in previous chapters, there are much more neutral sentences in an article than the emotional ones. Some methods like SVM and ELM are affected by the imbalanced data set. As shown in the table, the TNR obtained by these two methods are very high while the TPR are quite low. Over-sampling techniques like SMOTE [23] could alleviate the influence of the imbalanced data set. In our experiments, we over-sampled the positive training samples, and made the number of emotional sentences same as the number of neutral ones. One can see that the method proposed in this chapter (Ranking) could obtain a better TPR than SVM method. However, the TNR decreased about 15%. The ELMSmote method could get best TPR while a very low TNR. SVMSmote obtained 100% on TPR and 0% on TNR, which means the positive data is over over-sampled. The performance of methods using SMOTE technologies could increase if we adjust the number of positive samples that be over-sampled. However, it is unfair for the other methods as they are using the default parameters. Hence, we

Table 5.1: System performance obtained by different methods.

Methods	TPR	TNR	WA	UA	Precision	Recall	F1	Time
Ranking	0.496	0.743	0.660	0.620	0.496	0.496	0.496	19.7
SVM	0.454	0.891	0.744	0.673	0.681	0.454	0.544	4434.3
SVMSmote	0.999	0.001	0.337	0.500	0.337	0.999	0.504	9867.7
ELM	0.0	1.000	0.663	0.500	0.462	0.004	0.010	6.3
ELMSmote	0.614	0.393	0.468	0.503	0.347	0.614	0.432	1176.8

only report the performance of SVMSmote and ELMSmote with the default parameters here. The time consuming reported here includes the time of training a model and predicting the labels of new samples. One can see that the Ranking method and ELM method are faster than the other ones. SVMSMote and ELMSmote consume much more times than other methods as they need to over-sample the positive samples.

Table 5.2 shows the influence to the F1 value of weight parameters α, β and γ. To find the optimal parameters, we divide the whole data set into training, developing and testing data sets. As 5-fold cross validation was conducted above, the whole data set was divided into five groups randomly for five rounds. In this experiment, we select the first three sentence groups as the training set, the fourth group as the developing set, and the fifth group as the testing set for each round. The system performance reported is the average value of the performance obtained in the five rounds. In our experiments, we added the following restriction to parameters α_i, β_i and γ_i, $i = 1, 2$ in (5.4): $\alpha_1 = \alpha_2 = \alpha$, $\beta_1 = \beta_2 = \beta$, and $\gamma_1 = \gamma_2 = \gamma$ to make the weights of sentence-bigram association, sentence-term affinity, and bigram-term affinity consistent in the algorithm. All these parameters are set to be one of the following values: $0.0, 0.25, 0.50, 0.75, 1.0$ in our experiments. It can be seen from the table that when $\alpha = 1.0$, the best performance is obtained with $\beta = 0.75$ and $\gamma = 1.0$. The similar pattern can be found when

$\alpha = 0.25, 0.5$ and 0.75. When α is set to be one of the three values, the best performance is obtained when β is equal or a little less than α. One can also find that when the values of α and β are given ($\alpha \neq 0$ and $\beta \neq 0$), the system performance is influenced slightly by the value of γ. It means that the association of bigrams and terms does not help find the emotional sentences. To demonstrate the system performance more clearly, we plot the F1 values with different parameters in Figure 5.2. In the figure, one can find that when the value of α is given, the F1 value does not change much with the change of γ value. The highest F1 value, which is filled with dark red color, is obtained when β is equal to α or it is slightly less than it. The parameters are applied on the testing set. The results are shown in Table 5.3. One can find that the system performance did not change much because the default parameters $1, 1, 1$ is already close to the optimal ones $1, 0.75, 1.0$.

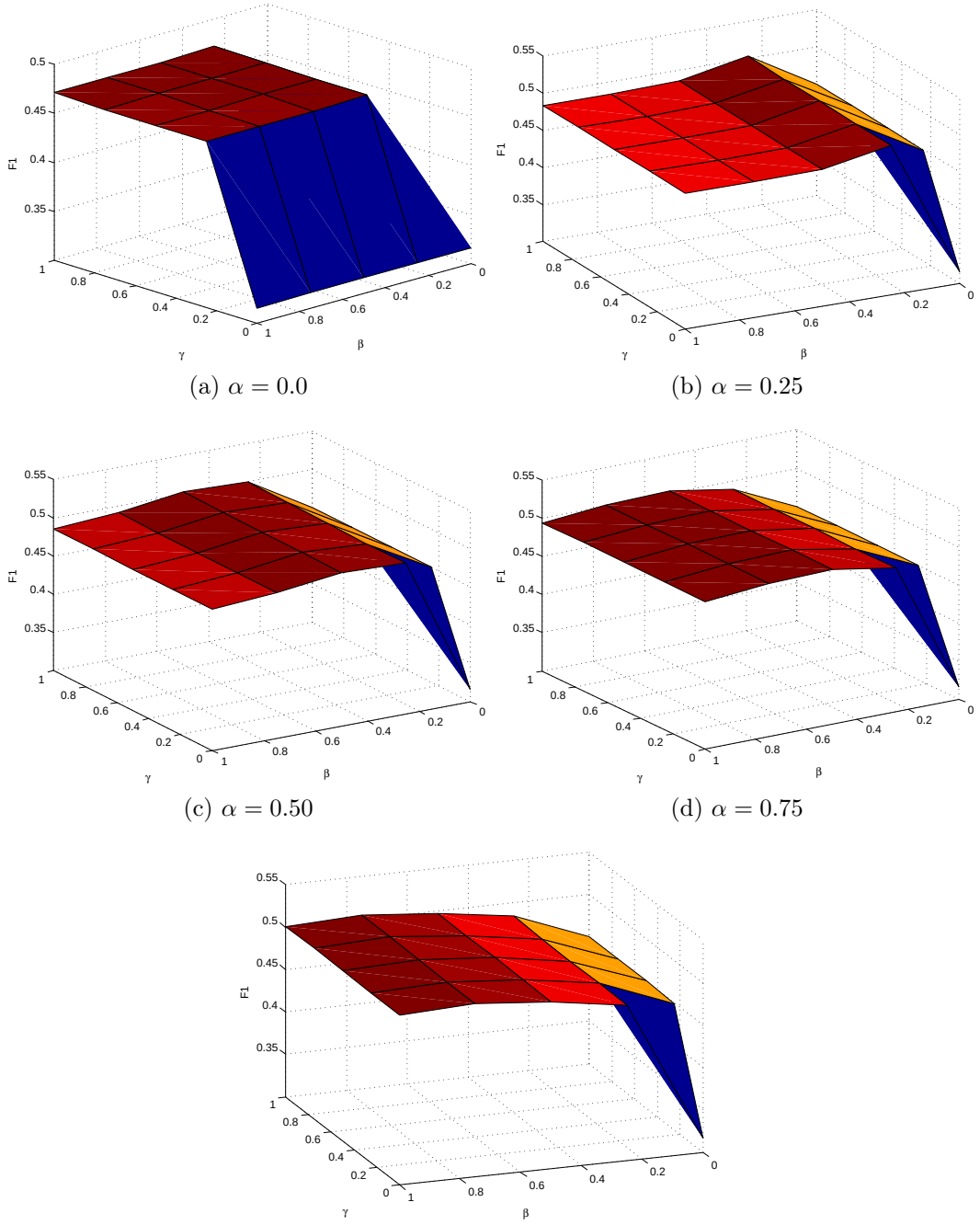
In the UIUC corpus, the two annotators may annotate different labels for a same sentence. We select the sentences that are labeled with same emotions by the two annotators to evaluate our algorithm. Ideally, the results should be better than those obtained using the sentences annotated by the first annotator because more people agree with the sentence labels. From Table 5.4, one can find that the TPR obtained by the Ranking method is lower compared to Table 5.1, while the TNR is higher. This is because that the corpus becomes more imbalanced. In the whole data set, there are 5163 emotional sentences and 10139 neutral sentences. In the highly agree corpus, we have only 2403 emotional sentences and 6016 neutral sentences. The ratio of emotional sentences to neutral sentences decreases from 0.5 to 0.4, which affects the system performance.

Table 5.2: F1 values with different parameters.

$\alpha = 0.0$						
		γ				
		0.0	0.25	0.5	0.75	1.0
β	0.0	0.316	0.316	0.316	0.316	0.316
	0.25	0.316	0.471	0.471	0.472	0.473
	0.5	0.316	0.470	0.471	0.471	0.471
	0.75	0.316	0.470	0.470	0.471	0.471
	1.0	0.316	0.470	0.470	0.470	0.471
$\alpha = 0.25$						
		γ				
		0.0	0.25	0.5	0.75	1.0
β	0.0	0.316	0.450	0.450	0.451	0.451
	0.25	0.501	0.501	0.502	0.502	0.504
	0.5	0.484	0.484	0.485	0.485	0.486
	0.75	0.483	0.483	0.483	0.483	0.483
	1.0	0.483	0.483	0.483	0.483	0.483
$\alpha = 0.5$						
		γ				
		0.0	0.25	0.5	0.75	1.0
β	0.0	0.316	0.450	0.450	0.450	0.450
	0.25	0.497	0.496	0.497	0.498	0.499
	0.5	0.500	0.501	0.501	0.501	0.502
	0.75	0.490	0.490	0.490	0.490	0.491
	1.0	0.484	0.484	0.484	0.485	0.485
$\alpha = 0.75$						
		γ				
		0.0	0.25	0.5	0.75	1.0
β	0.0	0.316	0.450	0.450	0.450	0.450
	0.25	0.489	0.488	0.489	0.489	0.489
	0.5	0.503	0.503	0.503	0.503	0.503
	0.75	0.501	0.501	0.501	0.501	0.501
	1.0	0.494	0.494	0.494	0.494	0.494
$\alpha = 1.0$						
		γ				
		0.0	0.25	0.5	0.75	1.0
β	0.0	0.316	0.449	0.450	0.450	0.450
	0.25	0.483	0.483	0.483	0.483	0.484
	0.5	0.497	0.497	0.497	0.497	0.497
	0.75	0.504	0.504	0.504	**0.505**	**0.505**
	1.0	0.501	0.501	0.501	0.501	0.501

Table 5.3: System performance obtained on the testing data set with adjusted parameters.

Parameters	TPR	TNR	WA	UA	Precision	Recall	F1
1,1,1	0.355	0.669	0.564	0.512	0.351	0.355	0.353
1,0.75,1.0	0.355	0.669	0.563	0.512	0.350	0.355	0.352



(a) $\alpha = 0.0$

(b) $\alpha = 0.25$

(c) $\alpha = 0.50$

(d) $\alpha = 0.75$

(e) $\alpha = 1.0$

Figure 5.2: ROUGE scores under different values of balance parameters α, β and γ.

Table 5.4: System performance of different methods on the highly agree corpus.

Methods	TPR	TNR	WA	UA	Precision	Recall	F1	Time
Ranking	0.411	0.765	0.664	0.588	0.411	0.411	0.411	7.4
SVM	0.388	0.939	0.781	0.663	0.717	0.388	0.503	340.8
SVMSmote	0.969	0.005	0.280	0.487	0.280	0.969	0.434	1510.2
ELM	0.0	1.000	0.715	0.500	0.0	0.0	0.0	1.4
ELMSmote	0.644	0.345	0.430	0.495	0.289	0.644	0.390	158.6

5.3.1 Discussion

From the experimental results, one can find that the values of γ, which denote the weights of bigram-term affinity in the iterative algorithm, did not affect the system performance heavily. One possible reason is that the method of computing the association between bigram-term in Eq. (5.3) is too simple. If a term appears in a bigram, the association degree is set to be 1. Otherwise, the association degree is set to be 0. This kind of relationship is already counted in the sentence-bigram association and sentence-term association. We tried another method of calculating bigram-term association, which is shown in (5.9).

$$w_{gt}(j,l) = \begin{cases} 1 & \text{if } t_l \text{ appears in } g_j \\ \max(sim(t_l, w_i)), i = 1,2 & \text{otherwise} \end{cases} \tag{5.9}$$

where w_1 and w_2 are two words in the bigram g_j. The function $sim(t_l, w_i)$ calculates the semantic similarity between two words. For each word, we obtain its TFIDF values using about 50,000 documents obtained on Wikipedia. Each word is represented as a vector, and the item in the vector is the TFIDF values. The similarities between words are calculated using the cosine simi-

larity between the two vectors:

$$sim(\mathbf{w}_1, \mathbf{w}_2) = \frac{\mathbf{w}_1 \cdot \mathbf{w}_2}{\| \mathbf{w}_1 \| \| \mathbf{w}_2 \|} \tag{5.10}$$

With this method of calculating bigram-term affinity, the similarity between a term and a bigram is considered. Then more associations will be found. However, some noise will be added if the similarity is not accurate enough. The system performance decreases using the association degrees of bigrams and terms obtained by this method. When the values of α and β are given ($\alpha \neq 0$ and $\beta \neq 0$), the system performance decreases if the value of γ increases. It means that the association between bigram-term does not help in finding emotional sentences. A better method of measuring association between bigrams and terms should be found in the future work.

In our experiments, we did not consider the relationship between bigram-bigram, sentence-sentence, and term-term because there is heavy increase of the computing time, and the methods of computing association between these items need to be investigated. In the future work, efficient methods considering these associations will be developed.

5.4 Summary

In this chapter, a method has been proposed to predict emotional sentences in an article. The emotion identification in text has been formulated as a ranking problem, and a mutual reinforcement learning method has been presented to solve the problem, which employed the association between words, bigrams, and sentences. Experimental results on a story corpus proved

the efficiency of the proposed method.

Chapter 6

Mutual-reinforcement Document Summarization using Data Association

6.1 Introduction

In this chapter, the method proposed in last chapter is applied on document summarization to extract essential sentences from a document by exploiting the mutual effects between terms, sentences and clusters. Three phrases are included in this method: document modeling, sentence clustering and sentence ranking. A document is modeled by a weighted graph with vertexes that represent sentences of the document. The sentences are clustered into different groups to find the latent topics in the article. To alleviate the influence of unrelated sentences in clustering, an embedding process is employed to optimize the document model. The sentences are then ranked according to the mutual effect between terms, sentence as well as clusters, and high-ranked sentences are selected to comprise the summarization of the document. The experimental results on the Document Understanding Conference (DUC) data sets demonstrate the effectiveness of the proposed

method in document summarization. The results also show that the embedding process for sentence clustering render the system more robust with respect to different cluster numbers.

Graph based ranking has drawn much attention in the machine learning field. It can be applied in protein ranking, information retrieval, and web page ranking etc. In the web page ranking, one page transports some weight to the pages connected to it. The PageRank [95, 96] algorithm calculates the weight of a page recursively by weighted summing the weights of all pages that link to it. A page linked to by many important pages is taken as an important page itself. The method has been extended and applied in document summarization. In the task of extracted based document summarization, the sentences of a document is ranked, and those with high rankings are selected as the summary of a document. For the sentence ranking, the weight is normally defined as the similarity between sentences. An extension of the PageRank algorithm, the topic-sensitive PageRank algorithm [97], was also employed for query-oriented summarization [98].

Extracting central sentences from a document can be formulated as a sentence ranking problem. In [96], a graph for the document is constructed where the vertexes of the graph denote sentences in the document, and the weight of each edge in the graph is the similarity between two connected sentences. To calculate the similarity, each sentence is represented by a vector, and the value of each word in the vector is the number of occurrences of the word in the sentence times the inverse document frequency of the word. The PageRank algorithm is implemented to rank all the sentences based on the graph model. In this method, the sentences that are similar with other important sentences will obtain high scores. The similarity can be considered

as the mutual-effect between sentence and sentence. While the iterative ranking method considering mutual-reinforcement between different items has also been proved to be effective for document summarization [99, 100, 93]. A mutual reinforcement method that considers relationship of terms and sentences was proposed in [100]. In this work, the saliency score of a term is determined by the sentences that it appears in, and the saliency score of a sentence is determined by the terms that it contains. The sentences in the document are partitioned into clusters and the ranking method is performed within each cluster. The algorithm is extended in [99] where the homogeneous relationship between words, the homogeneous relationship between sentences, and the heterogeneous relationship between words and sentences are all taken into account for calculating the scores of sentences. Moreover, three granularities were considered in [93] including document, sentence and term.

It has been proved that the sentences of a set of documents can be clustered into different groups, which were not considered in the above methods, to represent subtopics of the documents [101]. In this chapter, we propose a framework which considers the mutual effect between clusters, sentences and terms instead of the relationship between documents, sentences, and terms to employ the cluster level information and the latent theme information in the clusters for document summarization. A matrix and a weighted undirected graph model are first constructed for a document, where column vectors of the matrix and the vertexes of the graph represent sentences of the document. The sentences are then clustered into different groups according to the distance between two sentences. In order to alleviate the influence of unrelated sentences in sentence clustering, we employ an embedding process

to optimize the graph vertexes. A mutual-reinforcement algorithm is performed at last to calculate the ranks of all sentences. The sentences with high rankings are extracted as the summarization. The contributions of this chapter are summarized as follows:

(i) an embedded graph based sentence clustering method is proposed for sentence grouping of a document, which is robust with respect to different cluster numbers;

(ii) an iterative ranking method is presented considering the mutual-reinforcement between terms, sentences and sentence clusters; and

(iii) a document summarization framework considering sentence cluster information is proposed and the framework is evaluated using DUC data sets.

The remainder of this chapter is organized as follows: The proposed mutual-reinforcement ranking algorithm is presented in Section 6.2. Section 6.3 describes the experiments and analyzes the influence to system performance of different parameters. We conclude our work in Section 6.4.

6.2 Sentence Ranking Using Embedded Graph Based Sentence Clustering

In this chapter, document summarization is performed following three steps: document modeling, sentence clustering and sentence ranking. A document is modeled by a matrix with each sentence as a column vector. A weighted graph model is also built with sentences as its vertexes, and the weights of

the edges in the graph denote the distances between two sentences. Here the cosine similarities of sentence vectors are taken as distances between them. The sentences are clustered into different groups to find latent subtopic information in a document. The low similarities between unrelated sentences affect the accuracy of the sentence clustering. To reduce the influence of the low weights and enhance sentence clustering performance, an embedding algorithm is performed on the the graph. After mapping the vectors of sentences to a low dimensional space, the distances between sentences are calculated based on new vectors and the sentences are clustered using the new distances. At last, the sentences are ranked according to the mutual effects between sentences, terms and clusters based on the assumptions: (1) a sentence has a high rank if it is similar with many high ranking clusters and it contains many high ranking terms; (2) the rank of a cluster is high if it contains many high ranking sentences and many high ranking terms; (3) the rank of a term is high if it appears in many high ranking sentences and clusters. The affinity of term-sentence and cluster-sentence determine the ranks of sentences. The ranks of terms and clusters are determined by similar relationships accordingly. The sentences that obtain high ranks are selected as the content of a summary.

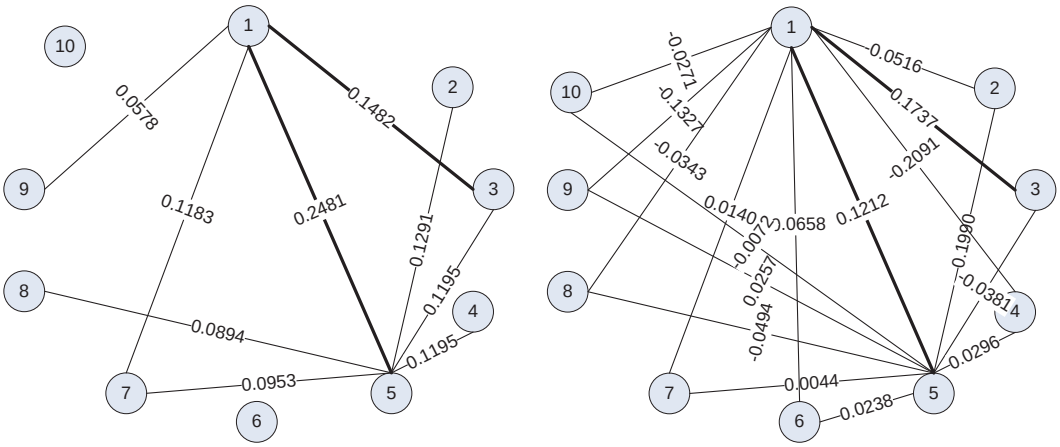
6.2.1 Document Modeling

To facilitate the document summarization process, we model a document with a text matrix $\mathbf{D} = [\mathbf{s}_1, ..., \mathbf{s}_n] \in \mathbb{R}^{m \times n}$, where m is the number of terms and n is the number of sentences of the document, and the column vectors $\mathbf{s}_i, i = 1, ..., n$, of $\mathbf{D}$ denote weighted term-frequency vectors of sentences. To

build the vector $\mathbf{s}_i$ of sentence s_i, stop words in the sentence are removed first and word stemming is performed. The term $T = \{t_1, t_2, \ldots, t_m\}$ denotes the complete term set of the document, and $\mathbf{s}_i = [s_{i1}, s_{i2}, ..., s_{im}]^T$ where $s_{ij} = 1$ if term t_j appears in sentence s_i and $s_{ij} = 0$ otherwise. Sentences are clustered into different topic groups in this work. To have a better presentation of the algorithm, a graph $G = \{V, E\}$ is built to model a document, where V is the vertex set of the graph representing sentences and E is the edge set of the graph. An edge is bridged between two vertexes if the similarity between the two sentences is greater than 0. The weight w_{ij} of an edge denotes the distance between sentences s_i and s_j which is the cosine similarity between the vectors of two sentences, i.e.,

$$d_{ij} = w_{sim}(\mathbf{s}_i, \mathbf{s}_j) \tag{6.1}$$

where the vectors $\mathbf{s}_i$ and $\mathbf{s}_j$ are weighted term-frequency vectors of sentence s_i and s_j. The cosine similarity is defined by (5.10). For an intuitive view, Fig. 6.1a illustrates a weighted undirected graph constructed for the document $fbis4 - 45908$ of topic set $d05$ in DUC-2001 which has 44 sentences and 374 tokens, where the vertexes of the graph represent the sentences to be ranked. In this figure, only the distances between Sentence 1 s_1 and the first 10 sentences, and the distances between Sentence 5 s_5 and the first 10 sentences are presented to make the figure clear. It can be seen from the figure that the cosine similarity between Sentence 1 and Sentence 3, 5, 7, 9 are all larger than 0, and $w_{sim}(s_1, s_i) = 0,\ i = 2, 4, 6, 8, 10$. The similarities $w_{sim}(s_1, s_3)$ and $w_{sim}(s_1, s_5)$ are greater than those between the other sentences, which are represented by thicker edges in the figure.



(a) Weighted Graph (b) Embedded Weighted Graph

Figure 6.1: An illustration of a graph constructed for a document.

6.2.2 Embedded Graph Based Sentence Clustering

Based on the graph model, the sentences are clustered into different groups according to the cosine similarities between sentences. However, the accuracy of the clustering will be diminished if all the edges with $w_{sim}(s_i, s_j) > 0$ are counted. For the document shown in Fig. 6.1a, the weight of the edge between sentence s_1 and sentence s_9 is extremely small. The possible reason is that sentence s_9 represents totally different semantic meanings with s_1, but they have a few common words like “go" and “take". In this case, s_1 is not related with s_9, while the distance between them is considered in sentence clustering. The phenomenon is prevalent and affects the clustering results. Although removing the stop words which appear frequently can reduce the affection of this kind of words, the stop words list is created manually and some words are stop words for some sentences while they are not neglectable for the others. To alleviate the influence of unrelated sentences, we embed the original matrix $\mathbf{D}$ of a document into a lower dimensional space inspired

by Locally Linear Embedding (LLE) [102, 103]. A sentence $\mathbf{d}_i$ which is a column vector of $\mathbf{D}$ is expressed as a linear combination of its n_i most similar sentences $\mathbf{d}_j, j = 1, 2, \ldots, n_i$, with

$$\mathbf{d}_i = \sum_{j \in \Omega_i} \gamma_{ij} \mathbf{d}_j \tag{6.2}$$

where Ω_i is the set of sentences most similar to $\mathbf{d}_i$. The dimensions of vectors $\mathbf{d}'_i$ and $\mathbf{d}'_j$ are reduced while keeping the linear combination relationship after embedding. The document matrix $\mathbf{D}$ is normalized before the embedding operation, and the cosine similarity in (6.1) can be written as

$$w_{sim}(\mathbf{d}_i, \mathbf{d}_j) = \frac{\mathbf{d}_i \cdot \mathbf{d}_j}{\|\mathbf{d}_i\| \times \|\mathbf{d}_j\|} = \mathbf{d}_i \cdot \mathbf{d}_j. \tag{6.3}$$

The Euclidean distance of two vectors is

$$\|\mathbf{d}_i - \mathbf{d}_j\| = \sqrt{\|\mathbf{d}_i\|^2 + \|\mathbf{d}_j\|^2 - 2\mathbf{d}_i \cdot \mathbf{d}_j} = \sqrt{2 - 2\mathbf{d}_i \cdot \mathbf{d}_j}. \tag{6.4}$$

From (6.3) and (6.4), it can be seen that the cosine similarity and the Euclidean distance measurement are equivalent in estimating the sentence relationship if $\|\mathbf{d}_i\| = \|\mathbf{d}_j\| = 1$. We adopt Euclidean distance in the following deduction for simplicity of presentation. In graph embedding, we minimize the following cost function of approximation error to determine the optimal weight matrix γ_{ij}

$$\varepsilon(\mathbf{W}) = \sum_i \|\mathbf{d}_i - \sum_{j \in \Omega_i} (\gamma_{ij} \mathbf{d}_j)\|^2 \tag{6.5}$$

subject to the constraint

$$\sum_{j \in \Omega_i} \gamma_{ij} = 1 \tag{6.6}$$

where $\gamma_{ij} = 0$ for $j \notin \Omega_i$, and $\mathbf{W}$ is the weight matrix composed by γ_{ij}. To calculate the solution of γ_{ij}, the approximation error cost function (6.5) is rewritten as in [102, 103]

$$\begin{aligned} \varepsilon(\mathbf{W}_i) &= \|\mathbf{d}_i - \textstyle\sum_{j\in\Omega_i}(\gamma_{ij}\mathbf{d}_j)\| \\ &= \|\mathbf{d}_i \textstyle\sum_{j\in\Omega_i}\gamma_{ij} - \sum_{j\in\Omega_i}(\gamma_{ij}\mathbf{d}_j)\| \\ &= \|\textstyle\sum_{j\in\Omega_i}\gamma_{ij}(\mathbf{d}_i - \mathbf{d}_j)\| \\ &= \textstyle\sum_{j\in\Omega_i}\gamma_{ij}\sum_{k\in\Omega_i}\gamma_{ik}(\mathbf{d}_i - \mathbf{d}_j)^T(\mathbf{d}_i - \mathbf{d}_k) \end{aligned} \tag{6.7}$$

where $\mathbf{W}_i = [\gamma_{i1}, \ldots, \gamma_{in_i}]$ are the weights connecting $\mathbf{d}_i$ to its neighbors. Let $\mathbf{C}_i(j,k) = (\mathbf{d}_i - \mathbf{d}_j)^T(\mathbf{d}_i - \mathbf{d}_k)$, and apply Lagrange multiplier η to (6.7). The error cost becomes

$$\varepsilon(\mathbf{W}_i) = \sum_{j\in\Omega_i}\gamma_{ij}\sum_{k\in\Omega_i}\gamma_{ik}\mathbf{C}_i(j,k) + \eta_i(\sum_{j\in\Omega_i}\gamma_{ij} - 1) \tag{6.8}$$

By requiring the partial derivatives with respect to each weight γ_{ij} to be zero

$$\frac{\partial\varepsilon(\mathbf{W}_i)}{\partial\gamma_{ij}} = \sum_{k\in\Omega_i}\gamma_{ik}\mathbf{C}_i(j,k) + \eta_i = 0,\ \forall j \in \Omega_i, \tag{6.9}$$

the solution $\mathbf{W}_i$ is found by solving the equations

$$\begin{aligned} \textstyle\sum_{k\in\Omega_i}\gamma_{ik}\mathbf{C}_i(j,k) + \eta_i &= 0 \\ \textstyle\sum_{k\in\Omega_i}\gamma_{ik} &= 1 \end{aligned} \tag{6.10}$$

The vectors of embedded sentences with enhanced relationship $\mathbf{d}'_i, i =$

$1, 2, \ldots, n$, are obtained by minimizing the cost function

$$\Phi(\mathbf{D}') = \sum_i \|\mathbf{d}'_i - \sum_{j \in \Omega_i} \gamma_{ij} \mathbf{d}'_j\|^2 \tag{6.11}$$

where $\mathbf{D}' = [\mathbf{d}'_1, \ldots, \mathbf{d}'_n]$ consists of the sentence vectors embedded into the low-dimensional space and n is the number of sentences. The cost function (6.11) can be written in matrix form as

$$\begin{aligned} \Phi(\mathbf{D}') &= \mathrm{Tr}[(\mathbf{D}' - \mathbf{D}'\mathbf{W})^T(\mathbf{D}' - \mathbf{D}'\mathbf{W})] \\ &= \mathrm{Tr}[(\mathbf{D}' - \mathbf{D}'\mathbf{W})(\mathbf{D}' - \mathbf{D}'\mathbf{W})^T] \\ &= \mathrm{Tr}[\mathbf{D}'(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W})^T\mathbf{D}'^T] \\ &= \mathrm{Tr}[\mathbf{D}'\bar{\mathbf{W}}\mathbf{D}'^T] \end{aligned} \tag{6.12}$$

where the symmetric matrix $\bar{\mathbf{W}} = (\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W})^T$. To make the problem well-posed, the following constraints are added [104]

$$\begin{aligned} \textstyle\sum_{i=1}^n \mathbf{d}'_i &= 0 \\ \tfrac{1}{n}\mathbf{D}'\mathbf{D}'^T &= \mathbf{I} \end{aligned} \tag{6.13}$$

where the first constraint is to assure that coordinates d'_i are centered at the origin, and the second constraint sets the covariance of the embedding vectors to unity. The minimum of (6.12) can be obtained by finding the $m'+1$ smallest eigenvectors of $\bar{\mathbf{W}}$ discarding the bottom eigenvector according to the Rayleigh-Ritz theorem, and the minimal value of $\Phi(\mathbf{D}')$ equals to the sum of the eigenvalues of $\bar{\mathbf{W}}$. To summarize, the $m' \times n$ embedded matrix $\mathbf{D}'$ consists of the $2nd, \cdots, (m'+1) - th$ eigenvectors of $\bar{\mathbf{W}}$ as its rows.

The high weight edges in the graph are kept, and low weight ones are

removed after the embedding is performed. An example is shown in Figures 6.1a and 6.1b. The high weight relationship such as $w_{sim}(s_1, s_3)$, $w_{sim}(s_1, s_5)$ and $w_{sim}(s_1, s_7)$ are kept positive, and the low weight like $w_{sim}(s_1, s_9)$ is adjusted to be negative. The similarities between s_5 and other sentences are updated in the same manner. The high weight relationship is enhanced and the low weight relationship is reduced after the graph embedding. It is noteworthy that $w_{sim}(s_1, s_5) > w_{sim}(s_1, s_3)$ in Fig. 6.1a, while $w_{sim}(s_1, s_5) < w_{sim}(s_1, s_3)$ after the embedding operation. The weight before embedding is a measure of the relationship between two sentences while the embedding operation keeps the relationship between a sentence and a set of neighbors. As a result, some weights will change relatively after the embedding operation. The performance of an embedding operation will improve if there are more points in the graph. However, sometimes the sentence number is very small comparing to the dimension of each point in document summarization, which leads to a failed embedding.

After the embedding, the distances between sentences are calculated based on new vectors in $\mathbf{D}'$ and the sentence clustering is performed using the new distances.

6.2.3 Mutual-reinforcement ranking

In [93], the authors proposed a document summarization method calculating the mutual effect between documents, sentences, and terms. To employ the cluster-level information and the latent theme information in the clusters for document summarization, we calculate the mutual affect between clusters, sentences and terms. As shown in Fig. 6.2, an undirected graph is constructed

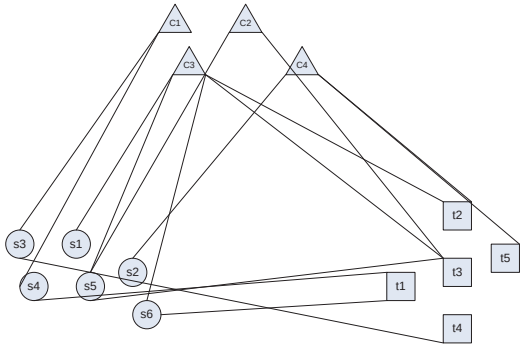


Figure 6.2: An undirected graph constructed for a document.

for a given document, where the term vertexes, sentence vertexes and cluster vertexes are connected by the weighted edges in the graph. The weight of the edge connecting term t_l and sentence s_i is defined as

$$w_{st}(i,l) = \mathbf{D}(l,i) \tag{6.14}$$

where $\mathbf{D}$ is the sentence-term matrix of the document. The weight of the edge connecting sentence s_i and cluster c_j is calculated by

$$w_{sc}(i,j) = w_{sim}(s_i, c_j) = \frac{\mathbf{s}_i \cdot \mathbf{c}_j}{\|\mathbf{s}_i\| \times \|\mathbf{c}_j\|} \tag{6.15}$$

where cluster c_j is represented by a vector which is summary of the vectors of all the sentences in this cluster $\mathbf{c}_j = \sum_{s \in c_j} \mathbf{s}$. The weight of the edge connecting term t_l and cluster c_j is

$$w_{ct}(j,l) = \mathbf{c}_j(l) \tag{6.16}$$

We first cluster the sentences and then calculate the rank of each sentence with:

$$\begin{cases} r(s_i) = \alpha_1 \sum_{j=1}^{k} w_{sc}(i,j) r(c_j) + \beta_1 \sum_{l=1}^{m} w_{st}(i,l) r(t_l) \\ r(c_j) = \alpha_2 \sum_{i=1}^{n} w_{sc}(i,j) r(s_i) + \gamma_1 \sum_{l=1}^{m} w_{ct}(j,l) r(t_l) \\ r(t_l) = \beta_2 \sum_{i=1}^{n} w_{st}(i,l) r(s_i) + \gamma_2 \sum_{j=1}^{k} w_{ct}(j,l) r(c_j) \end{cases} \tag{6.17}$$

where $r(s_i)$ is the rank of sentence s_i, $r(c_j)$ is the rank of cluster c_j, and $r(t_l)$ is the rank of term t_l. The integer m, n and k are numbers of terms, sentences and clusters respectively. The weights $w_{st}(i,l)$, $w_{sc}(i,j)$ and $w_{ct}(j,l)$ are calculated by (6.14), (6.15) and (6.16) respectively. The parameters α_i, β_i and γ_i, $i = 1, 2$ balance the influence of different mutual effects to the ranks of all items. In our experiments, the system performance did not increase significantly by considering the relationships between cluster-cluster, sentence-sentence, and term-term, while there is heavy increase of the computing time. Hence, we did not consider the above relationships. The formula (6.17) can be written in a matrix form:

$$\begin{cases} \mathbf{r}_s = \alpha_1 \mathbf{W}_{sc} \mathbf{r}_c + \beta_1 \mathbf{W}_{st} \mathbf{r}_t \\ \mathbf{r}_c = \alpha_2 \mathbf{W}_{sc}^T \mathbf{r}_s + \gamma_1 \mathbf{W}_{ct} \mathbf{r}_t \\ \mathbf{r}_t = \beta_2 \mathbf{W}_{st}^T \mathbf{r}_s + \gamma_2 \mathbf{W}_{ct}^T \mathbf{r}_c \end{cases} \tag{6.18}$$

where $\mathbf{r}_s = [r(s_1), \ldots, r(s_n)]^T$, $\mathbf{r}_c = [r(c_1), \ldots, r(c_k)]^T$, $\mathbf{r}_t = [r(t_1), \ldots, r(t_m)]^T$, $\mathbf{W}_{sc}(i,j) = w_{sc}(i,j)$, $\mathbf{W}_{st}(i,l) = w_{st}(i,l)$ and $\mathbf{W}_{ct}(j,l) = w_{ct}(j,l)$. The rank

is normalized after each iteration through:

$$\begin{cases} \mathbf{r}_s = \mathbf{r}_s / \|\mathbf{r}_s\| \\ \mathbf{r}_c = \mathbf{r}_c / \|\mathbf{r}_c\| \\ \mathbf{r}_t = \mathbf{r}_t / \|\mathbf{r}_t\| \end{cases} \tag{6.19}$$

where $\|\mathbf{x}\| = \sqrt{\sum x_i^2}, \mathbf{x} = [x_i]^T, i = 1, 2, \cdots$. The ranks of terms, sentences and clusters are initialized equally to be 1, and the algorithm is converged when the difference between the two successive ranks below a given threshold (here we use $1e-4$).

After the sentence ranking is obtained, we select sentences to compose the document summary. All the sentences are marked as unused at the beginning. The sentence with the highest score is selected as a part of the summarization and it is marked as used. The highest ranking sentence in the unused sentence set is added to the summary following the order of sentences in the given document. This sentence is also marked as used. The process is repeated until the desired length of the summarization is achieved.

6.2.4 Convergence analysis

As one of the fundamental problems of the iteration algorithm, convergence of the presented algorithm is discussed in this section. Our method can be written in the following format

$$\begin{cases} \mathbf{r}_c = \alpha_1 \mathbf{W}_{cc} \mathbf{r}_c + \beta_1 \mathbf{W}_{sc}^T \mathbf{r}_s + \gamma_1 \mathbf{W}_{ct} \mathbf{r}_t \\ \mathbf{r}_s = \alpha_2 \mathbf{W}_{sc} \mathbf{r}_c + \beta_2 \mathbf{W}_{ss} \mathbf{r}_s + \gamma_2 \mathbf{W}_{st} \mathbf{r}_t \\ \mathbf{r}_t = \alpha_3 \mathbf{W}_{ct}^T \mathbf{r}_c + \beta_3 \mathbf{W}_{st}^T \mathbf{r}_s + \gamma_3 \mathbf{W}_{tt} \mathbf{r}_t \end{cases} \tag{6.20}$$

where $\mathbf{W}_{cc}$, $\mathbf{W}_{ss}$, and $\mathbf{W}_{tt}$ are zero matrixes. The vectors $\mathbf{r}_c$, $\mathbf{r}_s$, and $\mathbf{r}_t$ denote the ranking scores of clusters, sentences, and terms. The term $\mathbf{W}_{cc}$ is the cluster-cluster affinity matrix, $\mathbf{W}_{sc}$ is the sentence-cluster affinity matrix, $\mathbf{W}_{ct}$ is the cluster-term affinity matrix, and so on. α_i, β_i, and $\gamma_i, i = 1, 2, 3$ are weights to balance the relative weight among cluster, sentence, and term. It is worth noting that α_i, β_i, and γ_i here have no relationship with those in (6.17) and (6.18). In [93], it is proved that if the matrix

$$\mathbf{M} = \begin{bmatrix} \alpha_1\mathbf{W}_{cc} & \beta_1\mathbf{W}_{sc}^T & \gamma_1\mathbf{W}_{ct} \\ \alpha_2\mathbf{W}_{sc} & \beta_2\mathbf{W}_{ss} & \gamma_2\mathbf{W}_{st} \\ \alpha_3\mathbf{W}_{ct}^T & \beta_3\mathbf{W}_{st}^T & \gamma_3\mathbf{W}_{tt} \end{bmatrix} \tag{6.21}$$

is stochastic and irreducible, out algorithm will converge at a unique ranking.

The following revises are taken to make $\mathbf{M}$ stochastic and irreducible. (i) Let $\mathbf{X}$ denote any of the following three matrixes $\mathbf{W}_{cc}$, $\mathbf{W}_{ss}$, and $\mathbf{W}_{tt}$. We revise $\mathbf{X}$ to $\bar{\mathbf{X}} = [1/k]_{k\times 1}[1]_{1\times k}$, where k is the order of $\mathbf{X}$. (ii) The matrixes $\mathbf{W}_{sc}$, $\mathbf{W}_{sc}^T$, $\mathbf{W}_{ct}$, $\mathbf{W}_{ct}^T$, $\mathbf{W}_{st}$ and $\mathbf{W}_{st}^T$ are normalized by columns, and denoted by $\bar{\mathbf{W}}_{sc}$, $\bar{\mathbf{W}}_{sc}^T$, $\bar{\mathbf{W}}_{ct}$, $\bar{\mathbf{W}}_{ct}^T$, $\bar{\mathbf{W}}_{st}$ and $\bar{\mathbf{W}}_{st}^T$. (iii) We force $\sum_{i=1}^3 \alpha_i = \sum_{i=1}^3 \beta_i = \sum_{i=1}^3 \gamma_i = 1$. To alleviate the effect of matrix $\bar{\mathbf{X}}$, we set α_1, β_2 and γ_3 be a small number.

It can be proved that $\bar{\mathbf{M}} = \begin{bmatrix} \alpha_1\bar{\mathbf{W}}_{cc} & \beta_1\bar{\mathbf{W}}_{sc}^T & \gamma_1\bar{\mathbf{W}}_{ct} \\ \alpha_2\bar{\mathbf{W}}_{sc} & \beta_2\bar{\mathbf{W}}_{ss} & \gamma_2\bar{\mathbf{W}}_{st} \\ \alpha_3\bar{\mathbf{W}}_{ct}^T & \beta_3\bar{\mathbf{W}}_{st}^T & \gamma_3\bar{\mathbf{W}}_{tt} \end{bmatrix}$ is stochastic and irreducible. We use $\bar{\mathbf{M}}$ to replace $\mathbf{M}$, and our method will converge to a unique ranking vector.

6.3 Experimental evaluation

6.3.1 Multi-document summarization

The proposed method is evaluated by the multi-document summarization task (Task 2) of Document Understanding Conference (DUC) 2001 [105], and DUC 2005. The task of DUC 2001 is to generate a generic summary with approximately 100 words for the given document sets. A total of 304 documents in 30 sets were provided as system input in DUC 2001. There were 50 document clusters and 1593 documents in DUC 2005, and a summary of about 250 words are aimed to be generated for each document cluster. It is worth noting that the original task of DUC 2005 is query based document summarization, which means that each document cluster was accompanied with a query description representing the information needed. However, our summarization method is not a query based document summarization method. As such, in the following experiments, all the experimental results are generated without considering the query information provided by DUC 2005. In our experiments, all the documents in the same document set are combined into one large document which is divided into individual sentences and the words in the Rijsbergen's stop words list [106] are removed to construct the document matrix. The word stemming operation is performed using Porter's stemming algorithm [107].

We use the well-known ROUGE system [108] which includes five methods, i.e., ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S and ROUGE-SU, to evaluate our system performance. ROUGE-N is computed as follows by using n-gram recall between the candidate summary and the reference sum-

maries

$$ROUGE - N = \frac{\sum_{s \in \{S_R\}} \sum_{n \in s} N_m(n)}{\sum_{s \in \{S_R\}} \sum_{n \in s} N(n)} \tag{6.22}$$

where the sentence set S_R denotes the reference summaries and s is a sentence in the set, n denotes n-gram and $N_m(n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries. The term $N(n)$ is the number of n-grams in the set of reference summaries. The length of longest common subsequence (LCS) between the candidate summary and reference summaries is employed by ROUGE-L, and ROUGE-W uses the weighted LCS as shown in (6.23) [108].

$$\begin{aligned} R_{lcs} &= \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i,C)}{m} \\ P_{lcs} &= \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i,C)}{n} \\ F_{lcs} &= \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}} \end{aligned} \tag{6.23}$$

where C is the candidate summary which contains n words and $r_i, i = 1, \ldots, u$ are sentences in a reference summary containing m words. $LCS_{\cup}(r_i, C)$ is the LCS score of the union longest common subsequence between reference sentence r_i and candidate summary C. ROUGE-W is an extension of ROUGE-L that assigns high weight to longer consecutive matches between a candidate summary and a reference summary. ROUGE-S calculates the skip-bigram cooccurrence in the reference and candidate summaries [108].

$$\begin{aligned} R_{skip2} &= \frac{SKIP2(X,Y)}{C(m,2)} \\ P_{skip2} &= \frac{SKIP2(X,Y)}{C(n,2)} \\ F_{skip2} &= \frac{(1+\beta^2)R_{skip2}P_{skip2}}{R_{skip2}+\beta^2 P_{skip2}} \end{aligned} \tag{6.24}$$

where $SKIP2(X, Y)$ is the number of skip-bigram matches between X and

Table 6.1: A comparison of results on DUC-2001.

System	ROUGE-1	ROUGE-2	ROUGE-L
MREG	**0.31972**	**0.06802**	**0.28120**
PR	0.30955	0.05411	0.27792
NMF	0.30287	0.06002	0.26839
LSA	0.27182	0.03676	0.24718

Y, β controlling the relative importance of P_{skip2} and R_{skip2}, and C is the combination function. ROUGE-SU extends ROUGE-S with the addition of unigram as counting unit, and ROUGE-SU4 means that the measure calculates n-grams with maximum skip-distance of 4. We use the ROUGE toolkit[1] 1.5.5 to perform the evaluation and here we report the average F-measure scores of ROUGE-1, ROUGE-2 and ROUGE-L. In our evaluation, both model and peer summaries are stemmed using Porter stemmer and superfluous words are pruned.

6.3.1.1 Performance comparison

The performance of MREG method on DUC-2001 is compared with the Latent Semantic Analysis (LSA) [109], PageRank algorithm (PR) [110], and Non-negative Matrix Factorization (NMF) [111]. The ROUGE scores obtained by these methods are shown in Table 6.1. It is observed in the table that MREG algorithm generates better ROUGE scores than the other methods.

The comparison of the MREG method with the best original participants of DUC-2001 is shown in Table 6.2[2]. The results of systems in DUC-2001

[1]http://berouge.com/default.aspx

[2]The list of participants is available from http://www-nlpir.nist.gov/projects/duc/pubs/2001slides/pauls_slides/duc2001.ppt.

Table 6.2: Performance comparison with the original participants of DUC-2001.

System	ROUGE-1	ROUGE-2	ROUGE-L
T	0.32540	0.07870	0.29709
MREG	**0.31972**	**0.06802**	**0.28120**
P	0.31304	0.06288	0.28048
V	0.29849	0.06078	0.26366
O	0.29528	0.05008	0.26888

Table 6.3: A comparison of results on DUC-2005.

System	ROUGE-1	ROUGE-2	ROUGE-SU4
MREG	0.3519	**0.0652**	**0.1202**
MRC [93]	**0.3633**	0.0632	0.1201
NMF	0.3231	0.0494	0.1021
LSA	0.3077	0.0378	0.0928

are calculated using the result data of submissions downloaded from DUC website[3]. It is revealed by the table that the MREG method generates comparable results with the top original participants in all the ROUGE measures.

The performance of MREG method on DUC-2005 is compared with the Mutual Reinforcement Principle (MRC) [93], Non-negative Matrix Factorization (NMF) [111], and Latent Semantic Analysis (LSA) [109] methods. The ROUGE scores of these methods are shown in Table 6.3. It is observed in the table that MREG algorithm generates competitive ROUGE scores with the MRC method, and better results than NMF and LSA methods.

[3]http://www-nlpir.nist.gov/projects/duc/data.html

6.3.1.2 Discussion of selective parameters

In our experiments, two kinds of parameters affect the system performance: the number of clusters k and the balance parameters α, β and γ in the ranking algorithm. The number of clusters $k = \lfloor n/f \rfloor$, where n is the number of sentences and f is an integer. The influences of these parameters are shown in the following figures.

Figures 6.3 and 6.4 demonstrate the ROUGE-1 and ROUGE-2 values obtained by KM and agglomerative (AGG) clustering algorithms with different values of f. It can be seen from Fig. 6.3 that the MREG algorithm which is named KM-TSC-EM in the figure almost always outperforms the baseline system which only calculates the mutual-reinforcement between sentences and clusters (KM-SC in the figure). The abbreviation TSC is short for Term, Sentence and Cluster, which means the algorithm is based on the mutual effects between these three items. EM is short for Embedded Graph based sentence clustering. The performance of KM-SC method fluctuates dramatically according to the number of clusters. The KM-SC-EM and KM-TSC-EM methods are more robust than the KM-SC method with respect to different cluster numbers. By comparing the curves of KM-TSC and KM-TSC-EM, we can see that the system performance is improved by using the sentence clusters based on embedded graph. Figure 6.4 indicates that the AGG-TSC-EM method also outperforms the AGG-SC method by using agglomerative clustering algorithm for different cluster numbers. To illuminate the robustness of MREG algorithm, the means and variances of the performances of different component combination are also compared in Tables 6.4 and 6.5. As shown in the table, both KM-TSC-EM and AGG-TSC-EM algo-

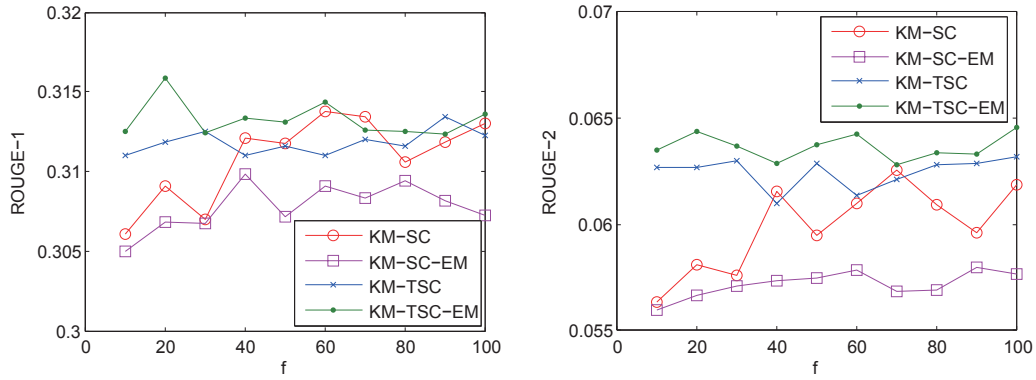


Figure 6.3: ROUGE scores obtained by K-means clustering algorithm under different values of f.

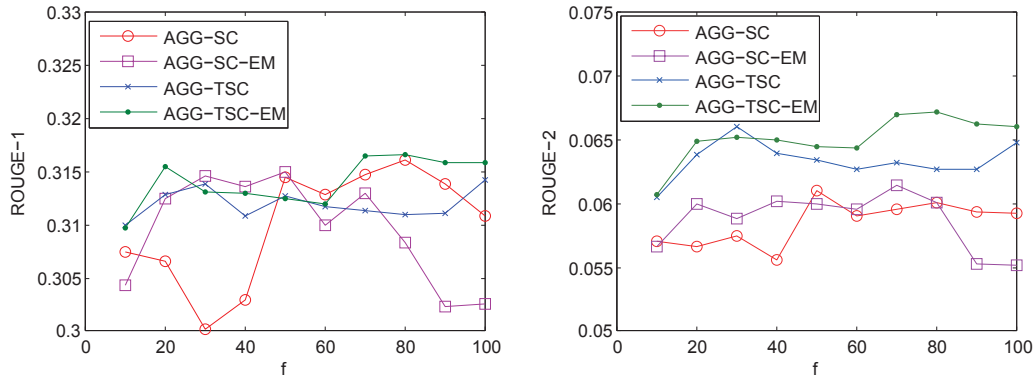


Figure 6.4: ROUGE scores obtained by agglomerative clustering algorithm under different values of f.

rithms improve the system performance slightly, and they reduce the variance to $16\% - 18\%$ of the original SC algorithm.

Figure 6.5 illustrates the influence to the performance of balance parameters α, β and γ when the cluster number is set to be $\lfloor n/100 \rfloor$, where n is the sentence number. In our experiments, we added the following restriction to parameters α_i, β_i and γ_i, $i = 1, 2$ in (6.17): $\alpha_1 = \alpha_2 = \alpha$, $\beta_1 = \beta_2 = \beta$, and $\gamma_1 = \gamma_2 = \gamma$ to make the weights of sentence-cluster affinity, sentence-term affinity, and cluster-term affinity consistent in the algorithm. All these

Table 6.4: Mean value of system performance with different component combinations using KM clustering algorithm.

Methods	Mean Value			
	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-W
KM-SC	0.3109	0.0599	0.2780	0.1329
KM-SC+EM	0.3078	0.0572	0.2758	0.1316
KM-TSC	0.3118	0.0625	0.2753	0.1329
KM-TSC+EM	0.3133	0.0637	0.2770	0.1335
AGG-SC	0.3100	0.0585	0.2756	0.1318
AGG-SC+EM	0.3096	0.0587	0.2787	0.1330
AGG-TSC	0.3120	0.0634	0.2742	0.1324
AGG-TSC+EM	0.3141	0.0651	0.2774	0.1337

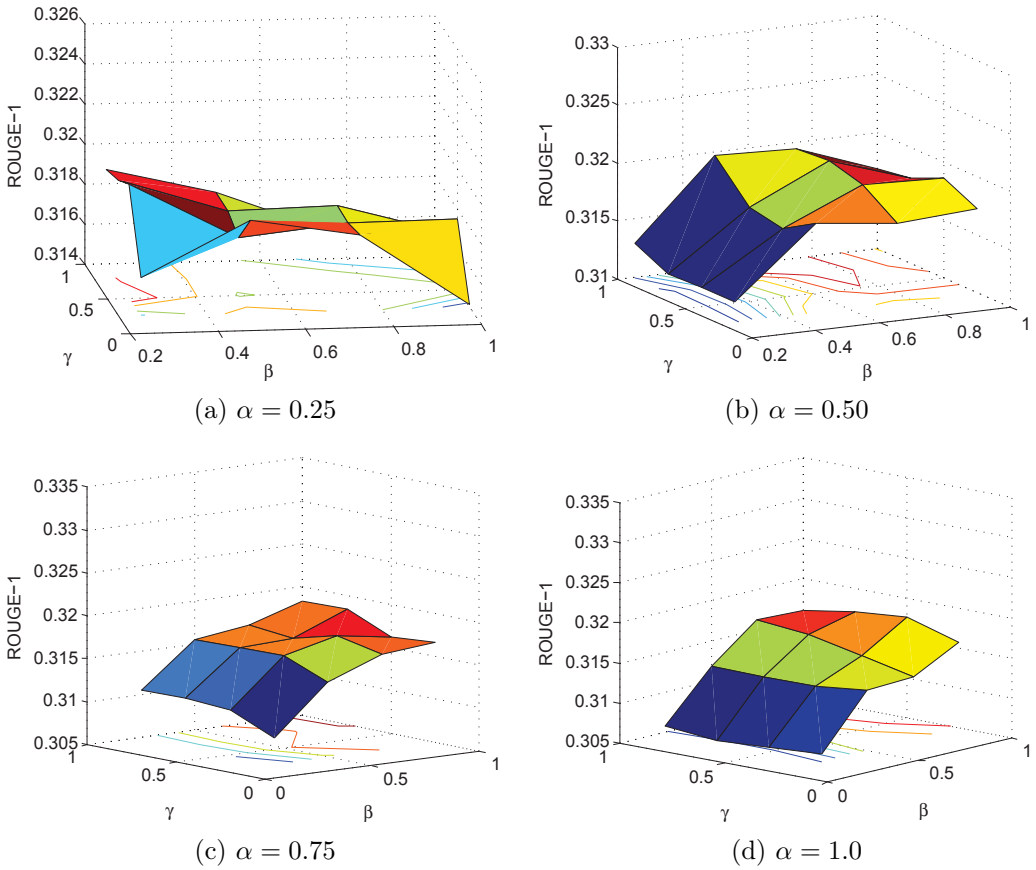
Table 6.5: Variance of system performance with different component combinations using KM clustering algorithm.

Methods	Variance($1e-4$)			
	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-W
KM-SC	0.1598	0.0769	0.1529	0.0429
KM-SC+EM	0.0770	0.0311	0.0642	0.0149
KM-TSC	0.0214	0.0227	0.0206	0.0078
KM-TSC+EM	0.0262	0.0096	0.0303	0.0056
AGG-SC	0.2977	0.0299	0.4336	0.0755
AGG-SC+EM	0.2465	0.0492	0.2273	0.0540
AGG-TSC	0.0191	0.0218	0.0152	0.0033
AGG-TSC+EM	0.0549	0.0339	0.0374	0.0093

parameters are set to be one of the following values: $0.25, 0.50, 0.75$, and 1.0 in our experiment. It can be seen from Fig. 6.5a that when $\alpha = 0.25$, the best performance is obtained with $\beta = 0.25$ and $\gamma = 0.50$. The same pattern can be found from Fig. 6.5b. The best performance is obtained at the point $\alpha = 0.50, \beta = 0.50, \gamma = 1.0$ where the ROUGE-1 score is 0.31972. In Figs. 6.5c and 6.5d, the best ROUGE-1 scores 0.31841 and 0.31769 are obtained at $\alpha = 0.750, \beta = 1.0, \gamma = 1.0$ and $\alpha = 0.750, \beta = 1.0, \gamma = 0.50$. We can predict that these two points are local optimal points. The result illuminates that the mutual effects between sentences and clusters is as important as the effects of sentences and terms to system performance. While the weight of mutual effects between clusters and terms should be greater than the other weights. In our experiment, the best result can be obtained when $\alpha : \beta : \gamma = 1 : 1 : 2$ and $f = 100$.

6.4 Summary

In this chapter, a mutual-reinforcement algorithm has been presented to rank nodes in a graph by calculating the mutual effects between nodes, features and clusters. The method has been evaluated by the tasks of multi-document summarization and emotion identification. The experimental results on DUC-2001 and DUC-2005 data sets showed the effectiveness of the proposed approach. Performance comparison of different combinations of components illustrated that the algorithm could improve system performance and it is more robust with respect to different cluster numbers.



(a) $\alpha = 0.25$ (b) $\alpha = 0.50$

(c) $\alpha = 0.75$ (d) $\alpha = 1.0$

Figure 6.5: ROUGE scores under different values of balance parameters α, β and γ.

Chapter 7

Incremental Learning for Data Association

7.1 Introduction

An association describes the connection between objects, and the degree of an association is a measurement of the connection. In last two chapters, some associations are applied to identify emotional sentences and to summarize documents. One can simply treat the association degree same as the distance or similarity between objects. Many methods of measuring association have been proposed in the field. Cosine similarity between vectors and pointwise mutual information are most popular measures which have been used in semantic similarity estimation [112], document clustering [113], affect recognition in text [114], etc. However, to the best of our knowledge, the method of learning association on a growing data set has not been studied yet. The model describing associations between objects should be able to adapt to new data without losing information learned from the old data. For example, we could obtain the semantic similarities between a set of words based on a document set. If a new document set is collected, the model should be able to learn new knowledge from the new data set, and make the

similarities closer to the real ones. A straightforward solution to this problem is that we combine the old and new data sets into one, and train the model again. It will be a waste of time to train the model on the old data again. In some cases, we even do not have the old data at all.

In this chapter, incremental learning is proposed for calculating associations between objects in a growing data set. Given a data set, new objects may be added, and the associations should be updated accordingly. Suppose that the association degrees between objects have been trained on an old data set, we will first learn the association degrees on the new data set, and then update the association degrees using the weighted summation of the old and new values. With this method, the associations can be updated without re-training the model on the old data set. Some associations may not be discovered because the whole data set is divided into two subsets. A self-upgrading method is proposed to find the hidden association information. A matrix called association matrix (AM) is constructed for a data set that each item denotes the association degrees between objects. Row vectors of the AM are taken as features of the objects, and the similarities between row vectors are considered as the similarities of objects. These values are added to the original association degrees to upgrade the AM. The proposed methods are applied on word similarity calculation and social network link prediction in our experiments. The results show that the proposed incremental learning method is able to generate competitive results with the method of re-training the model. The self-upgrading method is able to detect hidden association information in an AM. The contributions of this chapter are highlighted as follows:

(i) An incremental learning algorithm is proposed which makes an association model be able to adapt to new data;

(ii) A method of upgrading association degrees between objects is presented to find hidden association information; and

(iii) The properties of the proposed incremental learning algorithm are discussed by applying it on some specific association models.

The rest of this chapter is organized as follows: The proposed incremental learning and self-upgrading methods are presented in Section 7.2. Our method is evaluated by conducting experiments of calculating word similarity and predicting links in social networks in Section 7.3. We conclude our work in Section 7.4.

7.2 Incremental learning for association

In this section, we first introduce some different data association measurement methods. A general incremental learning method for learning data association on a growing data set is described, and the properties of the method applying on some specific association models are discussed. A self-upgrading method, which could discover hidden information stored in an AM, will be presented at last.

7.2.1 Data association

An association describes a connection between objects and an association degree measures the connection. We category the associations into two classes:

countable association and probabilistic association. The definitions of these two types of association are given respectively:

(i) Countable association: The countable association refers to the relationships between objects that can be counted using integers.

(ii) Probabilistic association: The probabilistic association is defined as the reflectance of the connection between objects in the probabilities manner.

Examples of countable association include items co-occurring in a shopping list, co-authors of a paper, etc. The countable association emphasizes on the existence of the connection between objects. The relationship between a word and a news topic, the probability of an object appears in a scene, etc. are typical examples of the probabilistic association. The probabilistic association emphasizes on the degree of the connections between objects. One can see that the countable association could be taken as a special type of probabilistic association that the degree can only be non-negative integers. In the following sections, we use the term *association* to present both *countable association* and *probabilistic association*.

Suppose there are N objects in a whole space, we could use a matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ so-called Association Matrix to store the relationship between these objects, where A_{ij} is the association degree of object i and j. There are many methods of measuring the degree of associations between objects. For countable association, the common neighbors [46] method is a widely used method of calculating scores between two objects

$$\mathrm{S(a,b)} = \mid \Gamma(\mathrm{a}) \cap \Gamma(\mathrm{b}) \mid \tag{7.1}$$

where $\Gamma(a)$ and $\Gamma(b)$ are the neighbor sets of objects a and b respectively. Many extended methods have been proposed based on this method [115]. For the probabilistic association, similarity is a widely used concept for measuring association between objects [53]. Comparing to the common neighbors, similarity method not only considers the commonality between objects, but also describes the differences between them. Cosine similarity defined by (5.10), which may be the most conventional similarity method, has been applied in different machine learning tasks like document retrieval and face recognition. Pointwise Mutual Information (PMI) normalizes the probability of co-occurrence of the two objects with their individual probabilities of co-occurrence [116]. Given events a and b, the degree of the association is determined by

$$S(a, b) = \log_2 \frac{\Pr(\mathsf{a}, \mathsf{b})}{\Pr(\mathsf{a}) \Pr(\mathsf{b})} \tag{7.2}$$

If $\frac{\Pr(\mathsf{a},\mathsf{b})}{\Pr(\mathsf{a}) \Pr(\mathsf{b})} > 1$, it is obvious that $0 < S(a, b) \leq 1 - \log \min\{\Pr(\mathsf{a}), \Pr(\mathsf{b})\}$ where the maximum is achieved when an object is the subset of the other one.

7.2.2 Incremental learning

A data set for learning associations will grow if new data is collected. The model of learning associations should be adapt to the new data without losing old information. Take word similarity calculation as an example. Suppose a set of similarity values are obtained on a corpora using (7.2). If a new corpora is obtained, we should update the association values based on the new corpora to make the similarities more close to the "real" ones. It will consume more time if we train the model again on the combination of the old

and new corpora. A general method is proposed to make the model adapt to the new data.

Suppose we have an existing association matrix $\mathbf{A}_o$ where each item in it denotes the association degree of two objects. If a new set of training samples are collected, two operations may be needed to update $\mathbf{A}_o$: updating existing nodes and adding new nodes. Let $\mathbf{A}_n$ be the association matrix obtained on the new training set. The association degrees after adaption is set to be the weighted summation of old values and new ones. The proposed incremental learning method is shown in Algorithm 4.

Algorithm 4 Association evolution

Let $S = S_o \cup S_n$ be the whole object set, which is the union of old object set and the new one.
Let $\mathbf{A} \in \mathbb{R}^{N \times N}$ be the association matrix obtained finally, where N is the number of unique objects in S.
Extend $\mathbf{A}_n$ to $\tilde{\mathbf{A}}_n \in \mathbb{R}^{N \times N}$
for all nodes $s \in S_n$ and $s' \in S_n$ **do**
    $\tilde{\mathbf{A}}_n(s, s') = \mathbf{A}_n(s, s')$
end for
for all node $s \in S_o - S_o \cap S_n$ and $s' \in S$ **do**
    $\tilde{\mathbf{A}}_n(s, s') = 0$
    $\tilde{\mathbf{A}}_n(s', s) = 0$
end for
Extend $\mathbf{A}_o$ to $\tilde{\mathbf{A}}_o \in \mathbb{R}^{N \times N}$ similarly
Calculate $\mathbf{A}$

$$\mathbf{A} = w\tilde{\mathbf{A}}_n + (1 - w)\tilde{\mathbf{A}}_o \tag{7.3}$$

where $w \in (0, 1)$ is a weight for balancing the new and old associations.
Self-upgrading of $\mathbf{A}$.

In the algorithm, the whole object set is obtained by combining the two sets S_n and S_o. Then we extend the two association matrices to contain all the objects. For objects not existing in S_n but belonging to S_o, the association degree of these objects with others are set to be zero in the extended AM.

Similarly, the association degree between objects existing in S_n only and other objects are also set to be zero. Then the association degrees of all objects are added up.

This is a general method for almost all kinds of measurements of association. It is difficult to compare the AM $\mathbf{A}$ obtained by (7.3) and the AM $\mathbf{A}_s$ obtained by training on S. We analyze two specific association measurements given by (7.1) and (5.10). For the common neighbor method, every item in $\tilde{\mathbf{A}}_n$ and $\tilde{\mathbf{A}}_o$ is a non-negative integer. According to the Algorithm 4,

$$wa_n + (1-w)a_o \leq a_n + a_o \tag{7.4}$$

where a_n and a_o are items in $\mathbf{A}_n$ and $\mathbf{A}_o$ respectively. The weight $w \in (0,1)$. An item a_s in $\mathbf{A}_s$ will not be smaller than $a_n + a_o$ by definition. Hence, the value of association degree in $\mathbf{A}$ will not be greater than those in $\mathbf{A}_s$.

If the association degree is computed using cosine similarity (5.10), the value of the association degree between objects a and b in $\mathbf{A}_n$ will be

$$s_n(a,b) = \frac{\mathbf{a}_1 \cdot \mathbf{b}_1}{\sqrt{(\mathbf{a}_1 \cdot \mathbf{a}_1)(\mathbf{b}_1 \cdot \mathbf{b}_1)}} \tag{7.5}$$

where $\mathbf{a}_1$ and $\mathbf{b}_1$ are vectors presenting a and b respectively. Suppose $\mathbf{a}_1$ and $\mathbf{b}_1$ are normalized, equation (7.5) can be re-written as

$$s_n(a,b) = \mathbf{a}_1 \cdot \mathbf{b}_1 \tag{7.6}$$

Similarly, the association degree in $\mathbf{A}_o$ can be written as $s_o(a,b) = \mathbf{a}_2 \cdot \mathbf{b}_2$. Assume the vector of a in $\mathbf{A}_s$ is $\mathbf{a}_1 + \mathbf{a}_2$, and the vector of b is $\mathbf{b}_1 + \mathbf{b}_2$. The

value in $\mathbf{A}_s$ will be

$$\begin{aligned} s_s(a,b) &= \frac{(\mathbf{a}_1+\mathbf{a}_2)\cdot(\mathbf{b}_1+\mathbf{b}_2)}{\sqrt{((\mathbf{a}_1+\mathbf{a}_2)\cdot(\mathbf{a}_1+\mathbf{a}_2))((\mathbf{b}_1+\mathbf{b}_2)\cdot(\mathbf{b}_1+\mathbf{b}_2))}} \\ &= \frac{(\mathbf{a}_1+\mathbf{a}_2)\cdot(\mathbf{b}_1+\mathbf{b}_2)}{\|\mathbf{a}_1+\mathbf{a}_2\|\|\mathbf{b}_1+\mathbf{b}_2\|} \end{aligned} \tag{7.7}$$

As $\| \mathbf{a}_1 \| = 1$, $\| \mathbf{a}_2 \| = 1$, and $\| \mathbf{a}_1 + \mathbf{a}_2 \| \leq \| \mathbf{a}_1 \| + \| \mathbf{a}_2 \| = 2$,

$$s_s(a,b) = \frac{(\mathbf{a}_1 + \mathbf{a}_2) \cdot (\mathbf{b}_1 + \mathbf{b}_2)}{\| \mathbf{a}_1 + \mathbf{a}_2 \| \| \mathbf{b}_1 + \mathbf{b}_2 \|} \geq 0.25(\mathbf{a}_1\cdot\mathbf{b}_1+\mathbf{a}_2\cdot\mathbf{b}_1+\mathbf{a}_1\cdot\mathbf{b}_2+\mathbf{a}_2\cdot\mathbf{b}_2) \tag{7.8}$$

The value in $\mathbf{A}$ will be

$$s(a,b) = w\mathbf{a}_1 \cdot \mathbf{b}_1 + (1-w)\mathbf{a}_2 \cdot \mathbf{b}_2 \tag{7.9}$$

We have

$$s_s(a,b) - s(a,b) \geq (0.25-w)\mathbf{a}_1\cdot\mathbf{b}_1 + (w-0.75)\mathbf{a}_2\cdot\mathbf{b}_2 + 0.25\mathbf{a}_2\cdot\mathbf{b}_1 + 0.25\mathbf{a}_1\cdot\mathbf{b}_2 \tag{7.10}$$

If $w = 0.5$,

$$s_s(a,b) - s(a,b) \geq 0.25(-\mathbf{a}_1 + \mathbf{a}_2) \cdot (\mathbf{b}_1 - \mathbf{b}_2) \tag{7.11}$$

One can see that the value of (7.11) depends on the vectors representing objects a and b obtained on the two data sets. If $\mathbf{a}_1 \approx \mathbf{a}_2$ and $\mathbf{b}_1 \approx \mathbf{b}_2$, we get $s(a,b) \leq s_s(a,b)$, which is same as (7.4). If the formula holds, we can deduce that some information will be lost using the incremental learning method.

The association evolution algorithm will reduce the time complexity comparing to re-training the model on the whole data set S. Some additional time is consumed by the matrix summation operation. We will show the

time consuming of the method in Section 7.3 using some examples.

7.2.3 Self-upgrading of an AM

It is possible that the association between two objects are not obtained by the incremental learning method. An example is shown in Figure 7.1. The old network S_o is composed by three notes 1, 2, and 3. The new data set is composed by notes 1, 4, and 5. Using the common neighbors method (7.1), we can obtain the extended association matrices for the two networks

$$\tilde{\mathbf{A}}_o = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$\tilde{\mathbf{A}}_n = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

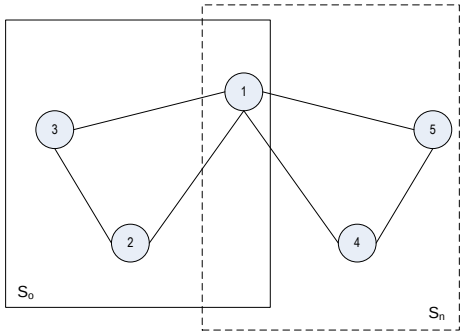


Figure 7.1: An example of a social network.

According to Algorithm 4, the weighted summation of $\tilde{\mathbf{A}}_o$ and $\tilde{\mathbf{A}}_n$ is

$$\mathbf{A} = \begin{bmatrix} 0 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0.5 & 0 \end{bmatrix}$$

where $w = 0.5$ in (7.3). One can find that the association degree of note 2 and 4 is 0. However, actually they have a common neighbor 1. To find this kind of hidden information, we could check the objects that associated with both of these two objects. Like in the real world, two people who have many common friends may be friends of each other.

Self-upgrading (SU) of an AM is based on this intuitive idea. The ith row vector in an AM stores the association degree of object i with all the other objects. This row vector could be taken as a feature of object i. If the row vectors of two objects are very similar to each other, we believe that the association degree between this two objects should be high. We calculate the similarities between row vectors of an AM and upgrade the values in it. The

rule of upgrading for each item in the association matrix A is set as follows:

$$\tilde{A}_{ij} = \alpha A_{ij} + (1-\alpha) f(\mathbf{v}_i, \mathbf{v}_j) \tag{7.12}$$

where A_{ij} is the association degree stored in the original $\mathbf{A}$. The function $f(\mathbf{v}_i, \mathbf{v}_j)$ calculates the similarity of the two items i and j, where $\mathbf{v}_i$ and $\mathbf{v}_j$ are row vectors of i and j respectively. The parameter α is used to balance the weight of original distance and the new distance. We call this method self-upgrading because the information for upgrading is obtained inside an AM itself. The upgrading process can be written in the matrix format:

$$\mathbf{A}_{k+1} = \alpha \mathbf{A}_k + (1-\alpha)\mathbf{F}_k \tag{7.13}$$

where $\mathbf{F} \in \mathbb{R}^{N \times N}$and $F_{ij} = f(\mathbf{v}_i, \mathbf{v}_j)$. For each iteration k, we calculate a new $\mathbf{F}$ and add it to the association matrix.

Lemma 2. *Let $a = A_{ij}$ and $f = F_{ij}$, the value of a will be around f after some iterations.*

Proof. For each item a in A, if $a_k > f_k$, we have

$$a_{k+1} - a_k = (\alpha - 1)a_k + (1-\alpha)f_k = (1-\alpha)(f_k - a_k) < 0 \tag{7.14}$$

where k is the iteration number. Hence, if $a_k > f_k$, the value of a_k will become smaller in the upgrading process until $a_k \leq f_k$. Similarly, the value of a_k will increase if $a_k < f_k$. Hence, we can deduce that after some iterations, the value of a_k will be around f_k. □

Theorem 3. *For each item a in $\mathbf{A}$, $a_k \in [0,1]$ when $k \to \infty$ if $f_k \in [0,1]$.*

Proof. Given $f_k \leq 1$, if $a_k > 1$, we have $a_k > f_k$. Hence $a_{k+1} < a_k$ according to Lemma 2. If $a_{k+1} > 1 \geq f_{k+1}$, then $a_{k+2} < a_{k+1}$ until $a_{k+n} \leq 1$. If $a_{k+n} \leq 1$, we have

$$a_{k+n+1} = \alpha a_{k+n} + (1-\alpha) f_{k+n} \leq 1 \tag{7.15}$$

Similarly, we can prove that $a_k \geq 0$ when $k \to \infty$. □

7.3 Experimental results

In this section, experiments are conducted to evaluate the efficiency of the proposed methods. In the first experiment, we apply the proposed methods on calculating word similarities using Wikipedia and WordNet. In the second one, we predict links in a social network.

7.3.1 Word similarity calculation

Semantic similarities between words are basic knowledge for understanding natural language. Taking Wikipedia as a resource to solve natural language processing problems is a popular method in the field [56, 117]. In this experiment, we will learn the similarities of words using a subset of Wikipedia, and then learn the similarities again using another subset. The incremental learning method is applied to combine the results obtained on the two subsets. The self-upgrading method is applied based on the combined results to improve the system performance.

Term frequency–inverse document frequency (TF–IDF) is employed to calculate the relationship between a word and a document. The raw fre-

quency $f(t,d)$, which is the frequency of word t appears in a document d, is taken as the term frequency $tf(t,d)$. Inverse document frequency is defined as

$$idf(t,D) = \log \frac{\mid D \mid}{\mid \{d \in D : t \in d\} \mid} \tag{7.16}$$

where $\mid D \mid$ is the number of documents in the corpus, and $\mid \{d \in D : t \in d\} \mid$ is the number of documents that contain the term t. For a corpus containing $\mid D \mid$ documents, a matrix with dimension $N \times \mid D \mid$ is generated that each row is the TF-IDF vector of a word, and N is the number of terms. The cosine similarities of each pair of the row vectors are calculated and taken as the similarity between words. The WordSimilarity-353 collection (WS353) [118] is taken as a standard data set to evaluate the algorithms. In the WS353 collection, a set of 353 word pairs are collected and the related score of each word pair is given by 13-16 human judgments. In WS353 corpus, there are a total of 437 different words. We will calculate all the similarities between these words, and the similarities of 353 pairs of words in WS353 are selected to be compared with results given by human judgments. Spearman rank-order correlation coefficient is employed as the criteria. We first calculate the similarities between words in WS353 using the first $10,000$ documents (1st Set) in Wikipedia[1]. A Spearman rank-order correlation will be obtained by comparing the results and the similarities given by human. The second $10,000$ documents (2nd Set) in Wikipedia are selected to calculate the similarities of words in WS353. With the new similarities obtained, the incremental learning algorithm is applied to update the old similarity data. The weight w in (7.3) is set to be 0.5. We also learned the word sim-

[1]http://en.wikipedia.org/wiki/Wikipedia:Database_download

Table 7.1: The results of computing word similarity using the adaption model.

	1st set	2nd set	1st+2nd set	Incremental	SU
Correlation	0.340	0.349	0.384	0.395	0.419
Time for TF-IDF (s)	323.3	347.4	662.9	0	0
Time for Compare(s)	1.6	1.5	3.0	0.004	1.3

ilarities on the combination of the first and the second document sets. The self-upgrading algorithm is also applied on the updated similarity matrix. The results are shown in Table 7.1.

From the table, one can see that similar results were obtained on the first and second document sets. The system performance will have a 3% improvement if we learn the similarities using the combination of the two sets. The result obtained by the incremental learning method is 1% better than that obtained on the combination set. With the Self-upgrading method, the system performance will have another 2% improvement.

The time consuming of all the methods is also shown in Table 7.1. We can find that about 300 seconds are consumed by learning associations on the first and second subsets. For the combination of the two subsets, the time for computing TF-IDF and similarities almost equals to the summation of the times used by the separated data sets. If one applies the incremental learning method, no additional time is needed to re-calculate the TF-IDF. It costs 0.004 second to compute the similarity because only some matrix summation is needed. Another 1.3 seconds are consumed by the self-upgrading method. It is worth noting that we run five iterations for the SU method. The 1.3 seconds is the summation of the time used by these five iterations. The best result is obtained at the second iteration. The method of calculating distances between vectors of the association matrix is *euclidean*.

7.3.1.1 Self-upgrading in word similarity calculation

To demonstrate the efficient of the self-upgrading algorithm, we conduct two more experiments of calculating word similarities. Here we employ the Lin method and JCn method proposed in [53] and [54], which compute the similarities between words using WordNet [52]. A tool written in JAVA language was employed to implement the two methods[2] . The data set WS353 is used to evaluate the algorithms. We calculate the similarities of all the word pairs in the data set to obtain an association matrix. Then the Self-Upgrading method is applied to this matrix. The word similarities in the new matrix are taken to compare with those int the WS353 collection. We applied two methods of calculating distances between row vectors of a association matrix: cosine similarity and euclidean distance. Given two vectors $\mathbf{a}$ and $\mathbf{b}$, the cosine distance between them is calculated using (5.10). The euclidean distance is defined as

$$f(a,b) = \sqrt{(\mathbf{a}-\mathbf{b})(\mathbf{a}-\mathbf{b})'} \tag{7.17}$$

The results are shown in Table 7.2. From the table, we can see that the Lin method with SU could obtain a 2% improvement than the original method. The JCn method with SU achieved a 8% improvement, which proved the efficiency of the method. Figure 7.2 shows the system performance of the SU method with different distance calculation metric. From the figure, we can see that Lin method with euclidean Self-Upgrading and JCn method with cosine self-upgrading could improve the system performance. Both of these methods achieved best performance at the second iteration. At the first

[2]Pure Java WordNet Similarity Library at http://nlp.shef.ac.uk/result/software.html

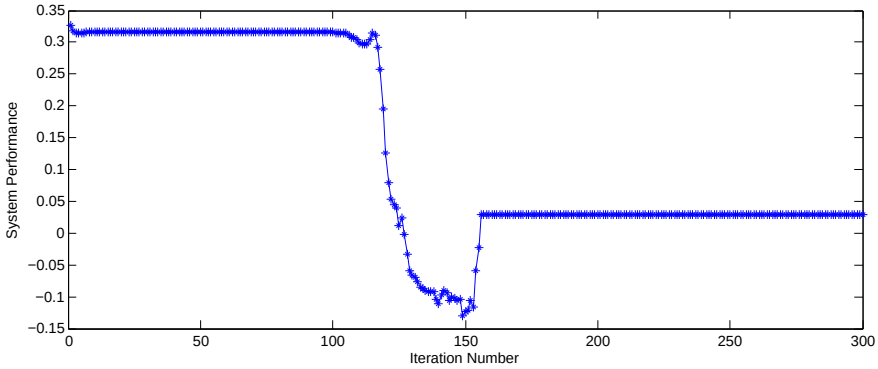
Table 7.2: Word relatedness.

Method	Lin	JCn
Correlation	0.33	0.30
Correlation with Self-Upgrading	0.35	0.38

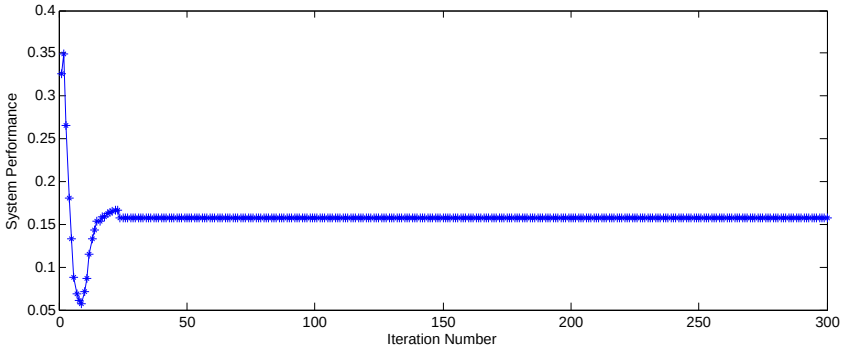
iteration, the weight of similarity obtained by the SU method is very low, and the effect is not obvious. After the second iteration, the performance starts decreasing as the weight becomes very high, and it forbids the original information stored in the association matrix. As shown in Figures 7.2a and 7.2d, these two methods didn't obtain better performance than the method without SU, which indicates that the information added by SU is not helpful for the task. One can also find that all the four methods will converge after several iterations. The euclidean SU method converges faster than the cosine SU method.

7.3.2 Link recommendation in a social network

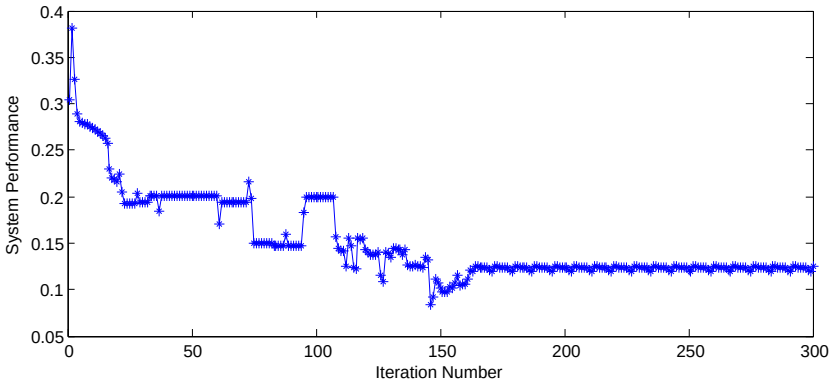
Digital Bibliography Project (DBLP) is a bibliography of computer science. The information like titles, authors, published year, etc. of numerous papers is stored in DBLP. The co-authorship in the data set is taken as the links between authors and studied in this experiment. We obtained all the papers published in the journals with name containing "IEEE" between 1998 and 2003. The co-authorship in these papers is the ground truth of our experiment. We select the papers published in years 1999 and 2000 as our first training set. Papers in 2001 and 2002 are the second training set. A total of about $370,000$ authors are obtained. We randomly selected five small groups of authors to test our algorithm. Each group contains 300 authors. Two authors are connected if they are co-authors of a paper. The common



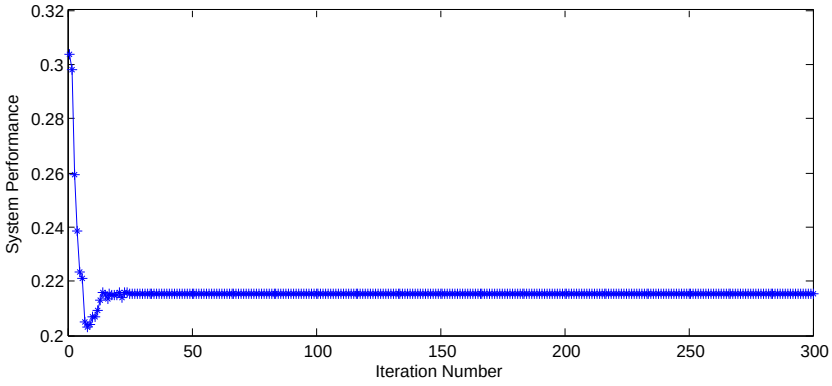
(a) Lin method with cosine self-upgrading.



(b) Lin method with euclidean self-upgrading.



(c) Jcn method with cosine self-upgrading.



(d) Jcn method with euclidean self-upgrading.

Figure 7.2: System performance of Lin and JCn methods with different iteration numbers.

Table 7.3: Definitions of TP, FP, FN and TN.

	Actual Co-Authors	
Predicted Co-Authors	TP	FP
	FN	TN

neighbors [46] method is used to calculate the association between authors. Our task is to predict the co-authorship in the whole data set using the two training sets. Precision, recall, and F1 are used to evaluate our algorithm. The definitions of TP, FP, FN, and TN are shown in Table 7.3. We first obtain two association matrices on the two trainings sets, and then apply the incremental learning method to generate a new association matrix. If the value of a_{ij} is bigger than zero, we will predict that there is a link between the authors i and j.

The results are shown in Table 7.4. One can see that the incremental method was able to generate better results than learning on single training sets. However, the results are worse than those obtained by training on both data sets, which proved the deduction in Section 7.2.2. After applying SU method, the F1 score was increased, which means some hidden information has been detected. By looking deeper at the results, one can find that the Recall value increased while the Precision value decreased. This can be explained that some true links are found by the SU methods, while some false links are also predicted at the same time. For example, a famous professor may co-author many papers with a lot of different people. Then all people in this set have a common co-author, and the cosine similarity between their vectors will not be zero. In our method, they are predicted as co-authors. However, they may just know each other, and never write papers together. The results are useful for recommending friends in a social network.

Table 7.4: The results of co-author prediction.

Group	Criteria	1st	2nd	1st+2nd	Incremental	SU
1	Precision	0.247	0.440	0.583	0.577	0.567
	Recall	0.170	0.321	0.498	0.459	0.495
	F1	0.190	0.357	0.527	0.497	0.518
2	Precision	0.220	0.643	0.817	0.813	0.808
	Recall	0.179	0.559	0.764	0.727	0.770
	F1	0.189	0.583	0.779	0.752	0.781
3	Precision	0.280	0.387	0.607	0.600	0.594
	Recall	0.226	0.339	0.562	0.540	0.553
	F1	0.239	0.355	0.576	0.558	0.567
4	Precision	0.243	0.465	0.641	0.628	0.603
	Recall	0.176	0.352	0.563	0.515	0.550
	F1	0.190	0.386	0.589	0.551	0.563
5	Precision	0.283	0.300	0.553	0.537	0.530
	Recall	0.202	0.234	0.493	0.431	0.477
	F1	0.223	0.254	0.512	0.464	0.493

7.4 Summary

In this chapter, an incremental learning algorithm for data association has been proposed to save the re-training time on a growing data set. A self-upgrading method has been presented to dig the hidden information in an association matrix. Experiments on word similarity estimation and link prediction in social network proved the effectiveness of the proposed method.

Time consuming is a problem of the SU method. We have to compute $N(N-1)/2$ times to obtain the distances of each pair of points. If N is a large number, the SU method may even consume more time and space than learning on the combined data set. A more efficient algorithm is needed in the future study.

Chapter 8

Conclusion

8.1 Imbalanced pattern classification

The thesis has investigated technologies of data analysis for emotional sentence identification. The problem has been formulated as an imbalanced pattern classification problem in Chapter 2 and Chapter 3 as there are much more neutral sentences than the emotional sentences in an article. A LDA based classifier has been proposed and a SIMPLS based classifier has been employed to solve the imbalanced pattern classification problem. The experimental results on the UIUC children' corpus showed that better accuracy of emotional sentences could be obtained using the algorithms without losing accuracy of neutral sentences. The unweighted accuracy could achieve about 66% on the corpus with quite simple features like special punctuations, number of subjective words, etc.

A system that fuses multiple classifiers has been constructed to improve the performance of emotion identification in Chapter 4. The final prediction of a new sample is set to be the weighted summation of the scores obtained by different classifiers. Experimental results demonstrated that the fusion system was bale to obtain better F1 value than those obtained by single classifiers. The experimental results also showed that simply appending features

may even make system performance worse for some classifiers. Classifier fusion or feature extraction technologies may solve this problem.

8.2 Data association

The association between words, bigrams and sentences have been employed to calculate the emotion score in a sentence in Chapter 5 using a mutual reinforcement learning algorithm. The sentences with higher emotional scores are taken as emotional sentences. Experimental results showed that the method was able to detect emotional sentences using the associations. The bigram-term association was proved to be not important for the emotional sentence identification, which inspired us to explore deeper associations between them. Considering the association between terms, sentences, and clusters, the proposed method has been applied on document summarization tasks in Chapter 6. The experimental results demonstrated the efficiency of the method.

Incremental learning for data association has been discussed in Chapter 7 because the data association should be updated with the growth of a data set. A self-upgrading method has been presented to dig the hidden information in an association matrix. The method has been evaluated using word similarity calculation and link prediction in a social network. The experimental results showed that the self-upgrading method could generate competitive results with re-training the model while using less time.

8.3 Limitations and future work

We have explored the techniques of data analysis for emotion identification in text. Limitations of our work are discussed and the research topics are recommended for further investigation.

(i) The work in this thesis focuses on the classifiers and algorithms for emotional sentences identification, while the features extracted from an article have not been studied amply, which are very important to achieve a good system performance. More useful features will be discovered in the future work.

(ii) The proposed classifiers for imbalanced classification are all linear classifiers. They cannot solve problems like XOR problem in neural network [82]. The system performance of these classifiers will decrease if there are few testing samples because the distribution of testing samples are different from the distribution of training samples. In the future, more research is necessary to make the classifiers be more robust to the distribution of the data sets.

(iii) the method proposed in Chapter 5 employs data association to calculate emotional scores of sentences using a mutual reinforcement learning method. It is difficult to add new features to this method as it will consume much more time to calculate associations between new elements and existing elements. New methods more extensible should be investigated while they should keep the efficiency.

(iv) The proposed incremental learning method of data association is quite general that can be applied for most of the data association measure-

ments. In this case, the properties of the method are difficult to analyze mathematically. More work is needed to be done to prove the efficiency of the method on specific data association models.

Bibliography

[1] C. Alm, Affect in text and speech, Ph.D. thesis, University of Illinois at Urbana-Champaign (2008).

[2] R. Calix, S. Mallepudi, B. Chen, G. Knapp, Emotion recognition in text for 3-d facial expression rendering, IEEE Transactions on Multimedia 12 (6) (2010) 544 –551.

[3] D. Ghazi, D. Inkpen, S. Szpakowicz, Hierarchical versus flat classification of emotions in text, in: Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, Association for Computational Linguistics, 2010, pp. 140–146.

[4] H. He, E. Garcia, Learning from imbalanced data, IEEE Transactions on Knowledge and Data Engineering 21 (9) (2009) 1263–1284.

[5] B. Pang, L. Lee, Opinion mining and sentiment analysis, Foundations and Trends in Information Retrieval 2 (1-2) (2008) 1–135.

[6] C. Bartneck, Integrating the occ model of emotions in embodied characters, in: Workshop on Virtual Conversational Characters, Citeseer, 2002.

[7] S. Aman, S. Szpakowicz, Identifying expressions of emotion in text, in: Text, Speech and Dialogue, Springer, 2007, pp. 196–205.

[8] J. Wiebe, T. Wilson, C. Cardie, Annotating expressions of opinions and emotions in language, Language Resources and Evaluation 39 (2) (2005) 165–210.

[9] C. Strapparava, R. Mihalcea, Learning to identify emotions in text, in: Proceedings of the ACM symposium on Applied computing, ACM, 2008, pp. 1556–1560.

[10] M. Marcus, M. Marcinkiewicz, B. Santorini, Building a large annotated corpus of english: The penn treebank, Computational linguistics 19 (2) (1993) 313–330.

[11] C. Alm, D. Roth, R. Sproat, Emotions from text: machine learning for text-based emotion prediction, in: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2005, pp. 579–586.

[12] C. Strapparava, R. Mihalcea, Semeval-2007 task 14: affective text, in: Proceedings of the 4th International Workshop on Semantic Evaluations, Association for Computational Linguistics, 2007, pp. 70–74.

[13] P. Katz, M. Singleton, R. Wicentowski, Swat-mp: the semeval-2007 systems for task 5 and task 14, in: Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), no. June, 2007, pp. 308–313.

[14] F. Chaumartin, Upar7: A knowledge-based system for headline sentiment tagging, in: Proceedings of the 4th International Workshop

on Semantic Evaluations, Association for Computational Linguistics, 2007, pp. 422–425.

[15] A. Balahur, J. M. Hermida, A. Montoyo, Building and exploiting emotinet, a knowledge base for emotion detection based on the appraisal theory model, IEEE Transactions on Affective Computing 3 (1) (2012) 88–101.

[16] M. Shaikh, H. Prendinger, I. Mitsuru, Assessing sentiment of text by semantic dependency and contextual valence analysis, Affective Computing and Intelligent Interaction (2007) 191–202.

[17] A. Ortony, G. Clore, A. Collins, The cognitive structure of emotions, Cambridge Univ Pr, 1990.

[18] M. Shaikh, H. Prendinger, M. Ishizuka, A linguistic interpretation of the occ emotion model for affect sensing from text, Affective information processing (2009) 45–73.

[19] C. Adam, A. Herzig, D. Longin, A logical formalization of the occ theory of emotions, Synthese 168 (2) (2009) 201–248.

[20] M. Wasikowski, X. Chen, Combating the small sample class imbalance problem using feature selection, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010) 1388–1400.

[21] M. Mazurowski, P. Habas, J. Zurada, J. Lo, J. Baker, G. Tourassi, Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance, Neural Networks 21 (2) (2008) 427–436.

[22] T. Nguyen, K. Chang, S. Hui, Supervised term weighting centroid-based classifiers for text categorization, Knowledge and Information Systems (2012) 1–25.

[23] N. Chawla, K. Bowyer, L. Hall, W. Kegelmeyer, Smote: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.

[24] S. Chen, H. He, E. Garcia, Ramoboost: Ranked minority oversampling in boosting, Neural Networks, IEEE Transactions on 21 (10) (2010) 1624–1642.

[25] E. Ramentol, Y. Caballero, R. Bello, F. Herrera, Smote-rsb*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory, Knowledge and Information Systems (2011) 1–21.

[26] S. Köknar-Tezel, L. Latecki, Improving svm classification on imbalanced time series data sets with ghost points, Knowledge and information systems 28 (1) (2011) 1–23.

[27] X. Liu, J. Wu, Z. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39 (2) (2009) 539–550.

[28] H. Zhao, Instance weighting versus threshold adjusting for cost-sensitive classification, Knowledge and Information Systems 15 (3) (2008) 321–334.

[29] Y. Sun, M. Kamel, A. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recognition 40 (12) (2007) 3358–3378.

[30] M. Kukar, I. Kononenko, Cost-sensitive learning with neural networks, in: Proceedings of the 13th European conference on artificial intelligence (ECAI-98), Citeseer, 1998, pp. 445–449.

[31] J. Kwok, Moderating the outputs of support vector machine classifiers, IEEE Transactions on Neural Networks 10 (5) (1999) 1018–1031.

[32] B. Wang, N. Japkowicz, Boosting support vector machines for imbalanced data sets, Knowledge and information systems 25 (1) (2010) 1–20.

[33] G. Wu, E. Chang, Kba: Kernel boundary alignment considering imbalanced data distribution, IEEE Transactions on Knowledge and Data Engineering 17 (6) (2005) 786–795.

[34] Y. Tang, Y.-Q. Zhang, Granular svm with repetitive undersampling for highly imbalanced protein homology prediction, in: IEEE International Conference on Granular Computing, IEEE, 2006, pp. 457–460.

[35] F. Vilariño, P. Spyridonos, J. Vitrià, P. Radeva, Experiments with svm and stratified sampling with an imbalanced problem: Detection of intestinal contractions, Pattern Recognition and Image Analysis (2005) 783–791.

[36] H. Qu, G. Li, W. Xu, An asymmetric classifier based on partial least squares, Pattern Recognition 43 (10) (2010) 3448–3457.

[37] S. S. Ge, H. He, C. Shen, Geometrically local embedding in manifolds for dimension reduction, Pattern Recognition 45 (4) (2012) 1455–1470.

[38] D. Huang, S. Ge, Z. Zhang, Speaker state classification based on fusion of asymmetric simpls and support vector machines, in: Twelfth Annual Conference of the International Speech Communication Association, 2011.

[39] G. Upton, I. Cook, Oxford dictionary of statistics (2002).

[40] M. Lesk, Word-word associations in document retrieval systems, American documentation 20 (1) (1969) 27–38.

[41] L. Doyle, Indexing and abstracting by association, American Documentation 13 (4) (1962) 378–390.

[42] H. Stiles, The association factor in information retrieval, Journal of the ACM (JACM) 8 (2) (1961) 271–279.

[43] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, in: ACM SIGMOD Record, Vol. 29, ACM, 2000, pp. 1–12.

[44] M. Zaki, Generating non-redundant association rules, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp. 34–43.

[45] Z. Zheng, R. Kohavi, L. Mason, Real world performance of association rule algorithms, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 401–406.

[46] M. Newman, Clustering and preferential attachment in growing networks, Physical Review E 64 (2) (2001) 025102.

[47] L. Adamic, E. Adar, Friends and neighbors on the web, Social networks 25 (3) (2003) 211–230.

[48] H. Song, T. Cho, V. Dave, Y. Zhang, L. Qiu, Scalable proximity estimation and link prediction in online social networks, in: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, ACM, 2009, pp. 322–335.

[49] P. Resnik, Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language, Journal of Artificial Intelligence Research 11 (1999) 95–130.

[50] G. Varelas, E. Voutsakis, P. Raftopoulou, E. Petrakis, E. Milios, Semantic similarity methods in wordnet and their application to information retrieval on the web, in: Proceedings of the 7th annual ACM international workshop on Web information and data management, ACM, 2005, pp. 10–16.

[51] I. Kaur, A. Hornof, A comparison of lsa, wordnet and pmi-ir for predicting user click behavior, in: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 2005, pp. 51–60.

[52] C. Fellbaum, Wordnet, Theory and Applications of Ontology: Computer Applications (2010) 231–243.

[53] D. Lin, An information-theoretic definition of similarity, in: Proceedings of the 15th international conference on Machine Learning, Vol. 1, San Francisco, 1998, pp. 296–304.

[54] J. Jiang, D. Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, arXiv preprint cmp-lg/9709008.

[55] D. Bollegala, Y. Matsuo, M. Ishizuka, A relational model of semantic similarity between words using automatically extracted lexical pattern clusters from the web, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2, Association for Computational Linguistics, 2009, pp. 803–812.

[56] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using wikipedia-based explicit semantic analysis, in: Proceedings of the 20th international joint conference on Artifical intelligence, 2007, pp. 1606–1611.

[57] P.-N. Tan, et al., Introduction to data mining, Pearson Education India, 2007.

[58] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. Verkamo, et al., Fast discovery of association rules, Advances in knowledge discovery and data mining 12 (1996) 307–328.

[59] J. Pei, J. Han, R. Mao, et al., Closet: An efficient algorithm for mining frequent closed itemsets, in: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2000, pp. 21–30.

[60] G. Webb, Efficient search for association rules, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp. 99–107.

[61] H. Huang, X. Wu, R. Relue, Association analysis with one scan of databases, in: Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on, IEEE, 2002, pp. 629–632.

[62] T. Murata, S. Moriyasu, Link prediction of social networks based on weighted proximity measures, in: Web Intelligence, IEEE/WIC/ACM International Conference on, IEEE, 2007, pp. 85–88.

[63] M. Al Hasan, V. Chaoji, S. Salem, M. Zaki, Link prediction using supervised learning, in: SDM06: Workshop on Link Analysis, Counter-terrorism and Security, 2006.

[64] D. Cheung, J. Han, V. Ng, C. Wong, Maintenance of discovered association rules in large databases: An incremental updating technique, in: Data Engineering, 1996. Proceedings of the Twelfth International Conference on, IEEE, 1996, pp. 106–114.

[65] D. Cheung, S. Lee, B. Kao, et al., A general incremental technique for maintaining discovered association rules, in: Proceedings of the fifth international conference on database systems for advanced applications (DASFAA), 1997, pp. 185–194.

[66] C. Diehl, G. Cauwenberghs, Svm incremental learning, adaptation and optimization, in: Proceedings of the International Joint Conference on Neural Networks, Vol. 4, IEEE, 2003, pp. 2685–2690.

[67] A. Shilton, M. Palaniswami, D. Ralph, A. Tsoi, Incremental training of support vector machines, IEEE Transactions on Neural Networks 16 (1) (2005) 114–131.

[68] R. Polikar, L. Upda, S. Upda, V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 31 (4) (2001) 497–508.

[69] N. Littlestone, M. Warmuth, The weighted majority algorithm, in: 30th Annual Symposium on Foundations of Computer Science, IEEE, 1989, pp. 256–261.

[70] J. Xie, Z. Qiu, The effect of imbalanced data sets on lda: A theoretical and empirical analysis, Pattern recognition 40 (2) (2007) 557–562.

[71] J. Xue, D. Titterington, Do unbalanced data have a negative effect on lda?, Pattern Recognition 41 (5) (2008) 1558–1571.

[72] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, K. Mullers, Fisher discriminant analysis with kernels, in: Proceedings of the IEEE Signal Processing Society Workshop, IEEE, 1999, pp. 41–48.

[73] S. Petridis, S. Perantonis, On the relation between discriminant analysis and mutual information for supervised linear feature extraction, Pattern Recognition 37 (5) (2004) 857–874.

[74] H. Wang, W. Zheng, Local temporal common spatial patterns for robust single-trial eeg classification, IEEE Transactions on Neural Systems and Rehabilitation Engineering 16 (2) (2008) 131–139.

[75] Y. Liu, W. Rayens, Pls and dimension reduction for classification, Computational Statistics 22 (2) (2007) 189–208.

[76] D. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: An overview with application to learning methods, Neural Computation 16 (12) (2004) 2639–2664.

[77] D. N. A. Asuncion, UCI machine learning repository (2007).

[78] O. Viikki, K. Laurila, Cepstral domain segmental feature vector normalization for noise robust speech recognition, Speech Communication 25 (1-3) (1998) 133–147.

[79] M. Barker, W. Rayens, Partial least squares for discrimination, Journal of chemometrics 17 (3) (2003) 166–173.

[80] E. Tang, P. Suganthan, X. Yao, A. Qin, Linear dimensionality reduction using relevance weighted lda, Pattern recognition 38 (4) (2005) 485–493.

[81] K. Morik, P. Brockhausen, T. Joachims, Combining statistical learning with a knowledge-based approach-a case study in intensive care monitoring, in: Machine Learning-International Workshop then conference, Morgan Kaufmann Publishers, INC., 1999, pp. 268–277.

[82] T. Nitta, Solving the xor problem and the detection of symmetry using a single complex-valued neuron, Neural Networks 16 (8) (2003) 1101–1105.

[83] C. Fellbaum, WordNet: An electronic lexical database, The MIT press, 1998.

[84] C. Strapparava, A. Valitutti, Wordnet-affect: an affective extension of wordnet, in: Proceedings of LREC, Vol. 4, Citeseer, 2004, pp. 1083–1086.

[85] B. Schuller, S. Steidl, A. Batliner, F. Schiel, J. Krajewski, The INTERSPEECH 2011 Speaker State Challenge, in: Interspeech (2011), ISCA, Florence, Italy, 2011.

[86] S. de Jong, Simpls: an alternative approach to partial least squares regression, Chemometrics and intelligent laboratory systems 18 (3) (1993) 251–263.

[87] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2006) 489–501.

[88] N. Liang, P. Saratchandran, G. Huang, N. Sundararajan, Classification of mental tasks from eeg signals using extreme learning machine, International Journal of Neural Systems 16 (1) (2006) 29–38.

[89] R. Zhang, G. Huang, N. Sundararajan, P. Saratchandran, Multicategory classification using an extreme learning machine for microarray gene expression cancer diagnosis, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 4 (3) (2007) 485–495.

[90] D. Wang, G. Huang, Protein sequence classification using extreme learning machine, in: IEEE International Joint Conference on Neural Networks IJCNN'05, Vol. 3, IEEE, 2005, pp. 1406–1411.

[91] E. Baum, On the capabilities of multilayer perceptrons, Journal of complexity 4 (3) (1988) 193–215.

[92] N. Brummer (Ed.), FoCal Multiclass Toolkit, https://sites.google.com/site/nikobrummer/focalmulticlass.

[93] F. Wei, W. Li, Q. Lu, Y. He, Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization, in: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2008, pp. 283–290.

[94] D. Klein, C. Manning, Accurate unlexicalized parsing, in: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1, Association for Computational Linguistics, 2003, pp. 423–430.

[95] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web, Tech. rep., Technical report, Stanford Digital Library Technologies Project (1998).

[96] G. Erkan, D. Radev, LexRank: Graph-based lexical centrality as salience in text summarization, Journal of Artificial Intelligence Research 22 (1) (2004) 457–479.

[97] W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, ACM, 2003, p. 273.

[98] J. Otterbacher, G. Erkan, D. Radev, Using random walks for question-focused sentence retrieval, in: Proceedings of the conference on Human

Language Technology and Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2005, pp. 915–922.

[99] X. Wan, J. Yang, J. Xiao, Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction, in: annual meeting of Association for Computational Linguistics, Vol. 45, 2007, p. 552.

[100] H. Zha, Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering, in: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2002, pp. 113–120.

[101] S. Harabagiu, F. Lacatusu, Using topic themes for multi-document summarization, ACM Transactions on Information Systems (TOIS) 28 (3) (2010) 1–47.

[102] Y. Pan, S. Ge, A. Al Mamun, Weighted locally linear embedding for dimension reduction, Pattern Recognition 42 (5) (2009) 798–811.

[103] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323.

[104] S. Ge, F. Guan, Y. Pan, A. Loh, Neighborhood linear embedding for intrinsic structure discovery, Machine Vision and Applications 21 (3) (2010) 391–401.

[105] P. Over, Introduction to duc-2001: an intrinsic evaluation of generic news text summarization systems, in: DUC-01 Workshop on Text Summarization, Vol. 1430, 2001, p. 3345.

[106] C. J. van Rijsbergen, Information Retrieval, Butterworth-Heinemann, 1979.

[107] W. Frakes, R. Baeza-Yates, Information retrieval: Data structures & algorithms, Prentice Hall Englewood Cliffs, NJ, 1992.

[108] C. Lin, Rouge: A package for automatic evaluation of summaries, in: Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), 2004, pp. 25–26.

[109] Y. Gong, X. Liu, Generic text summarization using relevance measure and latent semantic analysis, in: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2001, pp. 19–25.

[110] R. Mihalcea, P. Tarau, TextRank: Bringing order into texts, in: Proceedings of EMNLP, Barcelona: ACL, 2004, pp. 404–411.

[111] J. Lee, S. Park, C. Ahn, D. Kim, Automatic generic document summarization based on non-negative matrix factorization, Information Processing & Management 45 (1) (2009) 20–34.

[112] R. Mihalcea, C. Corley, C. Strapparava, Corpus-based and knowledge-based measures of text semantic similarity, in: Proceedings of the national conference on artificial intelligence, Vol. 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, p. 775.

[113] A. Huang, Similarity measures for text document clustering, in: Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand, 2008, pp. 49–56.

[114] S. Kim, A. Valitutti, R. Calvo, Evaluation of unsupervised emotion models to textual affect recognition, in: Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, Association for Computational Linguistics, 2010, pp. 62–70.

[115] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, Journal of the American society for information science and technology 58 (7) (2007) 1019–1031.

[116] P. Turney, Mining the web for synonyms: Pmi-ir versus lsa on toefl, in: Proceedings of the 12th European Conference on Machine Learning, Springer-Verlag, 2001, pp. 491–502.

[117] R. Mihalcea, Using wikipedia for automatic word sense disambiguation, in: Proceedings of NAACL HLT, Vol. 2007, 2007, pp. 196–203.

[118] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, E. Ruppin, Placing search in context: The concept revisited, in: Proceedings of the 10th international conference on World Wide Web, ACM, 2001, pp. 406–414.

Publications and Awards

The contents of this thesis are based on the following papers that have been published, accepted, or submitted to the peer-reviewed journals and conferences.

Journal papers

[1] Shuzhi Sam Ge, Zhengchen Zhang, Incremental Learning for Data Association, Knowledge and Information Systems, Under Review

[2] Zhengchen Zhang, Shuzhi Sam Ge, Hongsheng He, Qun Zhang, LDA-Imbalance: A Linear Discriminant Analysis based Classifier for Imbalanced Data Sets, Data Mining and Knowledge Discovery, Under Review

[3] Dong-Yan Huang, Zhengchen Zhang, Shuzhi Sam Ge, Speaker State Classification Based on Fusion of Asymmetric Simple Partial Least Squares (SIMPLS) and Support Vector Machines, Computer Speech and Language, Under Review

[4] Hongsheng He, Shuzhi Sam Ge, Zhengchen Zhang, An Attention-Driven Robotic Head with Biological Saccade Behaviors for Social Robots, Autonomous Robots, Under Review

[5] Zhengchen Zhang, Shuzhi Sam Ge, Hongsheng He, Mutual-Reinforcement Document Summarization Using Embedded Graph Based Sentence Clustering, Information Processing and Management, 2012

[6] Shuzhi Sam Ge, Hongsheng He, Zhengchen Zhang, Bottom-up saliency detection for attention determination, Machine Vision and Applications, 2011

[7] Hongsheng He, Shuzhi Sam Ge, Zhengchen Zhang, Visual Attention Prediction Using Saliency Determination of Scene Understanding for Social Robots, Special Issue on Towards an Effective Design of Social Robots, International Journal of Social Robotics, 2011

[8] John-John Cabibihan, Wing-Chee So, Sujin Saj, Zhengchen Zhang, Telerobotic Pointing Gestures Shape Human Spatial Cognition, International Journal of Social Robotics, 2012

Conference papers

[1] Zhengchen Zhang, Shuzhi Sam Ge and Keng Peng Tee, Emotional Sentence Identification in a Story, Workshop at SIGGRAPH ASIA (WASA) 2012

[2] Shuzhi Sam Ge, Zhengchen Zhang and Hongsheng He, Weighted graph model based sentence clustering and ranking for document summarization, The 4th International Conference on Interaction Sciences (ICIS), 2011

[3] Shuzhi Sam Ge, John-John Cabibihan, Zhengchen Zhang, Yanan Li, Cai Meng, Hongsheng He, Mohammadreza Safi Zadeh, Yinbei Li and Jiaqiang Yang, Design and Development of Nancy, a Social Robot, The 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2011

[4] Dong-Yan Huang, Shuzhi Sam Ge and Zhengchen Zhang, Speaker State Classification Based on Fusion of Asymmetric SIMPLS and Support Vector Machines, Twelfth Annual Conference of the International Speech Communication Association, 2011

[5] Hongsheng He, Zhengchen Zhang and Shuzhi Sam Ge, Attention determination for social robots using salient region detection, International Conference on Social Robotics, 2010

[6] Xinyang Li, Shuzhi Sam Ge, Yaozhang Pan, Keum-Shik Hong, Zhengchen Zhang, Xiaosu Hu, Feature extraction based on common spatial analysis for time domain parameters, The 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2011

International awards

1. Dong-yan Huang, Shuzhi Sam Ge, and Zhengchen Zhang, The Sleepiness Sub-Challenge Prize, the INTERSPEECH 2011 Speaker State Challenge