



Probabilistic Distance Measure Algorithm Implementation

By

Mr. Kjitjate Sapmanee

Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Science
in Computer Science
Assumption University

September, 2001

Probabilistic Distance Measure Algorithm Implementation

By

Mr. Kjitjate Sapmanee
Adm. No. G3929802



**Submitted in Partial Fulfillment of
The Requirements for the Degree of
Master of Science
in Computer Science
Assumption University**

September 2001

The Faculty of Science and Technology

Thesis Approval

Thesis Title Probabilistic Distance Measure Algorithm Implementation

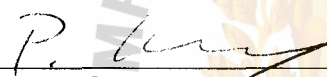
By Mr. Kjitjate Sapmanee

Thesis Advisor Associate Professor Dr. Peter Haddawy


Academic Year 1/2001


The Department of Computer Science, Faculty of Science and Technology of Assumption University has approved this final report of the **twelve** credits course. **SC7000 Master Thesis**, submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Approval Committee:

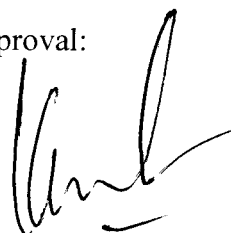

(Associate Professor Dr. Peter Haddawy)
Advisor



(Dr. Thitipong Tanprasert)
Committee Member


(Dr. Jirapun Daengdej)
Committee Member


(Asst. Prof. Dr. Surapong Auwatanamonkol)
Representative of Ministry of
University Affairs

Faculty Approval:

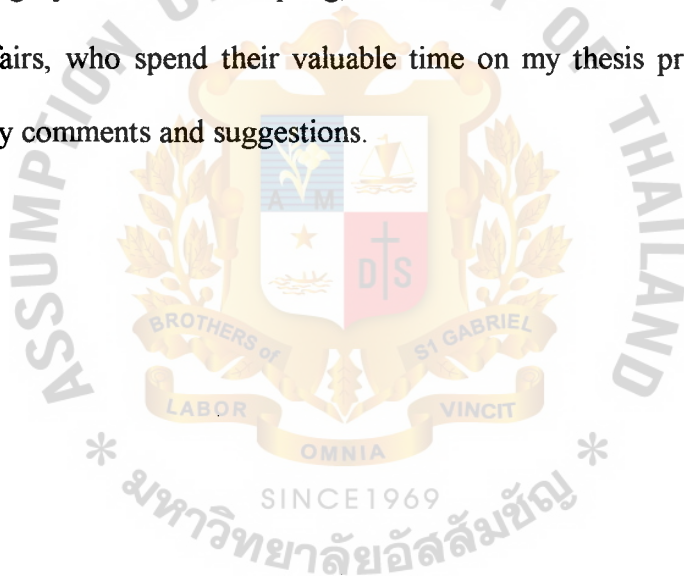

(Asst. Prof. Dr. Tang Van To)
Director


(Asst. Prof. Dr. Pratit Santiprabhob)
Dean

ACKNOWLEDGEMENT

First of all, I would like to thank you my advisor, Dr. Peter Haddawy for his kindly advice and helpful assistance. He always shows his kindness, encouragement and suggestions when this thesis faced the problems. Secondly, I would like to give many thanks to my family those always cheer me up and encourage me to continue the study in this master degree.

I would like to acknowledge my committees, Dr. Thitipong Tanprasert, Dr. Jirapun Daengdej and Dr. Surapong, the external committee from ministry of university affairs, who spend their valuable time on my thesis presentation and give me the worthy comments and suggestions.



ABSTRACT

Collaborative filtering systems use a database of user preferences to find users with similar preferences in order to recommend items to a given user. A central part of such systems, on which the quality of the recommendations depends, is the measure used to determine similarity of preferences. When a user asks for recommendations, the system uses the preferences of other similar users to predict the preferences of the current user for unpurchased or unseen items. In this thesis, a probabilistic distance measure approach is used in creating an algorithm that has desired properties of a good similarity measure. The distance between two users' preferences is defined as the probability that two randomly chosen items are ranked differently by the two users. The algorithm is empirically evaluated against a commonly used collaborative filtering algorithm based on the Pearson correlation coefficient. The exhaustive experiments show the behavior of the algorithms over various databases and show the optimal similarity neighborhood for each algorithm

TABLE OF CONTENTS

ACKNOWLEDGEMENT.....	I
ABSTRACT.....	II
LIST OF FIGURES.....	III
LIST OF TABLES.....	IV
LIST OF TABLES.....	IV
LIST OF GRAPHS.....	V
INTRODUCTION	1
Rationale of the Study	1
Problem Statement.....	2
Objective of the Study	2
Methodology of the Study.....	2
Scope and Limitation	3
CHAPTER 1: BACKGROUND AND RELATE TOPICS	4
1.1 Background of the Problem.....	4
1.2 Similarity Measurement	6
Existing Similarity Measure.....	6
Similarity Theories	8
1.3 Missing Data.....	10
1.4 The EachMovie database	12
CHAPTER 2: REVIEW OF THE LITERATURES.....	14

2.1 Personalization Systems.....	14
1. Rule-based systems	14
2. CASE (Computer-Assisted Self-Explication)	16
3. Endorsement systems	17
4. Collaborative filtering	18
2.2 Collaborative Filtering Systems	20
GAB	20
GroupLens	22
Do-I-Care	23
Pointers & Digests	24
Phoaks	25
2.3 Collaborative Filtering Algorithms.....	26
Correlation-based Prediction	26
Support Vector Method.....	28
Correlation-based Prediction Using Clusters of Users	32
2.4 Empirical Evaluation of Collaborative Filtering Algorithms.....	35
Empirical Evaluation from Danyel paper [Swami]	35
Empirical Evaluation from Breese paper [Breese98]	39
CHAPTER 3: DESIGN AND METHODOLOGY	41
3.1 The Probabilistic Distance Measure Algorithm Design	41
Case 1: No score is given	44
Case 2: A score s is given for item i	46
Case 3: One user gives score for two items	48
Case 4: Both users give a score on different item	50
Case 5: Both users give a item score on the same item	52

Case 6: Only one of I_1 score is missing	54
Case 7: Only one of I_2 score is missing	56
Case 8: All scores are given	58
3.2 Metrics Qualification	59
3.3 Experiment Methodology.....	60
CHAPTER 4: ALGORITHM AND DATA STRUCTURE	64
Architecture Overview	64
Algorithm and Data structure of Modules	66
Part 1: Data Input Module.....	66
Part 2: Similarity Measure Module.....	68
Part 3: Prediction Experiment Module	74
Part 4: Output Module	82
CHAPTER 5: EMPIRICAL ANALYSIS	83
5.1 Evaluation Criteria	83
5.2 Experiment Results	85
5.3 Discussions	108
5.4 Conclusions	110
CHAPTER 6: FUTURE WORKS	111
CHAPTER 7: REFERENCES	112

LIST OF FIGURES

Figure 3.1: Probability calculation when no score is given.....	44
Figure 3.2: Probability calculation when only one score is given	46
Figure 3.3: Probability calculation when one user gives scores for two items.....	48
Figure 3.4: Probability calculation when both users give a score on different items..	50
Figure 3.5: Probability calculation when both users give a score on the same item...	52
Figure 3.6: Probability calculation when only one of I_1 score is missing	54
Figure 3.7: Probability calculation when only one of I_2 score is missing	56
Figure 3.8: Probability calculation when all item scores are given	58
Figure 3.9: Experimental data sets	61
Figure 4.1: Overall algorithm of experimental program	65
Figure 4.2: Algorithm of data input nodule.....	66
Figure 4.3: Probabilistic Distance Measure Algorithm.....	70
Figure 4.4: Pearson correlation coefficient algorithm.....	72
Figure 4.5: Prediction Experiment Module	74
Figure 4.6: Movie votes filling back algorithm	76
Figure 4.7: Prediction Based on Correlation Based Algorithm	78
Figure 4.8: Prediction Based on Most Similarity Votes Algorithm.....	80

LIST OF TABLES

Table 2.1: Performance comparison reported from Breese paper..... 40

Table 4.1: Encoding table for two users’ preferences 69

Table 5.1: Optimum point of the algorithms 110



LIST OF GRAPHS

Graph 5.1: Mean error of users with 10 to 50 votes (AWS)..... 86

Graph 5.2: Mean error of users with 10 to 50 votes (COR) 86

Graph 5.3: Mean error of users with 50 to 100 votes (AWS)..... 87

Graph 5.4: Mean error of users with 50 to 100 votes (COR) 87

Graph 5.5: Mean error of users with 100 to 150 votes (AWS)..... 88

Graph 5.6: Mean error of users with 100 to 150 votes (COR)..... 88

Graph 5.7: Mean error of users with 150 to 250 votes (AWS)..... 89

Graph 5.8: Mean error of users with 150 to 250 votes (COR)..... 89

Graph 5.9: Mean error of users with 10 to 250 votes (AWS)..... 90

Graph 5.10: Mean error of users with 10 to 250 votes (COR)..... 90

Graph 5.11: Error variance of users with 10 to 50 votes (AWS)..... 91

Graph 5.12: Error variance of users with 10 to 50 votes (COR)..... 91

Graph 5.13: Error variance of users with 50 to 100 votes (AWS)..... 92

Graph 5.14: Error variance of users with 50 to 100 votes (COR)..... 92

Graph 5.15: Error variance of users with 100 to 150 votes (AWS) 93

Graph 5.16: Error variance of users with 100 to 150 votes (COR)..... 93

Graph 5.17: Error variance of users with 150 to 250 votes (AWS)..... 94

Graph 5.18: Error variance of users with 150 to 250 votes (COR)..... 94

Graph 5.19: Error variance of users with 10 to 250 votes (AWS)..... 95

Graph 5.20: Error variance of users with 10 to 250 votes (COR)..... 95

Graph 5.21: Mean error of users with 10 to 50 votes (AWS)..... 96

Graph 5.22: Mean error of users with 10 to 50 votes (COR) 96

Graph 5.23: Mean error of users with 50 to 100 votes (AWS)..... 97

Graph 5.24: Mean error of users with 50 to 100 votes (COR).....	97
Graph 5.25: Mean error of users with 100 to 150 votes (AWS).....	98
Graph 5.26: Mean error of users with 100 to 150 votes (COR).....	98
Graph 5.27: Mean error of users with 150 to 250 votes (AWS).....	99
Graph 5.28: Mean error of users with 150 to 250 votes (COR).....	99
Graph 5.29: Mean error of users with 10 to 250 votes (AWS)	100
Graph 5.30: Mean error of users with 10 to 250 votes (COR).....	100
Graph 5.31: Error variance of users with 10 to 50 votes (AWS).....	101
Graph 5.32: Error variance of users with 10 to 50 votes (COR).....	101
Graph 5.33: Error variance of users with 50 to 100 votes (AWS)	102
Graph 5.34: Error variance of users with 50 to 100 votes (COR).....	102
Graph 5.35: Error variance of users with 100 to 150 votes (AWS).....	103
Graph 5.36: Error variance of users with 100 to 150 votes (COR).....	103
Graph 5.37: Error variance of users with 150 to 250 votes (AWS).....	104
Graph 5.38: Error variance of users with 150 to 250 votes (COR).....	104
Graph 5.39: Error variance of users with 10 to 250 votes (AWS).....	105
Graph 5.40: Error variance of users with 10 to 250 votes (COR).....	105
Graph 5.41: Average of Mean Prediction error (2,000 users data set).....	106
Graph 5.42: Average of Mean Prediction error (500 users data set).....	106
Graph 5.43: Average of Prediction Error Variance (2,000 users data set).....	107
Graph 5.44: Average of Prediction Error Variance (500 users data set).....	107

INTRODUCTION

Rationale of the Study

Similarity measures are widely used in the field of Information Retrieval. Information retrieval is concerned with the representation, storage, organization and accessing information items [Salton83]. One area, which has received particular attention in recent years, is the use of similarity measures in recommender systems used for personalization. Similarity measures lie at the heart of collaborative filtering, one of the most popular techniques for building recommender systems. Collaborative filtering has been effectively used by companies such as Amazon.com to recommend items like books and music CDs to customers. Collaborative filtering works by finding people who have similar likes or interests and then making recommendations based on items one person likes that the other one has not yet experienced or purchased. The quality of the result produced by the collaborative filtering system depends to a large extent on the quality of the similarity measure used.

A variety of similarity measures have been proposed and used in collaborative filtering. Some of them are not true metrics. Those similarity measures may also have limitations on scaling i.e. the result is vastly different between large and small set of data. This study presents another approach which scaling does not have high effect on measuring and it satisfies the conditions of metric. Moreover, the proposed similarity measure algorithm is found to achieve high accuracy.

This thesis presents an exact algorithm for computing probabilistic distance when preferences of users are defined as numeric ratings. This computation uses an implementation of Ha and Haddawy's [Ha99] probabilistic distance measure, which measures the similarity of two people's preferences. The measure has a number of advantages over other commonly used similarity measures, which we demonstrate through empirical comparisons.

Problem Statement

Similarity measure plays main role on the performance of personalization systems. It acts like the core engine of those systems but there is not enough concern on improving such engine to match desired conditions such as metric specifications, accuracy and missing data treatment.

Objective of the Study

- To review the similarity measure algorithms and to criticize their advantages and disadvantages with other type of algorithms.
- To develop an algorithm based on probabilistic distance measure algorithm. The newly developed algorithm uses probability to predict missing information. The outcome from such an algorithm will be data-size independent, and highly reliable and accurate.
- To do empirical evaluation of the accuracy of the developed probabilistic distance measure algorithm.

Methodology of the Study

This study reviews the similarity measures from various sources, including proceeding papers and textbooks. Relevant information of this algorithm will be used

to define and evaluate new algorithm based on probabilistic distance measure proposed in this study.

The new probabilistic distance measure algorithm will be tested by using a sample database. The selected sample database for testing purpose is *Eachmovie* (the movie database collected by Digital Equipment Corporation. The detailed information is in section 1.5). This database is chosen because it has many real life data.

Scope and Limitation

The use of similarity measure algorithms will be concisely discussed. The topics will include their behavior and advantages of these algorithms.

The proposed similarity measure algorithm will be developed based on the probabilistic distance measure. The newly developed algorithm will rely on the probability of the correspondent of the preferences of the two objects. For instance, the probability of the partial preferences of the two users will be calculated.

The implementation of this proposed algorithm will be shown and explained. Finally, the evaluation of its performance will be made.

CHAPTER 1: BACKGROUND AND RELATE TOPICS

1.1 Background of the Problem

Products in the market have been evolved continuously. In addition, new advertising approaches have been utilized to increase products' competitiveness. Customers have to choose the product that satisfactorily fulfills their needs. However, choosing the product that most satisfies the needs from hundreds of product varieties are usually a trial and error process. In general, the process requires unnecessary time, and often the chosen product does not perfectly fulfill the customer's need.

A number of technologies have been developed to help customers choose the products that fit their needs most. Most of the recent technologies are computer-based. One of the technologies that were developed is personalization system. The personalization system is a computer-based system that gathers and analyses the existing information relevant to the given problems. Results from this system will be a good or even the optimal solutions for the given problems. Currently, there are various kinds of personalization system evolved, which they are built to apply for different situations.

The personalization system that is relating to this study is a collaborative filtering system (more details will be discussed in Chapter 2). The system is based on a similarity measure algorithm. Such an algorithm gathers information of the considered similar users from the database. Then, the preferences of the considered similar users will be analyzed. At the end of the algorithm, the system will provide the recommendation to the user. The recommendation quality is highly dependent on

the relevance of information of the other users that are gathered. This relevance is governed by the quality of the similarity measure algorithm. In other words, using the appropriate similarity measure algorithm results in getting the highly relevant information, which is the fundamental for providing the optimal recommendation to the user.



1.2 Similarity Measurement

Existing Similarity Measure

Presently, a number of existing similarity measurements have been evolved. These are both for general purposes and specific purposes that create for some certain tasks. The following equations are the examples of similarity measurements:

- Euclidean distance

$$d(x, y) = \sqrt{\sum_{i=1}^l (x_i - y_i)^2}$$

where x_i, y_i are the i th coordinates of x and y respectively.

- Tanimoto Distance

$$S_T(x, y) = \sqrt{\frac{x^T y}{\|x\|^2 + \|y\|^2 - x^T y}}$$

where $x^T y$ is inner product of x, y which is equal to $\sum_{i=1}^l x_i y_i$.

- Mahalanobis

$$d(x, y) = \sqrt{(x - y)^T B (X - Y)}$$

where B is a symmetric positive definite matrix.

- Pearson correlation

$$r(x, y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

where \bar{x}, \bar{y} is average of x, y respectively.

- Cosine coefficient

$$\cos(x, y) = \frac{\sum_{i=1}^l (x_i y_i)}{\sqrt{\sum_{i=1}^l (x_i)^2 \sum_{i=1}^l (y_i)^2}}$$

Mostly, these measurements deal with commonly given preferences from two users. It means that in case of comparing similarity of the two users, both users must have some common preferences. Then those preferences will be used to calculate the distance between the two users.

Similarity Theories

This section reviews some theories from Pattern Recognition by Sergios Theodoridis [Sergios98] which concerning similarity measurement. The review begins with definitions concerning measurement between vectors.

A dissimilarity measure (DM) d on X is a function.

$$d : X \times X \rightarrow \mathfrak{R}$$

where \mathfrak{R} is the set of real number such that

$$\exists d_0 \in \mathfrak{R} : -\infty < d_0 \leq d(x, y) < +\infty, \forall x, y \in X$$

where

$$d_0 = d(x, x), \forall x \in X$$

$$d(x, y) = d(y, x), \forall x, y \in X$$

If in addition

$$d(x, y) = d_0 \text{ if and only if } x = y$$

$$d(x, z) \leq d(x, y) + d(y, z), \forall x, y, z \in X$$

d is called a metric dissimilarity measure.

A similarity measure (SM) s on X is a function.

$$s : X \times X \rightarrow \Re$$

where \Re is the set of real number such that

$$\exists s_0 \in \Re : -\infty < s(x, y) \leq s_0 < +\infty, \forall x, y \in X$$

where

$$s_0 = s(x, x), \forall x \in X$$

$$s(x, y) = s(y, x), \forall x, y \in X$$

If in addition

$$s(x, y) = s_0 \text{ if and only if } x = y$$

$$s(x, y)s(y, z) \leq [s(x, y) + s(y, z)]s(x, z), \forall x, y, z \in X$$

s is called a metric similarity measure.

1.3 Missing Data

A problem commonly found in real-life applications is how to deal with missing data. This means that for some users, their preferences are partially unknown. This may be a consequence of a failure of the measuring device. The following techniques are some commonly used techniques for handling this missing data situation [Sergios98].

1. Discard all unknown preferences. This approach may be used when the number of unknown preferences is small compared with the total number of preferences. If this is not the case, the nature of the problem may be affected.
2. For the missing preferences, find mean value based on the available values of all known preferences of that user. Then substitute this value for the missing preferences.
3. For all the pairs l of preferences x_i and y_i of the user x and y define b_i as

$$b_i = \begin{cases} 0, & \text{if both } x_i \text{ and } y_i \text{ are available} \\ 1, & \text{otherwise} \end{cases}$$

Then, the proximity between x and y is defined as

$$\phi(x, y) = \frac{l}{l - \sum_{i=1}^l b_i} \sum_{all \ i: b_i=0} \phi(x_i, y_i)$$

Where $\phi(x_i, y_i)$ denotes the proximity between the two preferences x_i and y_i . The rationale behind this approach is simple. Let $[a, b]$ be the interval of the allowable values of $\phi(x, y)$. The preceding definition ensures that the proximity measure between x and y spans all $[a, b]$, regardless of the number of unavailable preferences in both users.

4. Find the average proximity $\phi_{avg}(i)$ between all preferences of user x and y along all components $i = 1, \dots, l$. For some preferences that are not available, they are excluded from the computation of $\phi_{avg}(i)$. The proximity $\psi(x_i, y_i)$ between the i th preferences of x and y is defined as $\phi_{avg}(i)$ if at least one of the x_i and y_i is not available. If both x_i and y_i are available, The proximity $\psi(x_i, y_i)$ is defined as $\phi(x_i, y_i)$. Then, the proximity between the two user x, y is defined as

$$\phi(x, y) = \sum_{i=1}^l \psi(x_i, y_i)$$

1.4 The EachMovie database

The Compaq Systems Research Center ran the EachMovie recommendation service [Eachmovie] for 18 months to experiment with a collaborative filtering algorithm. During that time, some 72,916 users entered a total of 2,811,983 numeric ratings for 1,628 different movies (films and videos).

The file retrieved to use in the experimentation of this study is the vote score (vote.txt), which contains the following information:

Person_ID: Number
Movie_ID: Number
Score: Number (0 <= Score <= 1)
Weight: Number (0 < Weight <= 1)
Modified: Date/Time

Score is the rating provided by one person for a particular movie. The zero-to-five star rating used externally on *EachMovie* is mapped linearly to the interval [0,1]. A histogram of the Score values is shown below:

Score	Count
0	347191
0.2	150495
0.4	339718
0.6	701236
0.8	761676
1.0	511667

Weight is only relevant in the case of a Score of zero, in which case it distinguishes whether the person had seen the movie and rated a movie as zero stars (weight = 1) or indicates that the person had never planned to see the movie because that person thought that the movie was not good (weight < 1).

For the experiments in this study, a number of users were randomly picked to be the tested data sets. In order to test the scaling effect of user size, two types of data sizes were generated as follows:

- 500 users data sets
- 2,000 users data sets

In addition, to test the scaling effect of preferences size, for each type of data sizes, five data sets are randomly picked and they are categorized by number of votes as follows:

- 10 to 50 movie votes
- 51 to 100 movie votes
- 101 to 150 movie votes
- 151 to 250 movie votes
- 10 to 250 movie votes

CHAPTER 2: REVIEW OF THE LITERATURES

In addition to focusing on the collaborative filtering, this study reviews other personalization systems. The review covers behavior, benefit and advantages of each system. In addition, the comparison on those attributes and features among reviewed systems are made.

2.1 Personalization Systems

According to Principles of Internet Marketing [Hanson00], makers of mass-produced products vary their marketing approaches based on types of attributes and features that make up the product. Products with a few simple attributes compete primarily on value and price. Customers understand the tradeoffs and look for the best deal. On the other hand, when product features and attributes are complex and qualitative, brand name and image are most important. Personalization's value increases as the opportunities for differentiation, customization, and catering to individual tastes increase.

The following systems are four of the main online personalization systems. Each of these systems is suitable for different situations. The right choice of personalization system depends on customer needs and product attributes.

1. *Rule-based systems*

Rule-based systems use the information that a company develops. The content of these systems are information of the company's customers and its outcomes are used to make educated guesses of special offers, promotions, and information that it

should provide to visitors. These systems are capable of presenting specialized information to each arrival. Because rule-based systems work by observing behavior and predicting preferences, they work best in situations where the product space is not too complicated and attributes are quantifiable.

BroadVision is one of the leading providers of complex rule-based tools. The goal is to combine personalization with other important online features. The system must connect to the different databases used by a company, be available to the transaction parts of the site, interact with managers and developers, and effectively capture and manage knowledge. An example of a site that uses the BroadVision approach is the Kodak Picture Network. Kodak's system allows consumers to store and share pictures over the Internet. Anyone taking a picture can have it made into prints and stored online. People around the world can share pictures. The Kodak system needs to track all pictures, which are all need to be available for sharing and editing in real time.

From their interactive and transparently observing behaviors, rule-based systems gain benefits including abilities to offering promotions, connecting online ads, messages to current user. In addition, they can run time-sensitive content, statistic reports, and tracking traffic while users interact with the systems. In contrast with CASE System, they work without forcing users to answer questions and constantly fill in extensive questionnaires.

2. CASE (*Computer-Assisted Self-Explication*)

CASE is the online system that asks visitors about their preferences. Each system has a large database of possible choices. This database reflects the complicated set of user needs and products available. The objective of the system is to help users narrow their choices from thousands of possibilities to a few highly ranked alternatives. Because consumers differ in their tastes and many choices are possible, these systems must have active participation and active dialogue with consumers.

The fundamental of this approach is identifying the valuable attributes by the users. In general, products and services are composed of attributes that users can identify the most valuable attributes. This complicate identification process could be done manually by the user, but the accuracy, truthfulness and the usefulness of the answers from this manual identification may be questionable. Therefore, the challenge of building a CASE is to let user identify the desired attributes in a way that encourages accurate, truthful and useful answers.

The answers from the user lead to simplification of possible choices. Three features are important in making questionnaires reliable:

- Users should be asked whether there is any unacceptable feature
- User should be asked whether there is any compulsory feature
- Users then should be questioned about the importance of key features

An example for the CASE system is Personalogic. This system helps consumers on finding a bicycle to purchase. The system asks the consumers for their style and maximum price, then make questions about intended usage and comfort

issues. Each of the “must have, can’t use” elimination steps will reduce number of bicycle choices into top ranks set.

The CASE systems have advantages over rule-based systems by allowing users to identify their preferences directly instead of indirectly predicting the preferences from the users’ behaviors. That means that CASE systems are appropriate for the situation that there exists relatively small number of well-understood attributes and features to be evaluated.

3. *Endorsement systems*

Endorsement systems are appropriate for the situation that the products which consumers need do not differ greatly. However, judging quality and explaining the value of available products are a challenge. In these circumstances, a simple endorsement can be best. Endorsement systems are helpful when judging and evaluating the products before use are difficult. The difficulties include the list and measurement of the important attributes and generally are from the qualitative reasons such as complexity of the attributes.

In complicated settings, with experience goods and qualitative attributes, the best personalization service may simply be an endorsement. This is especially true when customer needs and tastes do not vary widely. In this case, the many various attributes can often be simplified to a guarantee that a minimum quality level has been achieved.

The ValueStar system takes this endorsement approach. Dealers and service providers that are part of the system have agreed to take customer survey for quality and satisfaction. The rankings from the survey are used to ensure that the system will recommend providers exceed a quality threshold. In addition, the system also keeps the geographic location of providers. When a user of the ValueStar system enters the zip code and category of service needs, the system returns the qualify vendors that near the user's location.

The major benefit of an endorsement system is a choice simplification. For example, customers need to know that a service provider is competent, honest and offers high quality. The endorsement system chooses the qualified service provider from its database and connects users with local preferred providers.

4. *Collaborative filtering*

Collaborative filtering is a leading approach when the product space is complicated and preferences are highly subjective, qualitative, and complex. The goal of these systems is to establish the system of educated word of mouth. The system matches different users who seem to share similar tastes. These similar individuals can then share recommendations and preferences about hard-to-judge products and services.

The collaborative filtering goal is to effectively identify individuals who share similar tastes. Recommendations are useful when people share preferences. This leads to computer-assisted word of mouth. The system starts by asking a visitor to rank and compare a number of alternatives. Then the system searches through its database and

tries to find other users with similar views. If matches are made, the system tries to find highly ranked choices of these matched users and become recommendations.

Amazon.com uses collaborative filtering in recommending their products online. When an Amazon.com user wants to find some new music CDs, the system searches through the list and get some other users, which their tastes fairly close the current user. Then the current user gets recommendation from the highly ranked CDs of those other users, which are not listened by the current user.

The collaborative filtering heavily relies on information from other users to make recommendations. It works well when the customer and product space settings are too complex for the marketer to effectively make personalization recommendations based on formal models.



2.2 Collaborative Filtering Systems

From Berkeley Workshop on Collaborative Filtering [Sims96], collaborative filtering system is a system that helps users to find the information they want by taking advantage of each others' experiences with information sources. These experiences can then be used to predict a particular user's preferences based on similarity between that particular user and other users.

Summary below provides an overall description of each of the systems, the problems that the system seeks to resolve and/or the system's comparative advantages. In addition, it covers the topics that are under study.

GAB

GAB is part of a group of collaborative filtering activities at Bellcore called "FINE" (pronounced "fee'nay", standing for Find It Now Engine). The other filtering activities in this group also include the Bellcore Video Recommender (a recommendation-based system whereby individuals rate videos), and Bellcore Advisor, which utilizes standard information retrieval techniques, that are now used for finding documents, to find people.

As described by Mark Rosenstein (Bellcore), GAB uses hierarchical decompositions made by individuals and provides browsing facilities over Web pages. GAB enables users to find equivalence classes of Web pages (i.e., under what class or classes of items their favorite URLs appear). Once those equivalence classes are located, GAB provides users with browseable interfaces of those representations.

GAB also enables a user to select and specify a personal grouping of others' opinions or choices to follow (e.g., hotlists, bookmarks).

For the middle level user who is interested in a particular domain, has some knowledge of Web space for that domain and is willing to let others be more active in searching out sources, *GAB* is ideal. However, for the novice who has little experience with a domain or the expert who has considerable familiarity, *GAB* is not as useful a tool.

Additionally, *GAB*'s ability to reach into individual's bookmarks and extract information (e.g., perhaps a personal bookmark on medical information) raises privacy concerns. To overcome this problem, however, users could edit properties of bookmark headings to be shared (using a keyword, "public") so that when the *GAB* search engine looked for "bookmarks," only those categories marked "public" would be indexed. Furthermore, since the "sociology of *GAB*" is such that sharing would be confined to certain workgroups or social groups (where individuals already know each other), privacy may not be as large a concern as it would be in large scale sharing.

GroupLens

In 1994, John Riedl and Paul Resnick developed a system that concerns of quality decisions for information retrieval as well as the burgeoning amount of information to be retrieved. That developed system is now known as "GroupLens." These GroupLens' developers designed an architecture structured around a server (nicknamed the "better bit bureau") by using Usenet newsgroups as the domain. That particular system collects ratings from individuals. Then produces predictions of quality for those individuals based on those ratings. According to a GroupLens flyer, the system "combines your opinions about articles you have already read with the opinions of others who have done likewise and gives you a personalized prediction for each unread news article. The prediction is on a scale from 1-5, and indicates to you how likely you are to find the article useful."

A key feature of GroupLens is its *open* architecture. This feature allows other researchers to create clients that work with GroupLens servers or to even replace those servers if improvements can be suggested.

The major limitation to further development is its bigness (Joe Konstan, University of Minnesota). With a potential 22 million Usenet users reading and rating articles, Usenet newsgroups constitute a huge database. Another concern is heterogeneity, and whether work performed in one domain can be carried over into another. A third matter pertains to startup; that is, whether folks will continue participating (recommending) if they do not obtain "instant gratification."

Do-I-Care

This system lets a user know proper time to revisit a favorite Web page and alerting a user to an interesting change on that page. These are the primary functions of Do-I-Care. In addition to collaborative filtering, Do-I-Care is concerned with the issue of re-discovery and the utility of machine learning in agents. Do-I-Care is a "sister project" to Mike Pazzani's Syskill and Webert discovery agent.

A Do-I-Care user trains an agent over time to look for the kinds of changes in which he or she would be interested. This training is achieved by a single user indicating that he or she "cares" or "doesn't care" about a certain item. The technology uses a simple Bayesian classifier and simple text parser that looks for key words. Agents can be cascaded, allowing collaborative use. An individual can use others' efforts for free.

One concern relates to privacy. Unlike some other systems, Do-I-Care involves the explicit rating of other people's work. That refers to an act, which in itself may be organizationally or socially problematic once those ratings are shared.

Pointers & Digests

Lotus is developing a form of attributed filtering which, according to Kate Ehrlich (Lotus), is "based on an implicit social contract we believe exists in small workgroups."

As Ehrlich explained, the familiarity which stems from belonging to a workgroup (and theoretically, from attributed filtering), provides two inherent advantages: (1) when one receives a recommendation, one can evaluate it against what is known about the other person; and (2) individuals are more likely to make recommendations to people they know.

Another contrast is the fact that Pointers supports those who send recommendations (in fact, only "recommenders," not receivers, require software); queries cannot be made against the system. For example, if a Notes user is reading a newswire database and comes across an article about librarians, that user can recommend the article by adding comments to a form and then mailing that form to one or more colleagues. The colleague then receives a semi-structured e-mail message containing the title of the article, the name of the database in which it was a hypertext link, and the name and comments of the recommender.

A key feature of Pointers is that it is not tied to a particular domain; i.e., any Notes database to which a particular user has access is a possible source for sending messages. Further, it is a "push" model, in the sense that it is pushing information "out" to other users (whether or not those users request it) by identifying sources and documents.

In fact, incorporating the "push" model of Pointers with the "pull" model of more conventional filtering and retrieval systems is an issue now under study.

Another Lotus system called *Information Digest*, is also under development. Information Digest provides lists of recommendations organized into sections. Several issues, however, require further analysis. For example, Digest is not as useful for on-demand searches unless there is a large enough pool of recommendations available.

Phoaks

Yet another filtering technique, one called Phoaks, uses frequency of mentioned data within Usenet news groups (e.g., how often people mention URLs; why they mention URLs). Will Hill and his colleagues at AT&T Research are now evaluating these data for their potential to recommend Web resources. According to Hill, their "bias" is the following: how far can they go without having to ask a user for any data?

One of Phoaks' primary advantages stems from its "one person, one vote" approach. Because the system will only accept one recommendation per individual, it prevents any single person from "spamming" the system with multiple recommendations.

Some of the issues now under study include: human interfaces to the social filtering data (i.e., what is useful and useable); privacy vs. connectedness design issues; dependence upon noncompetition; credit-assignment to recommenders, issues of credentials; interoperable systems (e.g., providing PICS server to Phoaks' data).

2.3 Collaborative Filtering Algorithms

This part will describe three prediction algorithms: a Pearson correlation-based method, the support vector method, and a scalable Pearson correlation-based method that uses clustering to improve scalability and accuracy.

Correlation-based Prediction

From Breese paper [Breese98], the correlation-based method is the most popular prediction technique in collaborative filtering applications. It was originally used in the GroupLens project [Resnick94]. The basic idea is to compute a user's predicted rating of an item as a weighted average of the ratings given to that item by other users. It assumes that the predicted vote of the active user for item j , $p_{a,j}$, is a weighted sum of the votes of the other users.

$$p_{a,j} = \bar{v}_a + k \sum_{i=1}^n w(a,i)(v_{i,j} - \bar{v}_i)$$

where n is the number of users in the collaborative filtering database with nonzero weights. k is a normalizing factor such that the absolute values of the weights sum to unity. The weights $w(a,i)$ can reflect distance, correlation, or similarity between each user i and the active user.

Rather than summing all of the users in the system to generate the prediction, the GroupLens algorithm only considers the "neighborhood" of users who are well correlated with the current users. This is more efficient, since the average is computed

over a much smaller set of values, and more accurate, since the votes of potentially large numbers of poorly correlated users do not affect the current user [Herlocker99].

An appropriate range for the weights is $[-1, 1]$. The GroupLens project, hence, used the Pearson Correlation coefficient to compute the weight for each user's contribution to the predicted vote which defined the correlation between users a and i as:

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}}$$

Advantages

The Pearson Correlation Coefficient algorithm has advantages that it is popular (and therefore a good test for any evaluation framework), intuitive, easy to understand, and relatively accurate [Breese98].

Disadvantages

Pearson Correlation Coefficient suffers from poor scalability for large numbers of users. This is because the set of all users must be searched for well-correlated neighbors on each prediction request.

Support Vector Method

An alternative approach to prediction of discrete ratings is to view the problem as a classification task. The goal is to use existing users to identify voting classes, and to use these classes as a basis for prediction [Vapnik98].

Formally, consider the following *optimal margin classification* problem for two separable classes. Suppose that a set of n -dimensional points $X = \{x_1, \dots, x_N\}$ with binary labels $\{L_1, \dots, L_N\}$, where the labels are either $+1$ or -1 is given. The goal is to find a hyperplane that maximizes the *margin* between the two classes, i.e., maximizes the minimum distance (measured with respect to the hyperplane) between the two classes. Vapnik [Vapnik98] showed that if such a hyperplane existed, then it was unique. Once this hyperplane is determined, new unlabelled points can be classified by identifying their positions with respect to this hyperplane.

The support vector method is a technique for constructing the optimal margin hyperplane. The technique computes a set of non-negative coefficients $\{a_1, \dots, a_N\}$ and a constant b such that the class of a new unlabelled point x can be determined by computing

$$C(x) = -b + \sum_{i=1}^N a_i L_i x_i \cdot x$$

where the sign of $C(x)$ identifies the class. This classifier is referred to as a support vector machine (SVM). In practice, many of the coefficients are zero, so only a fraction of the training samples needs to be kept in order to build the classifier. The

coefficients are determined by solving a quadratic programming problem. The sequential minimization optimization (SMO) technique described in J. Platt Paper [Platt99] can be used. The SMO is an iterative method for finding the coefficients. The asymptotic training time is $O(n^2)$ where n is the number of training samples.

To handle the multi-class case, binary classes are used as follows:

- Given r classes, train binary classifiers $C_1(x), \dots, C_r(x)$ for each class, where $C_i(x)$ separates the members of class i from all the others.
- On a new point x , assign it to the class whose classifier has the largest value.

The classification technique can be applied to movie prediction as follows:

- Each of the six values on voting scale (0 to 5 stars) is treated as a “class”.
- For each movie, construct a multi-class classifier that separated the users who voted for that movie, using each user's vector of votes as a “point” for training.

The cost of building the classifiers is expensive, however, in a real-time system, these classifiers can be built off-line. Prediction time is effectively constant since it depends only on the number of training samples used, which does not change once the classifier has been constructed.

One difficulty in using SVMs for the prediction task is that there can be a significant amount of missing data. The policy for missing data is to assign a slightly negative score to any non-existent vote in the vote vector. The actual points are taken to be the votes that are re-scaled to the interval of $[-.5, .5]$, so that the missing vote has a value of 0. Since there are a large number of missing votes, it is allowed to treat

each vote vector as a sparse vector. By treating each vote separately, significant computational savings during both training and prediction could be achieved.

An additional issue of the support vector approach is that hyperplanes may not be the right shape for separating complicated data. However, by Mercer's Theorem [Courant53], ones can replace the inner product $x_i \cdot x$ in the classifier formula with any kernel (i.e., symmetric positive definite) function. Applying such a kernel function in the data's native space automatically corresponds to performing hyperplane separation in some other space. This is essentially a form of feature selection. Thus, ones can use SVMs to find non-linear separators in the data space that correspond to optimal hyperplane separators in some other (usually higher dimensional) space. Ones use Gaussian kernels of the following form:

$$K(x_i, x) = \exp(-C|x_i - x|^2)$$

The constant $C = .001$ was empirically determined to be a good value based on experiments with subsets of the full data set.

Advantages

The primary advantage of the support vector approach is that ones are guaranteed an “optimal” solution in the sense that their solution is the mathematically optimal margin hyperplane. Their classifier is well principled though the quality of predictions depends on whether or not separation exists naturally in the data (or in some feature space of the data).

A computational advantage of this approach is that prediction times are fast, constant in the number of support vectors per item.

Disadvantages

The primary disadvantage is the costly training time, although prediction times are comparable to the other techniques considered. Another disadvantage is that ones are required to choose a default voting policy, since the method does not naturally accommodate the problem of missing data.



Correlation-based Prediction Using Clusters of Users

According to Danyel paper [Swami], one way to address the scalability problems with Pearson Correlation Coefficient is to reduce the number of users that are examined for similarity with the current user. This can be done by looking only at a sample of the full data set or by assembling the users into clusters. The clusters can then be used to generate representative users for subsequent predictions. Alternatively, the clusters could be used to determine likely neighbors for the current user.

The implementation of a cluster-based predictor (henceforth referred to as the "Clustered Pearson" predictor) uses Pearson correlations both to form the clusters and to compute predictions using the clusters. Clusters are formed using an iterative method that is similar to the well-known k-means algorithm. A true k-means algorithm would require a distance metric between users, but correlations do not provide such a metric.

The Clustered Pearson algorithm proceeds as follows:

1. Randomly select k users to serve as initial cluster centers
2. Assign all users to the "best" cluster. A user's best cluster is defined to be the cluster center that this user is best correlated.
3. Choose a new center for each cluster. The new center is the user who has the best overall correlation with all the other users in the cluster. The mean square correlation between a user and all of its other cluster members is used as the measure of "best".

4. Repeat steps 2 and 3 until the clusters have stabilized, i.e. no users are assigned to new clusters. (In practice, the clusters always stabilize. In the experiments, the number of clusters will be varied from 100 to 5,000, and clusters always stabilize within 10 iterations.)
5. Create a profile of each cluster by computing the average rating given to each item that is rated by members of that particular cluster. The profile is a composite "user" with those votes to represent the cluster.

As mentioned above, there are at least two ways to use the clusters to produce a prediction: treating the clusters as composite users and using the clusters for pre-screening likely neighbors. To use the latter approach, ones would first find the clusters whose profiles best correlate with the current user. Then search within those clusters to find the user's neighborhood. Then take the former approach, i.e., compute predictions by treating the cluster profiles as users and building a neighborhood out of the profiles. The Pearson Correlation Coefficient predictor is used to find the neighborhoods and generate the predictions from the set of cluster profiles.

Advantages

The primary advantage of Clustered Pearson over Pearson Correlation Coefficient is its improved scalability, since the entire user data set has been reduced to a potentially much smaller number of composite users. Furthermore, these composite users possess a higher vote density since each composite user is the aggregation of its member users' votes.

Disadvantages

However, the Clustered Pearson method suffers from a large off-line training time requirement. Furthermore, it requires a policy for initial cluster selection (using a random policy). Finally, there is a possible instability due to the randomness of clusters.



2.4 Empirical Evaluation of Collaborative Filtering Algorithms

Empirical Evaluation from Danyel paper [Swami]

With referral from Danyel paper [Swami] evaluation, they ran each of the predictors using 20%, 40%, 60%, and 80% of the *Eachmovie* (Details in section 1.4) data set for training, testing predictions on the remainder. They presented results for the cases when the predictor knows about 5 of a user's votes (relatively few), 20 votes (approximately the median number of votes), and all-but-one vote (namely, the vote being predicted). Danyel paper [Swami] presents the evaluation of the following collaborative filtering algorithms:

- Pearson Correlation Coefficient
- Support Vector Predictor
- Cluster Pearson

Evaluation of each above algorithms is provided comparing to their two base line predictors, By-User-Average predictor and the By-Movie-Average predictor, detailed as follows:

Pearson Correlation Coefficient

Accuracy

The Pearson Correlation Coefficient predictor achieves a mean absolute error of approximately 0.88, which is a significant improvement over the baseline predictors. Furthermore, the absolute error variance is lower than that of the baseline, suggesting predictions that are more reliable. Finally, the weighted mean values also improve on the baseline predictors, demonstrating an improvement in accuracy of off-

mean votes. These results are comparable to those of previous studies [Herlocker99, Breese98], which found the Pearson Correlation Coefficient predictor to be the current state-of-the-art algorithm on the *EachMovie* data set.

It was also observed that with a sufficient sampling of the data set (i.e., 40% or more), the accuracy of Pearson Correlation Coefficient predictions were relatively insensitive to the number of given votes. This is a desirable property in on-line systems where ones do not want to require users to enter a large number of votes before obtaining accurate predictions.

Another interesting observation made in Danyel paper [Swami] is that the mean error for the Pearson Correlation Coefficient predictor is approximately +0.3, suggesting a tendency to bias predictions toward the high side. This approach stands in contrast to the baseline predictors whose mean error is close to zero or slightly negative.

Performance

Pearson Correlation Coefficient is impractical for real-time systems due to the long prediction times. Even for the smallest sample sets that fit into memory, the predictor requires time in the order of 1 to several minutes to generate a prediction.

Support Vector Predictor

Accuracy

As concluded in Danyel paper [Swami], the support vector method did not perform as well as the other predictors. In the mean absolute error sense, it was not improved upon the baseline predictors except with a large sample set and a large number of users votes. For example, with 80% of the data set available for training in the all-but-one case, it achieved a mean absolute error of 1.04. There was a consistent trend of improvement as more votes were given.

The prediction result is hypothesized that the support vector predictions could be less accurate because it is not designed to have a functionality that treats missing data.

Although the mean absolute error was not improved upon the baseline predictors, the weighted mean metric was. However, the weighted mean of the support vector predictor is still worse than that of the Pearson Correlation Coefficient.

Performance

Danyel proposed training the support vector predictors off-line since training was a relatively expensive operation. Recall that a classifier must be constructed for each item under their proposed scheme. Due to time restrictions, Danyel opted to train the classifiers at predict time, and fixed the maximum number of training samples used in advance to 500 users. This was an empirical value that seemed to work well in initial experiments, and kept the training time roughly constant (in the order of a

minute). Prediction times are in the order of a 10 milliseconds, making real-time prediction possible.

Clustered Pearson

Accuracy

According to Danyel paper [Swami], Clustered Pearson obtained the best overall mean absolute error across all algorithms with 40% of the data set for training when given all-but-one votes. That overall mean absolute error was 0.77. The weighted mean correspondingly reflects improved accuracy of difficult ratings. The fact that this occurred at 40% of the data set suggests that there may be an optimal sampling size that is less than 100%, although larger sampling sizes were close to this value (mean absolute error up to .85 at 60% size). The exact location of a possible minimum absolute mean absolute error could be determined with additional experimentation at larger sampling sizes.

Interestingly, the Clustered Pearson algorithm was improved upon the accuracy of the Pearson Correlation Coefficient algorithm. The most likely reason is the increase in vote density. Specifically, a cluster profile has rated at least many times and usually more movies are rated than any single member of that cluster.

Performance

The Clustered Pearson prediction times are sufficiently fast for real-time implementation, taking in the order of a few milliseconds to generate a prediction. The training times are relatively large, but it is assumed that such training will be done off-line.

Empirical Evaluation from Breese paper [Breese98]

In Breese paper [Breese98], an evaluation of collaborative filtering algorithms had been done using *Eachmovie* database. The experiments were done by defining the following protocols:

- AllBut1: Randomly select one vote for each test user in the data set and try to predict its value, given that all the users have voted for all other votes.
- Given2: Randomly select 2 votes from each test user and try to predict the remaining votes.
- Given5: Randomly select 5 votes from each test user and try to predict the remaining votes.
- Given10: Randomly select 10 votes from each test user and try to predict the remaining votes.

The following algorithms were tested:

- CR+: Pearson correlation with inverse user frequency, default voting and case amplification extensions.
- VSIM: Vector similarity with inverse user frequency transformation
- BN: Bayesian network
- BC: Bayesian network clustering models
- POP: Most popular item's scores (as base line performance)

The analyses were evaluated by ANOVA with the Bonferroni procedure for multiple comparison statistics. In the following table, RD stands for Required Difference. The difference between any two scores in a column must be at least as big

as the value in the RD row in order to be considered statistically significant at the 90% confident level for the experiment as a whole.

Table 2.1: Performance comparison reported from Breese paper

Algorithm	Given2	Given5	Given10	AllBut1
CR+	41.60	42.33	41.46	23.16
VSIM	42.45	42.12	40.15	22.07
BC	38.06	36.68	34.98	21.38
BN	21.64	30.50	33.16	23.49
POP	30.80	28.90	28.01	13.94
RD	0.75	0.75	0.78	0.78

(The higher scores indicate better performance)

Their results showed that Bayesian networks with decision trees at each node and correlation methods were the best performing algorithms over the experiments they had run. The results showed that Bayesian network performed best under the AllBut1 protocol. However, Bayesian network performed worse in less data circumstance, i.e. Given10, Given5 and Given2 respectively. However, the vector similarity and clustering methods were competitive for some of these limited-data scenarios. As shown in the experimental results, correlation method and vector similarity worked well in all situations. Overall, correlation method did a slightly superior job than vector similarity and thus, correlation method was the top performer. Bayesian Network performed worse than any of the other algorithms on *Given protocols*. However, Baysian Network was the top performer and competitive with correlation for the AllBut1 protocol. In conclusion, correlation algorithm yielded high performance in all experiments, except in the *Given2* experiment. This indicated that correlation algorithm performed well when adequate data regarding the active case were given.

CHAPTER 3: DESIGN AND METHODOLOGY

This chapter describes the design and methodology of this study's probabilistic distance measure algorithm. When similarity of two persons is being considered, there are cases that one person does not give a preference that another person gives. The existing similarity measure algorithms usually skip this missing value and consider only the complete ones. In fact, it does not mean that nothing can be calculated from this missing value because we still know the probability that these two users are similar by considering the missing preferences together with the known preferences. This will give the better probability of similarity between two persons.

3.1 The Probabilistic Distance Measure Algorithm Design

This thesis presents a similarity measure, which can be applied to collaborative filtering system to get some similar people from its database to give recommendations to a user. We extend an algorithm based on probabilistic distance measure of Ha and Haddwy [Ha99], which concerns on pair-wise preferences comparison of two users. This distance measure approach defines conflict between two users as the average of probability that pairs of preferences are ranked differently. We present an exact algorithm for computing probabilistic distance when preferences of users are defined as numeric ratings. The algorithm calculates similarity between two users by using case based pair-wise comparison of all preferences.

In order to calculate a probabilistic distance measure, this proposed algorithm focuses on one "target user", which we are comparing to another one. We examine one pair of items for both users, no matter the score are provided or not. If we have complete scores for both items, i.e. both users give scores on these both items, when

they both prefer the same item, we define that they “agree” on this pair of items. On the other hand, if they prefer different items we define that they “disagree” on this pair of items. In the other case that we do not have complete scores for both items, we calculate the probability of agreement for those two users. Then we continue to examine on another pair of items until similarity probability are calculated from all item pairs. Finally, after calculating average value of the similarity probabilities, we have probabilistic similarity of two users (or probabilistic distance of two users from the fact that it is equal to 1 - probabilistic similarity).

The similarity probability calculations of two users are separated into eight cases based on their behavior of scoring. In each case, we determine the probability that these two users are “agree” by separating agreement criteria into sub cases. We define the maximum item score n and minimum score 1. For the user who does not assign the scores, we assume that he assigns the scores with a uniform distribution of $1 / n$ over all values. In addition, by defining two users as U_1, U_2 and two items as I_1, I_2 , the similarity probability calculations are shown as follows:

Let

$$(n-1)+(n-2)+\dots+1=\frac{(n-1)}{2}[(n-1)+1]=\frac{n(n-1)}{2} \dots\dots\dots \text{Equation 3.1}$$

and

$$\text{Probability of } x \text{ ways to be true from total of } y \text{ ways} = \frac{x}{y} \dots\dots\dots \text{Equation 3.2}$$

For an item that no score is given, a user has n ways to give a score, numbered from 1 to n . Respectively, there are 0 up to $n - 1$ ways to give the score of the other

item’s score lower (i.e. there are $n - 1$ down to 0 ways to give the other item’s score higher). By adding up those ways, from equation 3.1, we have a total of $\frac{n(n-1)}{2}$ ways. Since all the combination of voting is n^2 ways, from Equation 3.2, we have:

Probability that a user gives two item scores and one score is higher (or lower) than

the other = $\frac{n-1}{2n}$

..... Definition 3.1



Case 1: No score is given

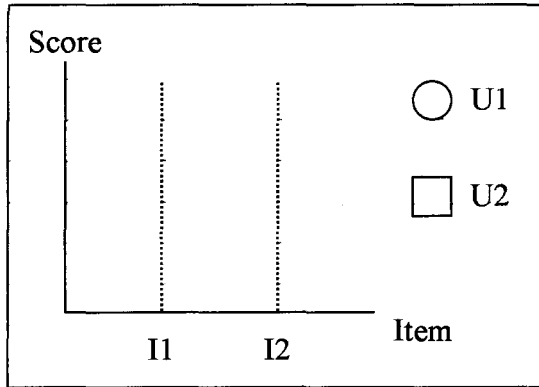


Figure 3.1: Probability calculation when no score is given

a. **Probability that both users prefer I_1**

From definition 3.1, probability that U_1 prefer I_1 is $\frac{(n-1)}{2n}$ and probability that U_2 prefer I_1 is $\frac{(n-1)}{2n}$. The probability that both users prefer I_1 becomes

$$\left(\frac{n-1}{2n}\right)^2$$

b. **Probability that both users prefer I_2**

From definition 3.1, probability that U_1 prefer I_2 is $\frac{(n-1)}{2n}$ and probability that U_2 prefer I_2 is $\frac{(n-1)}{2n}$. The probability that both users prefer I_2 becomes

$$\left(\frac{n-1}{2n}\right)^2$$

c. Probability that both users like both items equally

For an item, the score is numbered from 1 to n . Therefore, there are n ways that two users can give equal scores for this item. Since all the combination of voting is n^2 ways, the probability becomes $\frac{1}{n}$. For two items, probability becomes

$$\frac{1}{n^2}$$

By combining all of these distinct sub cases of the probability, if both users give no vote, we have similarity probability as

$$\text{S.P.} = \frac{2 + (n-1)^2}{2n^2}$$



Case 2: A score s is given for item i

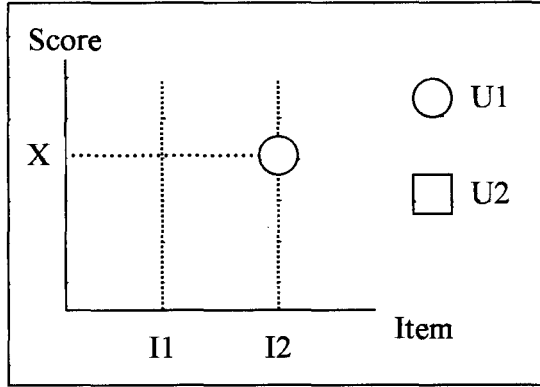


Figure 3.2: Probability calculation when only one score is given

a. Probability that both users prefer item i

For the user who gave a score s , there are $s - 1$ ways to give the other item score lower. From all of n possible ways to vote a score, the probability becomes

$\frac{s-1}{n}$. For the other user who gives no score, from definition 3.1, the probability that

a score is higher is $\frac{n-1}{2n}$. Combining these two users, the probability becomes

$$\frac{(s-1)(n-1)}{2n^2}$$

b. Probability that both users prefer the other item

For the user who gave a score s , there are $n - s$ ways to give the other item score higher. From all of n possible ways to vote a score, the probability becomes

$\frac{n-s}{n}$. For the other user who gives no score, from definition 3.1, the probability that

a score is higher is $\frac{n-1}{2n}$. Combining these two users, the probability becomes

$$\frac{(n-s)(n-1)}{2n^2}$$

c. Probability that both users like both items equally

For the item that the user votes score s , there is only 1 way to give score equally for the other item, from total of n ways. For the item that no one gives score, there are n ways to vote the score equally from the total of n^2 ways. So, the probability becomes

$$\frac{1}{n^2}$$

By combining all of these distinct sub cases of the probability, we have

$$\frac{2 + (n-s)(n-1) + (s-1)(n-1)}{2n^2}. \text{ Thus, if one score is given, we have the similarity}$$

probability as

$$\text{S.P.} = \frac{2 + (n-1)^2}{2n^2}$$

Case 3: One user gives score for two items

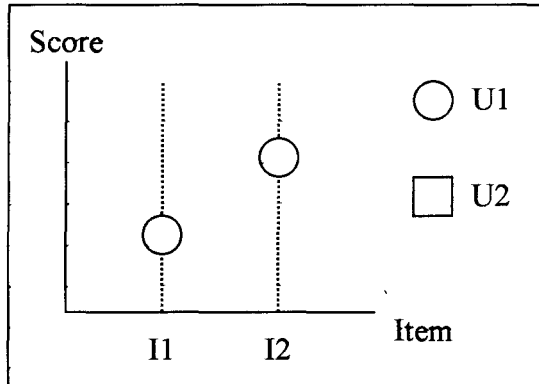


Figure 3.3: Probability calculation when one user gives scores for two items

- a. If the given scores are not equal to each other

From definition 3.1, for the user with no score given to give both item scores and prefer the same item as the other user, we have similarity probability as

$$\frac{n-1}{2n}$$

- b. If the given scores are equal

For the user who gives no score, there are n ways from the total of n^2 ways to give equal scores. Thus, the similarity probability becomes

$$\frac{1}{n}$$

By combining all of these distinct sub cases of the probability, if a user gives scores for two items, we have the similarity probability as

$$\text{S.P.} = \begin{cases} \text{The given scores are not equal : } \frac{(n-1)}{2n} \\ \text{The given scores are equal : } \frac{1}{n} \end{cases}$$



Case 4: Both users give a score on different item

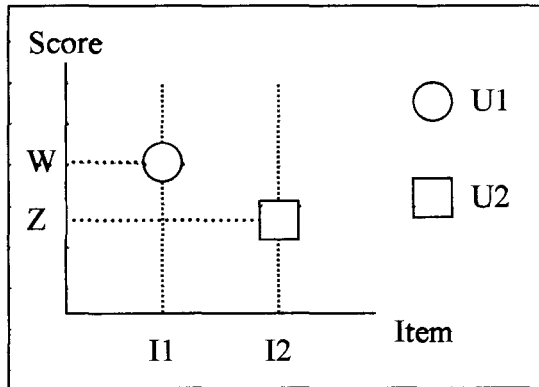


Figure 3.4: Probability calculation when both users give a score on different items

a. Probability that both users prefer I_1

For the user who gave I_1 score w , there are $w - 1$ ways to give I_2 score lower.

The probability that U_1 prefers I_1 becomes $\frac{w-1}{n}$. For the other user who gave I_2 score

z , there are $n - z$ ways to prefer I_1 , the probability becomes $\frac{n-z}{n}$. Considering two

users, the probability becomes

$$\frac{(w-1)(n-z)}{n^2}$$

b. Probability that both users prefer I_2

For the user who gave I_1 score w , there are $n - w$ ways to prefer I_2 . For the

other user who gave I_2 score z , there are $z - 1$ ways to prefer I_2 . The probability becomes

$$\frac{(n-w)(z-1)}{n^2}$$

c. Probability that both users like both items equally

From all of n^2 ways to vote the missing scores, there is only one way to give the other score so that the item scores are equal. The probability becomes

$$\frac{1}{n^2}$$

By combining all of these distinct sub cases of the probability, if both users give a score on different items, we have

$$\text{S.P.} = \frac{1 + (w-1)(n-z) + (z-1)(n-w)}{n^2}$$

where

w is a given score of I_1

z is a given score of I_2



Case 5: Both users give a item score on the same item

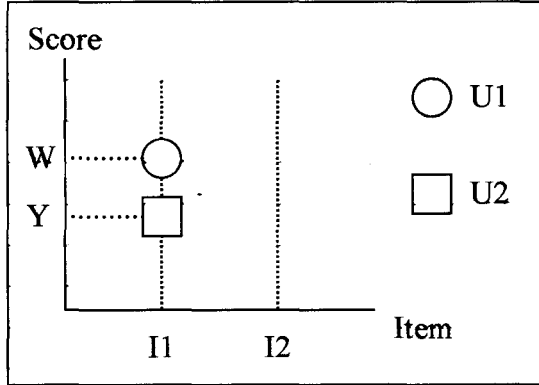


Figure 3.5: Probability calculation when both users give a score on the same item

a. Probability that both users prefer the first item

For the user who gave I_1 score w , there are $w - 1$ ways to prefer I_1 . For the other user who gave I_1 score y , there are $y - 1$ ways to prefer I_1 . The probability becomes

$$\frac{(w-1)(y-1)}{n^2}$$

b. Probability that both users prefer the second item

For the user who gave I_1 score w , there are $n - w$ ways to prefer I_2 . For the other user who gave I_1 score y , there are $n - y$ ways to prefer I_2 . The probability becomes

$$\frac{(n-w)(n-y)}{n^2}$$

c. Probability that both users like both items equally

From all of n^2 ways to vote the missing scores, there is only 1 way to give the other score so that the item scores are equal. The probability becomes

$$\frac{1}{n^2}$$

By combining all of these distinct sub cases of the probability, if both users give a score on the same item, we have

$$\text{S.P.} = \frac{1 + (w-1)(y-1) + (n-y)(n-w)}{n^2}$$

where

w is a given score of I_1

y is the other given score of I_1



Case 6: Only one of I_1 score is missing

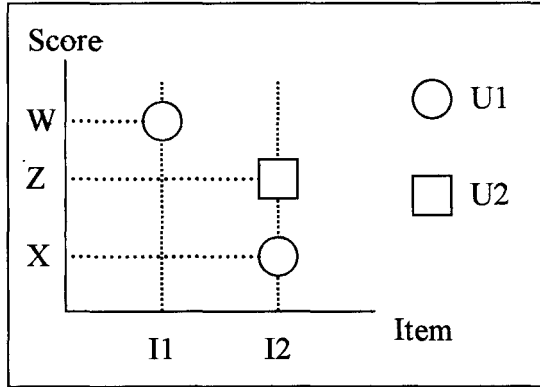


Figure 3.6: Probability calculation when only one of I_1 score is missing

- a. If score of $I_1 < I_2$ for a user who gives two scores

For the user who gave only I_2 score z , there are $z - 1$ ways to prefer I_2 . The probability becomes

$$\frac{z - 1}{n}$$

- b. If score of $I_1 > I_2$ for a user who gives two scores

For the user who gave only I_2 score z , there are $n - z$ ways to prefer I_1 . The probability becomes

$$\frac{n - z}{n}$$

c. If score of $I_1 = I_2$ for a user who gives two scores

From all of n ways to vote the missing scores, there is only 1 way to give the only missing score so that the scores are equal. The probability becomes

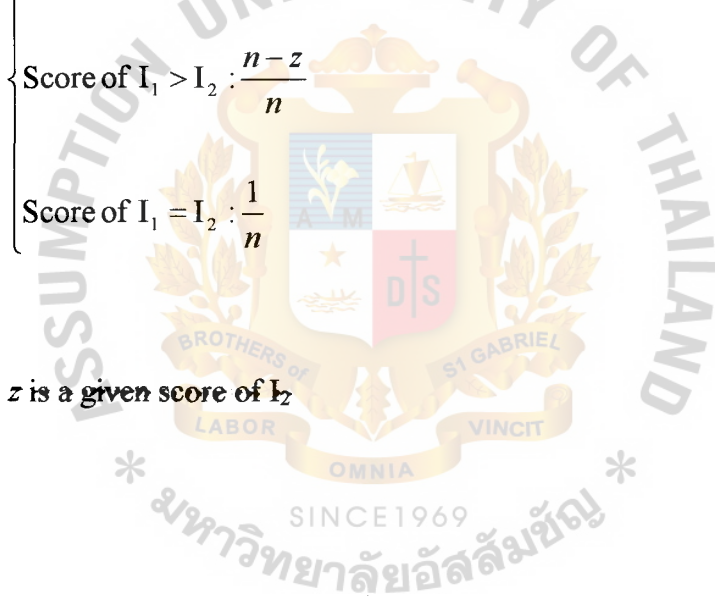
$$\frac{1}{n}$$

By combining all of these distinct sub cases of the probability, if only one of I_1 score is missing is

$$\text{S.P.} = \begin{cases} \text{Score of } I_1 < I_2 : \frac{z-1}{n} \\ \text{Score of } I_1 > I_2 : \frac{n-z}{n} \\ \text{Score of } I_1 = I_2 : \frac{1}{n} \end{cases}$$

where

z is a given score of I_2



Case 7: Only one of I_2 score is missing

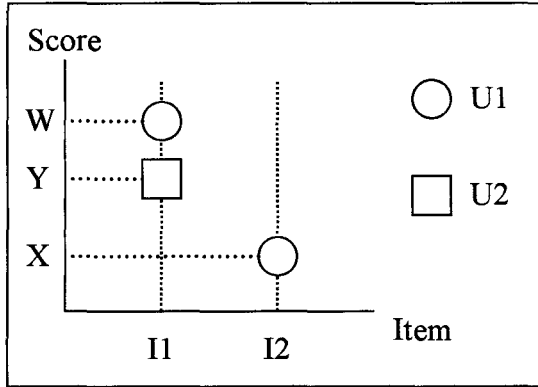


Figure 3.7: Probability calculation when only one of I_2 score is missing

- a. If score of $I_1 < I_2$ for a user who gives two scores

For the user who gave only I_1 score y , there are $n - y$ ways to prefer I_2 . The probability becomes

$$\frac{n - y}{n}$$

- b. If score of $I_1 > I_2$ for a user who gives two scores

For the user who gave only I_1 score y , there are $y - 1$ ways to prefer I_1 . The probability becomes

$$\frac{y - 1}{n}$$

c. If score of $I_1 = I_2$ for a user who gives two scores

From all of n ways to vote the missing scores, there is only 1 way to give the only missing score so that the scores are equal. The probability becomes


$$\frac{1}{n}$$

By combining all of these distinct sub cases of the probability, if only one of I_2 score is missing, we have

$$\text{S.P.} = \begin{cases} \text{Score of } I_1 < I_2 : \frac{n-y}{n} \\ \text{Score of } I_1 > I_2 : \frac{y-1}{n} \\ \text{Score of } I_1 = I_2 : \frac{1}{n} \end{cases}$$

where

y is a given score of I_1



Case 8: All scores are given

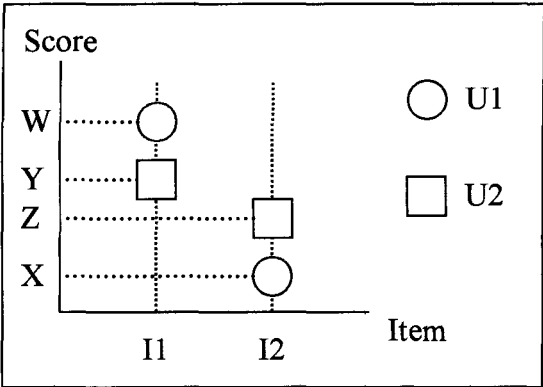


Figure 3.8: Probability calculation when all item scores are given

a. If both users prefer the same item

If both users prefer the same item or a user give two scores equally and the other user also give two scores equally, the similarity probability of these two users is 1.

b. If both users prefer different items

If both users prefer different item or a user equally like two items but the other user gives different score, the similarity probability of these two users is 0.

By combining all of these distinct sub cases of the probability, if all scores are given, we have

$$\text{S.P.} = \begin{cases} \text{If both user prefer the same item : 1} \\ \text{Otherwise : 0} \end{cases}$$

3.2 Metrics Qualification

This study's similarity probability that no prediction for user's score are made is as following:

$$\begin{cases} \text{If both user prefer the same item : 1} \\ \text{Otherwise : 0} \end{cases}$$

In addition, with common sense of probability, it can derive distance as following:

$$\text{Probabilistic distance} = 1 - \text{Similarity Probability}$$

With referral from Vu Ha paper [Ha99], they showed that the probabilistic distance measure on complete preference structures is a metric (and it is not a metric on partial preference structures).

For explanations of metric specifications, the following properties must hold for a distance measure to achieve a metric. (More details in section 1.2, Similarity Theories)

- **Self-similarity**

Distance between any two identical orders must be the same.

- **Minimality**

Distance between any pair of orders must be greater than or equal to the distance of any two identical orders.

- **Symmetrical**

Distance between orders A and B must be the same as distance of orders B and A .

- **Triangle inequality**

For any orders A , B and C , distance of A , C must be less than or equal to summation of distance of A , B with distance of B , C .

3.3 Experiment Methodology

This part describes the methodology used in this study. The software experiment method was used to investigate the efficiency of this study's probabilistic distance measure algorithm. The performance outcome will be compared with the other widely used method, Pearson correlation coefficient (Detailed information is mentioned on section 2.3 and section 2.4). Pearson correlation coefficient first appeared in the published literature in the context of Grouplens project as basis for the weights [Resnick94]. It is chosen to be the base comparison for this study because it achieves a good performance and it is well known by people in this field of study [Swami, Breese98, Herlocker99].

This study uses movie database from *Eachmovie* as experimental data (Details is provided in section 1.4). We assume that the probability that a person assigns a given rating to a movie is uniform over all rating values.

In order to evaluate the data size affects, ten sets of two data sizes are randomly picked and will be evaluated as the following:

- Five data sets of 500 users
- Five data sets of 2,000 users

The 500 users data sets represents the lower data size of users and 2,000 users data set represents the higher data size of users.

For each size of the data, five data sets are randomly picked and categorized to evaluate the data scaling affects as the following:

- Set of 10 to 50 movie votes
- Set of 50 to 100 movie votes
- Set of 100 to 150 movie votes
- Set of 150 to 250 movie votes
- Set of 10 to 250 movie votes

Low user preferences to high user preferences and mixed user preferences are thoroughly tested to evaluate the behavior of the similarity measure algorithm.

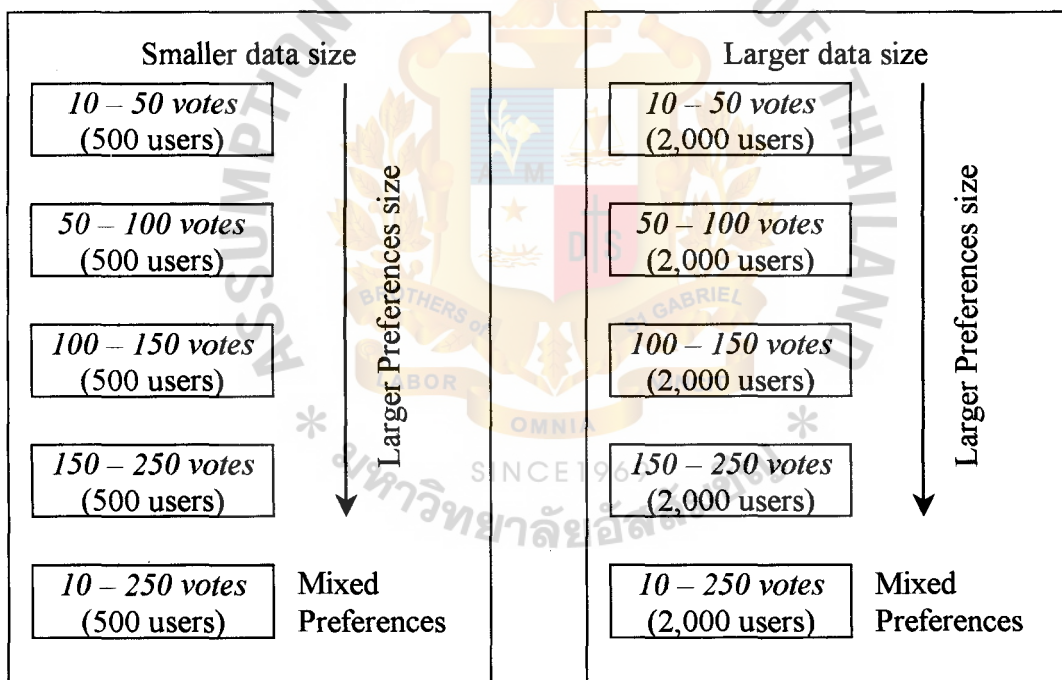


Figure 3.9: Experimental data sets

Two similarity measurements have been applied and evaluated as in the following:

1. Our Probabilistic Distance Measure algorithm
2. Pearson Correlation Coefficient

Two prediction algorithms have been used as following:

1. Average weighted sum (AWS): The simple prediction technique uses average of votes from other similar users. The average weighted sum extends the technique by multiplying the similarity of the user to his vote then sum up from all selected similar users and calculates the average value. The idea for this average weighted sum is to trust a user with more similarity than the less one. For example, If a user with similarity 0.5 says that the movie score is 6 and another user with similarity 0.3 says that the score is 4, then average weighted sum is $(0.5*6 + 0.3*4) / 0.8 = 5.25$ (the movie score is more likely to be 6 than 4)
2. Correlation based or Product Moment algorithm: One of the most popular prediction technique in collaborative filtering applications [Swami, Breese98]. It was originally used in the GroupLens project [Resnick94]. The basic idea is to compute a user's predicted rating of an item as a weighted average of the ratings given to that item by other users. It defines that the predicted vote of the active user a for item j , $p_{a,j}$, is a weighted sum of the votes of the other users:

$$p_{a,j} = \bar{v}_a + k \sum_{i=1}^n w(a,i)(v_{i,j} - \bar{v}_i)$$

where

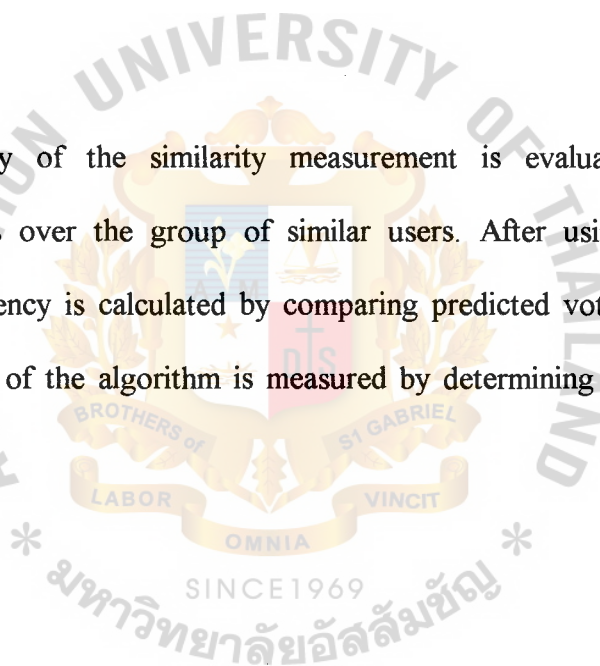
n is the number of users

k is a normalizing factor

$w(a,i)$ is similarity between user a and i

In each group of data sets, the experimental software randomly picks some test users (100 users for 500 users data sets, 500 users for 2,000 users data sets). Those test users are selected to do the experiment one by one. First, the software removes some of movie votes from the selected user (the experiment begins with removing 10%, after the performance has been tested, then try to remove 20% and 30%), keeping the original value. Then, similarity measurement is applied on the selected user over all other users to get a group of the most similar users (begins with 10% of all users, after the performance has been tested, increase to 20% and repeatedly do to the 100%).

The efficiency of the similarity measurement is evaluated by applying prediction algorithms over the group of similar users. After using the prediction algorithms, the efficiency is calculated by comparing predicted votes to the original votes. The reliability of the algorithm is measured by determining variance of those prediction errors.



CHAPTER 4: ALGORITHM AND DATA STRUCTURE

In this chapter, this study first describes the overall architecture of performance experimental program. Then continue on discussing the main algorithms and data structures used in each module

Architecture Overview

The performance experimental program obtains movie preferences from *Eachmovie* database. The overall algorithms are listed as follows:

1. The program randomly selects a number of users to test. From the test users, one by one, the program selects a user and calls his preference structure *Prototype*.
2. Cut some movie scores out from *Prototype*, treat it as if the scores are missing and try to predict.
3. Compute similarity of each user and find some of the most similar users who have vote on the movie scores it tries to predict. The program is implemented to do two methods, probabilistic distance measure and Pearson correlation coefficient in order to compare the performance results.
4. Predict the movie scores. When these movie scores are predicted, the prediction results are being compared to the true value and the accuracy performance is generated.
5. Write the performance result.

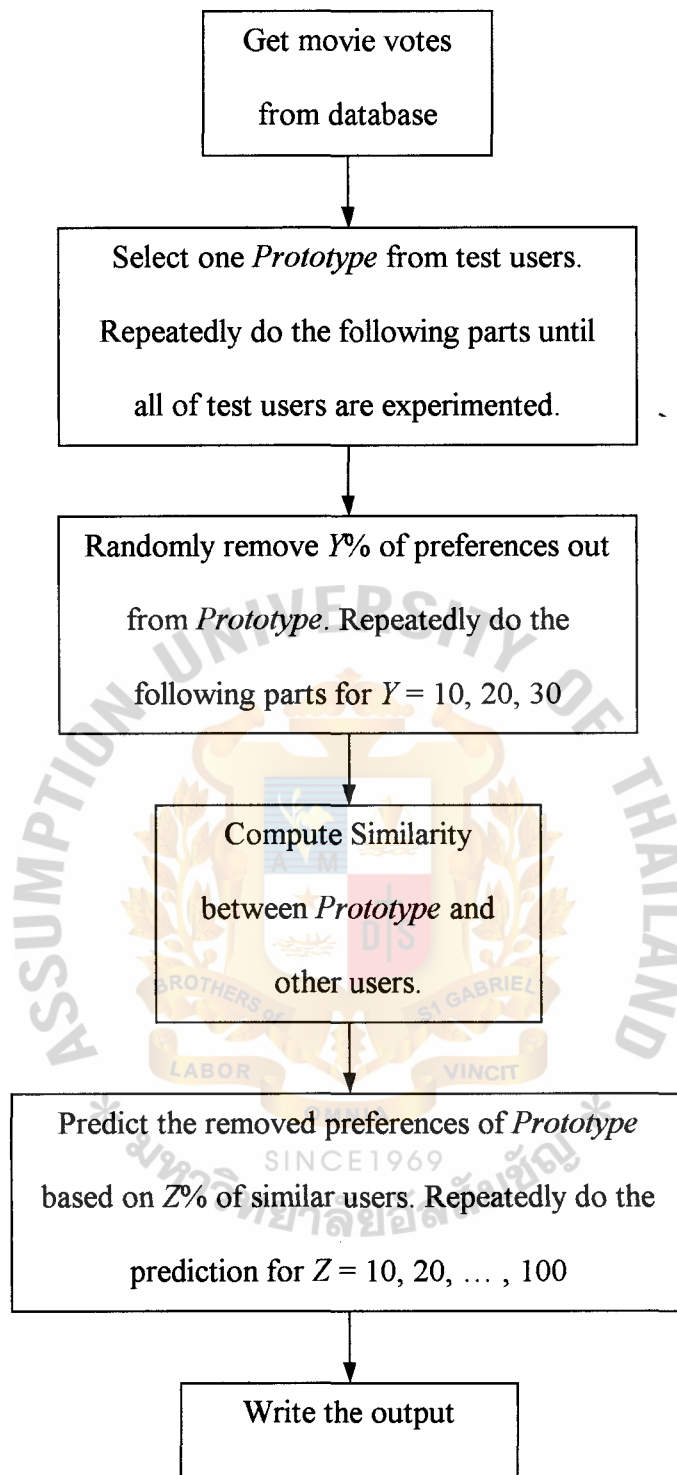


Figure 4.1: Overall algorithm of experimental program

Algorithm and Data structure of Modules

Part 1: Data Input Module

This part deals with the way to gather all data from database and keep it for further experimental phase.

Algorithm

From *Eachmovie* database, the data is read and ripping to be only the first three fields, which are user ID, movie ID, and score of movie. In this module, the movie preferences for a user are read and stored. The total number of movie votes are counted and put on together. Then the completely generated data is given back to caller.

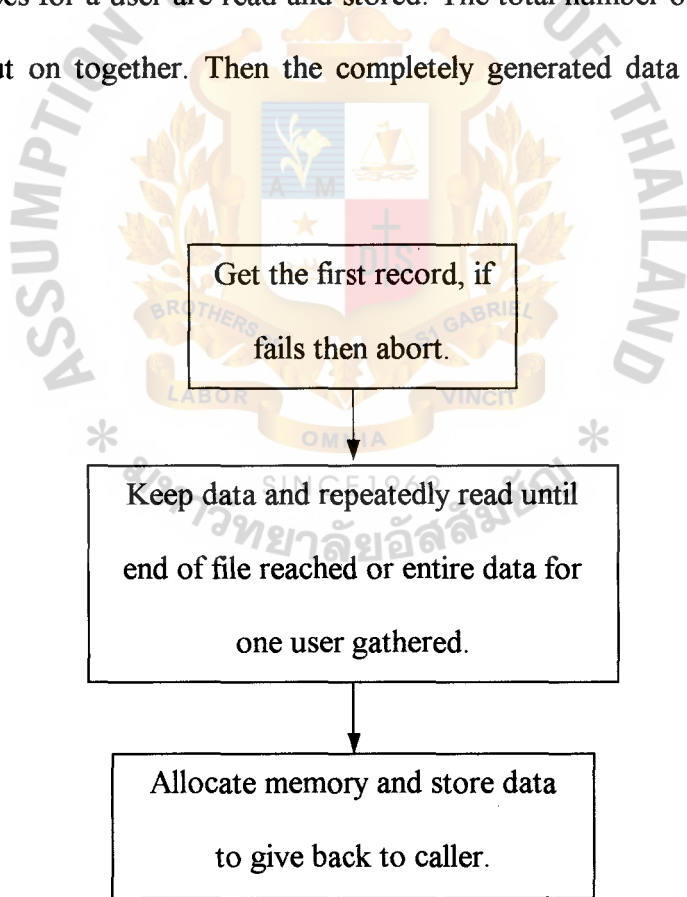


Figure 4.2: Algorithm of data input nodule

Data Structure

For latter experiment processes, link lists are used to keep user ID, number of movie votes, and a link to movie ID and movie score array. (Average value of voted movie scores will be used later)

Link List: *Profile*

User ID	Count	Average	Info *vote	Profile *next
---------	-------	---------	------------	---------------

User ID: The unique integer for specifying user

Count: Number of voted movies for this user

Average: Average value of voted movie scores

Info *vote: Pointer to Info array

Profile *next: Pointer to next user Profile

Array: *Info*

Enable	Movie ID	Score
--------	----------	-------

Enable: A Boolean to specify whether this movie score is removed or not.

Movie ID: The unique integer to distinguish the movies

Score: Voted score for the movie

Part 2: Similarity Measure Module

This part implements the study's probabilistic distance measure algorithm along with Pearson correlation coefficient to compare the prediction result later.

Probabilistic Distance Measure Algorithm

Its function is to measure agreement of every movie votes pair between two users. Firstly, the data has to be rearranged and sorted by movie. For any of movie that has been voted for either of user, put it to table array that composed of movie ID, the first user's score and the second user's score. Then every pair of the movie scores combination are picked. Considering two users giving two movie scores, it is defined that

w as first user's first movie score

x as first user's second movie score

y as second user's first movie score

z as second user's second movie score

The program encodes movie scores votes (w, x, y, z) into four bit *flag*. The *flag* will encode the behavior of movie scores votes into 16 cases, from binary 0000 ($w = 0, x = 0, y = 0, z = 0$) to 1111 ($w = 1, x = 1, y = 1, z = 1$). Next, based on *flag*, the calculation of similarity measurement is separated corresponding to the *flag*.

Table 4.1: Encoding table for two users' preferences

<i>w</i>	<i>x</i>	<i>y</i>	<i>z</i>	<i>flag</i>	Calculation to do (from Section 3.1 PDM Design)
0	0	0	0	0	Case 1: Both users give no score
0	0	0	1	1	Case 2: One user give one score
0	0	1	0	2	Case 2: One user give one score
0	0	1	1	3	Case 3: One user give two scores
0	1	0	0	4	Case 2: One user give one score
0	1	0	1	5	Case 5: Both users give one score on the same movie
0	1	1	0	6	Case 4: Both users give one score on different movie
0	1	1	1	7	Case 6: Both users gives score missing only one first movie
1	0	0	0	8	Case 2: One user give one score
1	0	0	1	9	Case 4: Both users give one score on different movie
1	0	1	0	10	Case 5: Both users give one score on the same movie
1	0	1	1	11	Case 7: Both users gives score missing only one second movie
1	1	0	0	12	Case 3: One user give two movie scores
1	1	0	1	13	Case 6: Both users gives score missing only one first movie
1	1	1	0	14	Case 7: Both users gives score missing only one second movie
1	1	1	1	15	Case 8: All scores are given from both users

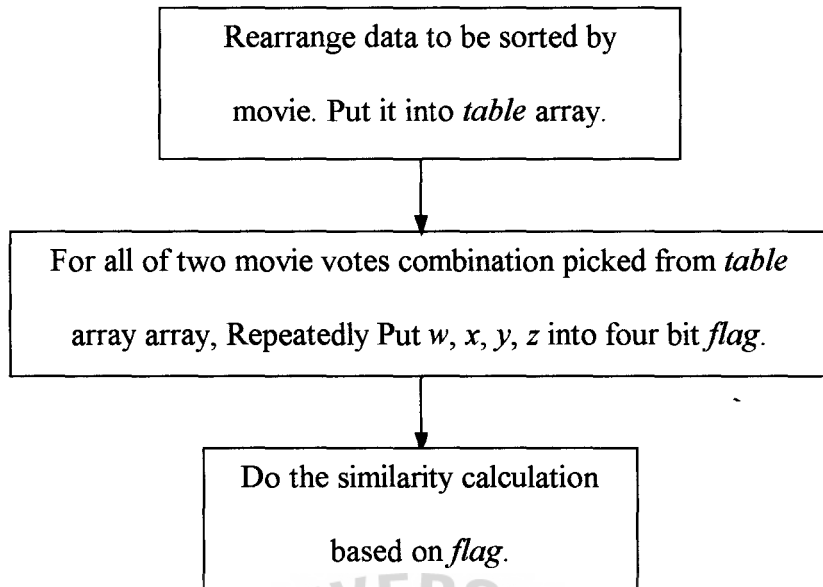


Figure 4.3: Probabilistic Distance Measure Algorithm

Data Structure

Link List: *Profile*

User ID	Count	Average	<i>Info *vote</i>	<i>Profile *next</i>
---------	-------	---------	-------------------	----------------------

User ID: The unique integer for specifying user

Count: Number of voted movies for this user

Average: Average value of voted movie scores

*Info *vote*: Pointer to Info array

*Profile *next*: Pointer to next user Profile

Array: *Info*

Enable	Movie ID	Score
--------	----------	-------

Enable: a flag to specify whether this movie score has to be removed or not.

Movie ID: The unique integer for specifying movie

Score: Voted score for the movie

Array: *table*

Movie ID	Score1	Score2
----------	--------	--------

Movie ID: The unique integer for specifying movie

Score1: First user's score for the movie

Score2: Second user's score for the movie

4 Bit data: *flag*

w	x	y	z
---	---	---	---

w: First user's first movie score

x: First user's second movie score

y: Second user's first movie score

z: Second user's second movie score

Pearson Correlation Coefficient Algorithm

Pearson correlation coefficient was defined the correlation between users a and i as:

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}}$$

where the summations over j are over the items for which both users a and i have recorded votes.

Firstly, the program has to calculate the average score of two users a and i . it has to rearrange the data to be sorted by movie. For any of movie that has been voted for either of user, put it to table array that composed of movie ID, first user's score and second user's score. Then we pick every pair of movie scores combination. Then for all movies j that both users a and i have a vote for it, does the calculation by the equation above.

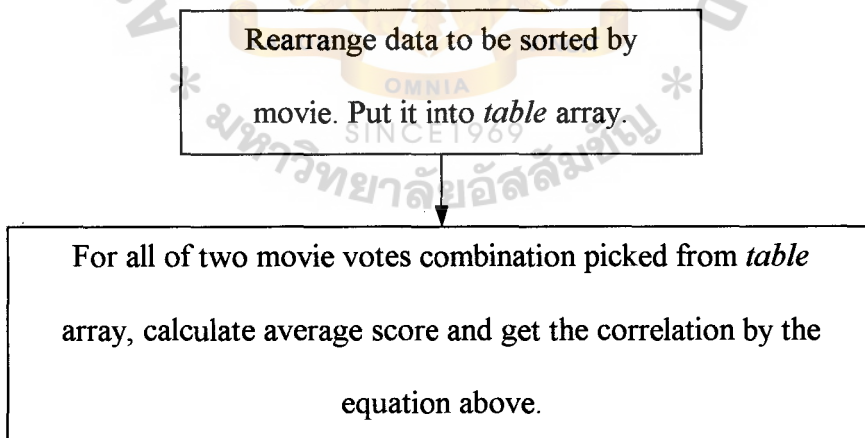


Figure 4.4: Pearson correlation coefficient algorithm

Data Structure

Link List: *Profile*

User ID	Count	Average	Info *vote	Profile *next
---------	-------	---------	------------	---------------

User ID: The unique integer for specifying user

Count: Number of voted movies for this user

Average: Average value of voted movie scores

Info *vote: Pointer to Info array

Profile *next: Pointer to next user Profile

Array: *Info*

Enable	Movie ID	Score
--------	----------	-------

Enable: a flag to specify whether this movie score has to be removed or not.

Movie ID: The unique integer for specifying movie

Score: Voted score for the movie

Array: *table*

Movie ID	Score1	Score2
----------	--------	--------

Movie ID: The unique integer for specifying movie

Score1: First user's score for the movie

Score2: Second user's score for the movie

Part 3: Prediction Experiment Module

This section implements methods to predict movie score of a user. This prediction experiment will show efficiency of similarity measurement from the fact that the better similarity measurement it uses, the better result of prediction it should gain. The whole algorithm begins with randomly picking up a user and calls that user a candidate. Then cut some movie votes off and keep the original value. Next, for this altered candidate user, does the similarity comparison with all other users and begin the prediction algorithms. Compares the prediction outcome with the original votes and keeps the result. At last, restores the value of the cut off movie votes and pick up another user to do the experiment all over again.

Cutting off Movie votes Algorithm

The program cuts off some movie votes from a specific user to do experiment in latter part by predicting the missing votes and compare the result with original value that it cut off. The whole algorithm is to randomly change enable flag of movie to zero for the specified percentage amount.

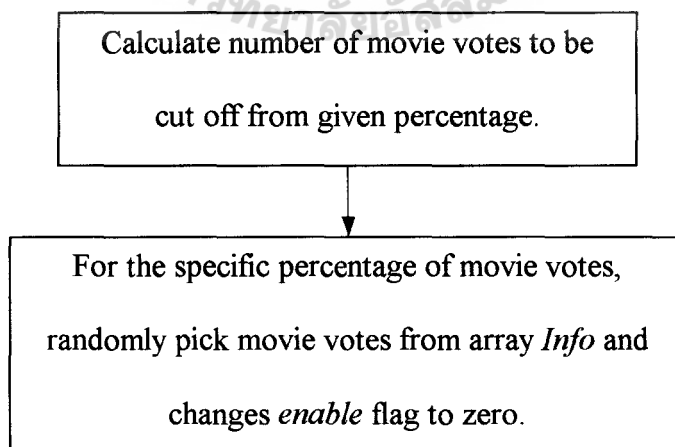


Figure 4.5: Prediction Experiment Module

Data StructureLink List: *Profile*

User ID	Count	Average	<i>Info *vote</i>	<i>Profile *next</i>
---------	-------	---------	-------------------	----------------------

User ID: The unique integer for specifying user

Count: Number of voted movies for this user

Average: Average value of voted movie scores

*Info *vote*: Pointer to Info array*Profile *next*: Pointer to next user ProfileArray: *Info*

Enable	Movie ID	Score
--------	----------	-------

Enable: a flag to specify whether this movie score has to be removed or not.

Movie ID: The unique integer for specifying movie

Score: Voted score for the movie

Movie Votes Filling Back Algorithm

After the experiment passed, the ripped off movie votes are filled back to original value. To do so, it changes the *enable* flag for all movie votes of a specific user back to non-zero value.

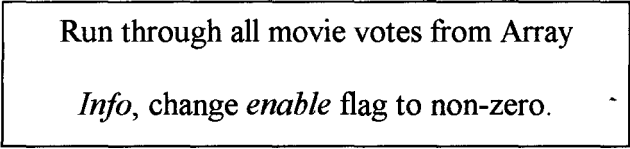


Figure 4.6: Movie votes filling back algorithm

Data Structure

Link List: *Profile*

User ID	Count	Average	<i>Info</i> *vote	<i>Profile</i> *next
---------	-------	---------	-------------------	----------------------

User ID: The unique integer for specifying user

Count: Number of voted movies for this user

Average: Average value of voted movie scores

Info *vote: Pointer to Info array

Profile *next: Pointer to next user Profile

Array: *Info*

Enable	Movie ID	Score
--------	----------	-------

Enable: a flag to specify whether this movie score has to be removed or not.

Movie ID: The unique integer for specifying movie

Score: Voted score for the movie

Prediction Based on Correlation Based Algorithm

The program finds the missing movie vote and tries to predict the score by using the equation:

$$p_{a,j} = \overline{v_a} + k \sum_{i=1}^n w(a,i)(v_{i,j} - \overline{v_i})$$

where n is the number of users that vote the same movie. k is a normalizing factor such that the absolute values of the weights sum to unity. The weights $w(a,i)$ is the similarity between each user i and the active user a .

Firstly, for the candidate user who it picks up to do this experiment, find movie votes that *Enable* flag are zero (these movie votes are assumed as missing and trying to predict these movie votes). Secondly, predicts the movie vote by running through all users and find users that also vote for this movie then uses the above equation to solve for prediction result. Then compares to the original value and keep the distance or error of prediction. Then for all other movie votes that enable flag are zero; repeatedly do the prediction again.

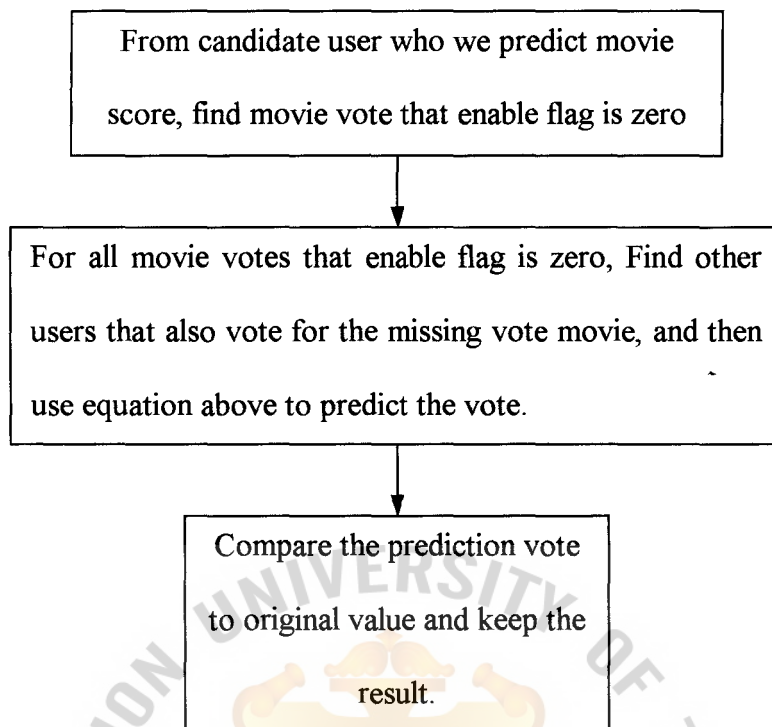


Figure 4.7: Prediction Based on Correlation Based Algorithm

Data StructureLink List: *Profile*

User ID	Count	Average	<i>Info *vote</i>	<i>Profile *next</i>
---------	-------	---------	-------------------	----------------------

User ID: The unique integer for specifying user

Count: Number of voted movies for this user

Average: Average value of voted movie scores

*Info *vote*: Pointer to Info array*Profile *next*: Pointer to next user ProfileArray: *Info*

<i>Enable</i>	Movie ID	Score
---------------	----------	-------

Enable: A flag to indicate whether movie score is missing or not.

Movie ID: The unique integer for specifying movie

Score: Voted score for the movie

Array: *Similar*

Similarity	<i>Profile *user</i>
------------	----------------------

Similarity: Similarity of the associated user compared to the candidate movie.

*Profile *user*: The associated user

Prediction Based on Most Similarity Votes Algorithm

The program predicts the movie score base on some of the most similar users votes. Firstly, it picks up a user then finds movie votes that *Enable* flag are zero (these movie votes are assumed as missing and trying to predict these movie votes). Secondly, predicts the movie vote by getting all movie votes from the specified amount of most similar users and get the average value of the movie votes. Then compares to the original value and keeps the distance or error of prediction. Then for all other movie votes that enable flag are zero; repeatedly do the prediction again.

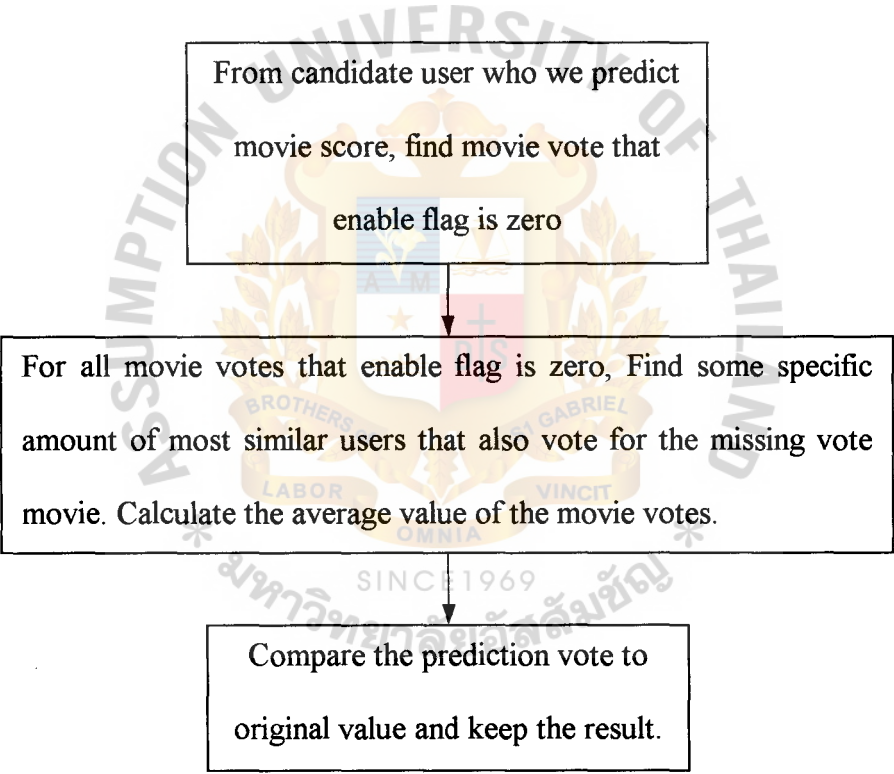


Figure 4.8: Prediction Based on Most Similarity Votes Algorithm

Data Structure

Link List: *Profile*

User ID	Count	Average	<i>Info *vote</i>	<i>Profile *next</i>
---------	-------	---------	-------------------	----------------------

User ID: The unique integer for specifying user

Count: Number of voted movies for this user

Average: Average value of voted movie scores

*Info *vote*: Pointer to Info array

*Profile *next*: Pointer to next user Profile

Array: *Info*

Enable	Movie ID	Score
--------	----------	-------

Enable: A flag to indicate whether movie score is missing or not.

Movie ID: The unique integer for specifying movie

Score: Voted score for the movie

Array: *Similar*

Similarity	<i>Profile *user</i>
------------	----------------------

Similarity: Similarity of the associated user compared to the candidate movie.

*Profile *user*: The associated user

Part 4: Output Module

This module task is to print out the results to the desire formatted pattern. From Prediction Experiment Module, all data has been saved in the floating point array that has dimension from the following:

3. Cut off percent
4. Methods of experiment
5. Number of users

The floating point array represents correlation between candidate user and all other users. We print out the result of one candidate user and randomly pick another candidate user to do the whole experiments all over again.



CHAPTER 5: EMPIRICAL ANALYSIS

5.1 Evaluation Criteria

By following the experiment methodology in Section 3.3, 500 random users from large data size sets (2,000 users) and 100 random users from small data size sets (500 users) are picked as test users. The tested users will exhaustively be tested with the evaluation criterions below:

This study runs the experiments to test the performance of two similarity measures:

- PDM: Probabilistic Distance Measure algorithm
- Pearson: Pearson Correlation Coefficient

The method to test similarity measure performance is to cut off some votes and use the prediction method based on other most similarity user's votes to predict the target votes. This study does three different cut off:

- 10%: Cut 10% of total votes off, represents more known preferences situation.
- 20%: Cut 20% of total votes off, represents average known preferences situation.
- 30%: Cut 30% of total votes off, represents less known preferences situation.

Two prediction algorithms are used in these experiments:

- Average weighted sum (AWS) algorithm (More details in section 3.3)
- Correlation based (Product Moment) algorithm (More details in section 3.3)

The exhaustive experiments are run to find the optimum value for number of similar users to use in prediction (10% to 100% of most similar users are tested). The experiments run by definition of the following protocols:

- 10% to 90%: Select users from 10% to 90% of most similarity to the active user and try to predict active user's preferences from the preferences of the selected users using average weighted sum method and correlation based algorithm.
- All: Utilizes all users to predict active user's preferences from the preferences of the selected users using average weighted sum method and correlation based algorithm.

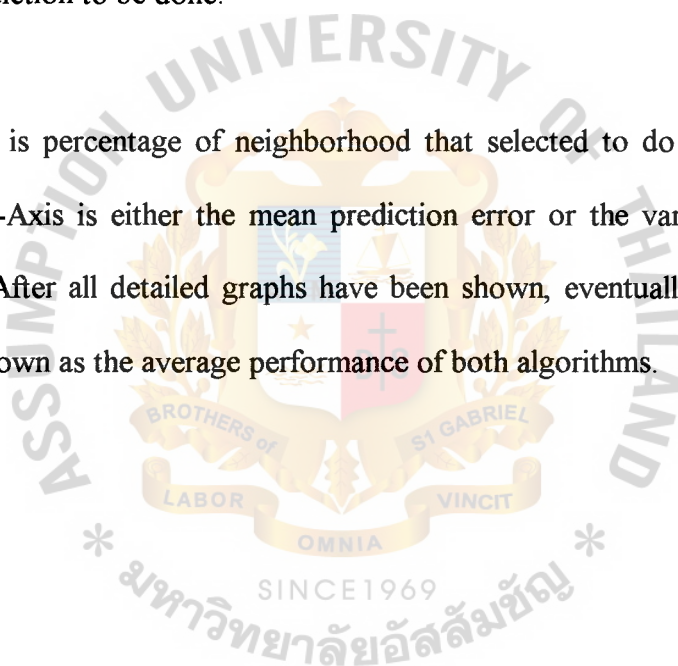
The experimental results are compared in two metrics:

- Mean Prediction error: The mean error of predicted votes shows how valid of the similarity measure. The lower number of mean prediction error indicates the better performance of similarity measure used.
- Variance of Prediction error: The variance of the prediction error shows reliability of the similarity measure. A good similarity measure should give reliable answers i.e. if similarity measure works for some users but completely wrong for others, it may not be preferred over another measure that has a slightly worse mean prediction error, but lower variance in the error.

5.2 Experiment Results

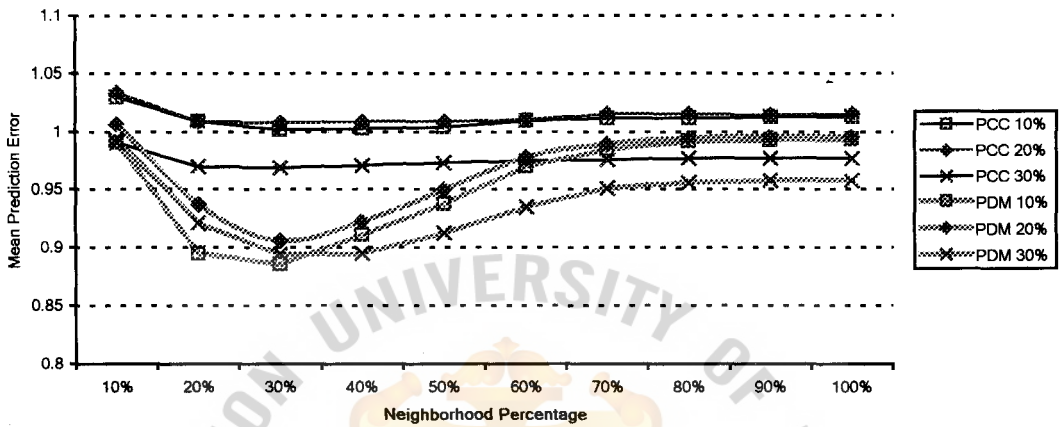
The experimental results of probabilistic distance measure algorithm and Correlation based algorithm are shown respectively from 2,000 users data sets results (Graph 5.1 to Graph 5.20) and 500 users data sets results (Graph 5.21 to Graph 5.40). PCC is denoted for Pearson Correlation Coefficient. PDM is Probabilistic Distance Measure. AWS is Average Weighted Sum prediction method. COR is Correlation based prediction. The number 10%, 20%, 30% is percentage of votes that has been cut off for the prediction to be done.

X-Axis is percentage of neighborhood that selected to do the prediction for active user. Y-Axis is either the mean prediction error or the variance, depends on which graph. After all detailed graphs have been shown, eventually, the summary of all graphs is shown as the average performance of both algorithms.



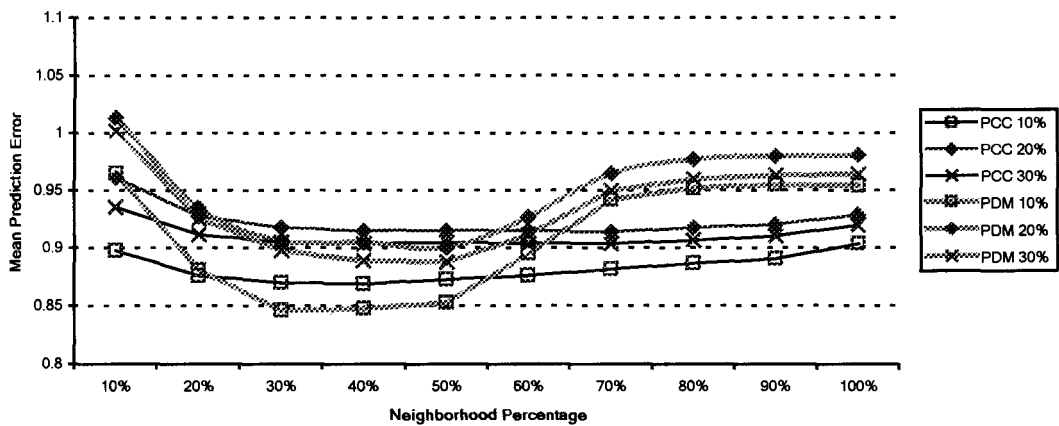
Graph 5.1: Mean error of users with 10 to 50 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.030	1.009	1.002	1.003	1.004	1.010	1.012	1.012	1.013	1.013
	PDM	0.991	0.895	0.886	0.911	0.938	0.970	0.985	0.992	0.993	0.994
20%	PCC	1.034	1.009	1.008	1.009	1.009	1.011	1.016	1.016	1.015	1.016
	PDM	1.007	0.937	0.906	0.923	0.950	0.978	0.990	0.995	0.996	0.996
30%	PCC	0.991	0.970	0.969	0.971	0.973	0.975	0.976	0.977	0.977	0.977
	PDM	0.993	0.921	0.894	0.895	0.913	0.935	0.951	0.956	0.958	0.958



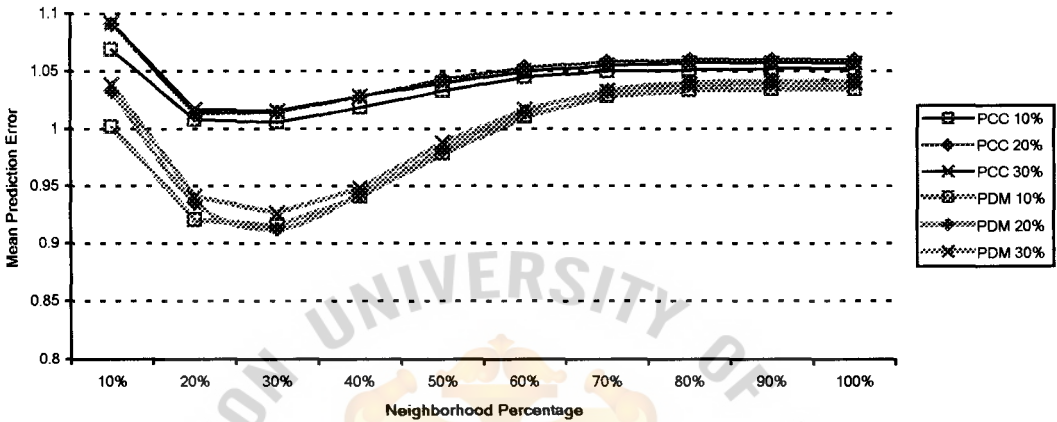
Graph 5.2: Mean error of users with 10 to 50 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.898	0.876	0.870	0.869	0.873	0.876	0.882	0.887	0.891	0.904
	PDM	0.965	0.881	0.846	0.848	0.853	0.895	0.942	0.952	0.955	0.954
20%	PCC	0.961	0.928	0.918	0.915	0.915	0.915	0.914	0.918	0.921	0.929
	PDM	1.014	0.935	0.907	0.905	0.901	0.927	0.965	0.977	0.980	0.981
30%	PCC	0.936	0.912	0.905	0.904	0.905	0.904	0.904	0.907	0.911	0.920
	PDM	1.002	0.926	0.898	0.889	0.888	0.912	0.950	0.960	0.963	0.964



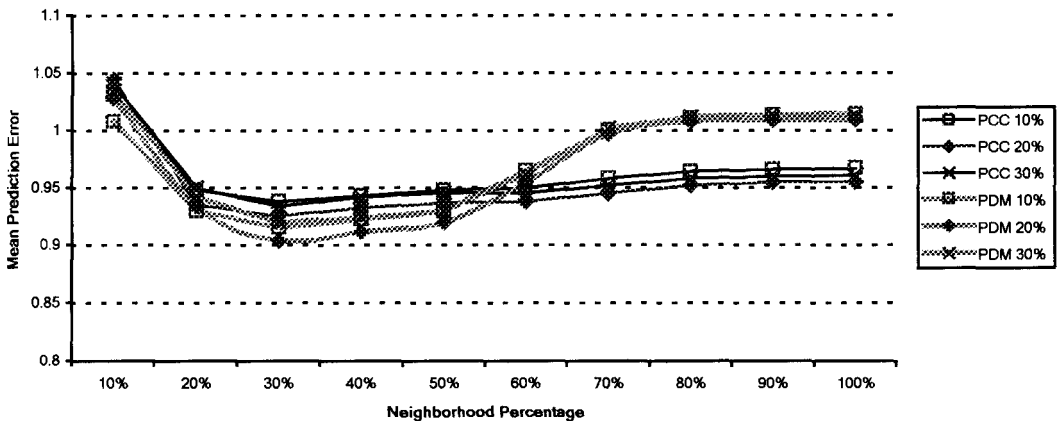
Graph 5.3: Mean error of users with 50 to 100 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.069	1.008	1.006	1.019	1.033	1.045	1.050	1.051	1.052	1.052
	PDM	1.002	0.921	0.916	0.941	0.978	1.011	1.029	1.034	1.035	1.035
20%	PCC	1.091	1.013	1.015	1.028	1.043	1.053	1.058	1.060	1.060	1.060
	PDM	1.033	0.936	0.913	0.943	0.982	1.015	1.033	1.040	1.041	1.041
30%	PCC	1.092	1.017	1.016	1.029	1.040	1.050	1.055	1.057	1.057	1.057
	PDM	1.039	0.941	0.927	0.948	0.988	1.017	1.033	1.039	1.040	1.040



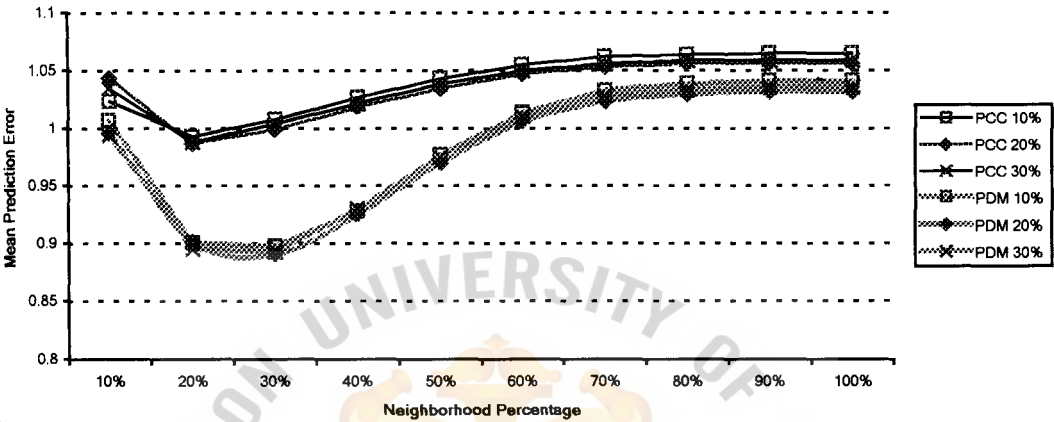
Graph 5.4: Mean error of users with 50 to 100 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.032	0.948	0.938	0.943	0.948	0.950	0.958	0.964	0.966	0.967
	PDM	1.008	0.930	0.916	0.923	0.930	0.965	1.001	1.011	1.014	1.015
20%	PCC	1.045	0.935	0.926	0.933	0.937	0.938	0.945	0.952	0.955	0.955
	PDM	1.029	0.934	0.904	0.912	0.920	0.955	0.997	1.008	1.009	1.010
30%	PCC	1.041	0.950	0.934	0.942	0.946	0.945	0.952	0.958	0.960	0.961
	PDM	1.035	0.942	0.920	0.924	0.930	0.962	1.002	1.012	1.014	1.014



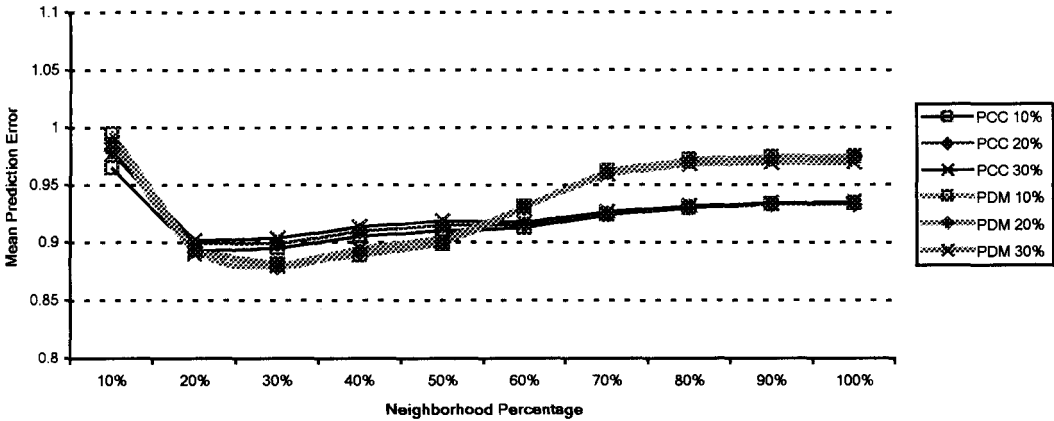
Graph 5.5: Mean error of users with 100 to 150 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.024	0.993	1.008	1.027	1.043	1.055	1.062	1.064	1.065	1.065
	PDM	1.007	0.901	0.898	0.928	0.977	1.014	1.033	1.039	1.041	1.041
20%	PCC	1.044	0.987	0.999	1.019	1.035	1.047	1.053	1.056	1.056	1.056
	PDM	0.996	0.902	0.891	0.925	0.970	1.006	1.024	1.030	1.032	1.032
30%	PCC	1.035	0.988	1.004	1.022	1.039	1.050	1.056	1.059	1.059	1.059
	PDM	0.994	0.896	0.893	0.930	0.976	1.011	1.029	1.034	1.036	1.036



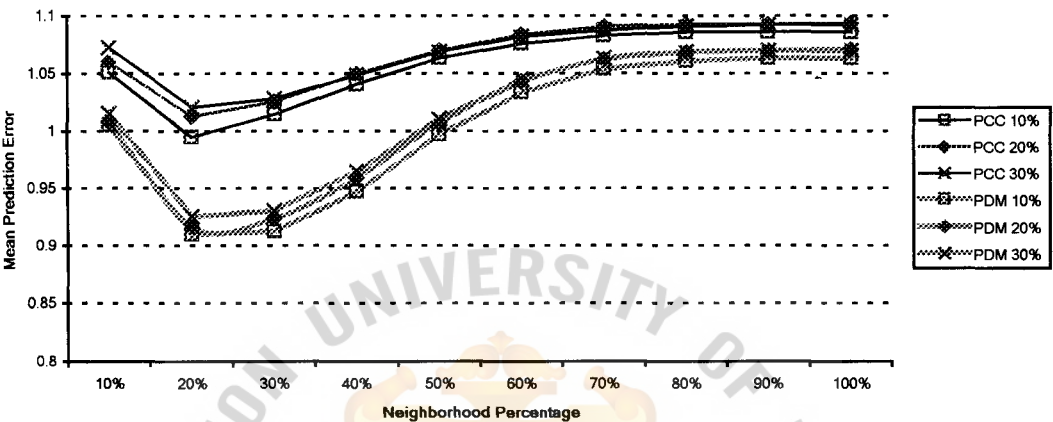
Graph 5.6: Mean error of users with 100 to 150 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.965	0.893	0.895	0.906	0.911	0.913	0.924	0.930	0.933	0.934
	PDM	0.995	0.893	0.881	0.889	0.899	0.931	0.962	0.972	0.974	0.975
20%	PCC	0.980	0.899	0.899	0.910	0.915	0.916	0.925	0.931	0.933	0.933
	PDM	0.987	0.894	0.879	0.892	0.900	0.930	0.961	0.971	0.973	0.974
30%	PCC	0.977	0.902	0.904	0.914	0.919	0.918	0.927	0.932	0.934	0.935
	PDM	0.987	0.891	0.880	0.895	0.904	0.931	0.959	0.968	0.969	0.970



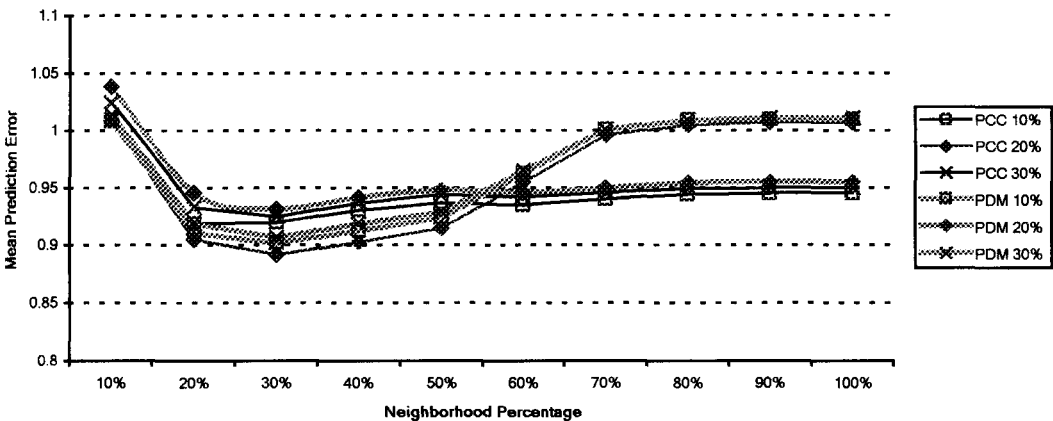
Graph 5.7: Mean error of users with 150 to 250 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.051	0.995	1.015	1.041	1.063	1.076	1.083	1.086	1.086	1.086
	PDM	1.005	0.910	0.913	0.947	0.997	1.034	1.054	1.061	1.063	1.063
20%	PCC	1.060	1.013	1.026	1.050	1.070	1.084	1.091	1.093	1.093	1.094
	PDM	1.008	0.917	0.924	0.959	1.007	1.043	1.062	1.069	1.070	1.071
30%	PCC	1.073	1.021	1.029	1.049	1.069	1.082	1.088	1.091	1.092	1.092
	PDM	1.016	0.926	0.931	0.965	1.011	1.045	1.063	1.069	1.070	1.070



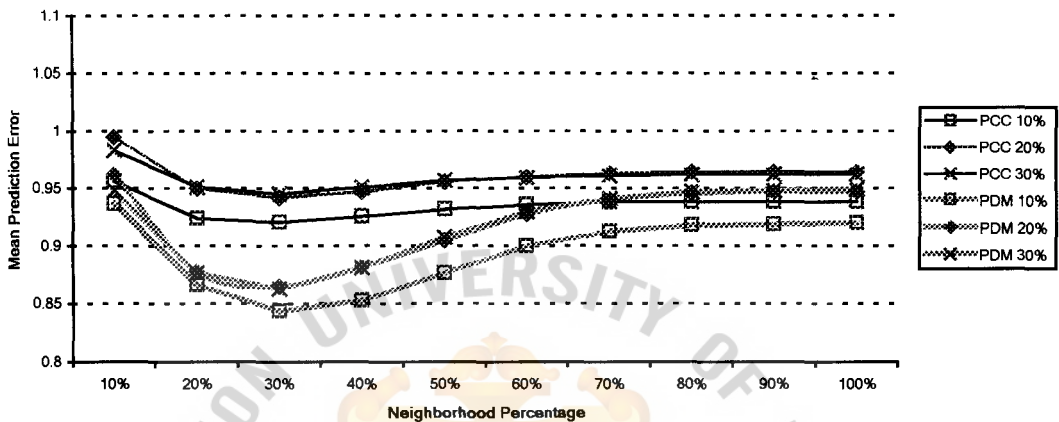
Graph 5.8: Mean error of users with 150 to 250 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.009	0.919	0.920	0.931	0.937	0.935	0.940	0.944	0.945	0.945
	PDM	1.009	0.905	0.892	0.903	0.915	0.955	0.996	1.005	1.007	1.007
20%	PCC	1.025	0.933	0.925	0.937	0.944	0.942	0.946	0.949	0.950	0.950
	PDM	1.008	0.911	0.902	0.913	0.924	0.962	1.001	1.009	1.010	1.010
30%	PCC	1.039	0.946	0.932	0.942	0.948	0.946	0.950	0.954	0.955	0.955
	PDM	1.014	0.919	0.907	0.919	0.929	0.965	1.001	1.009	1.011	1.011



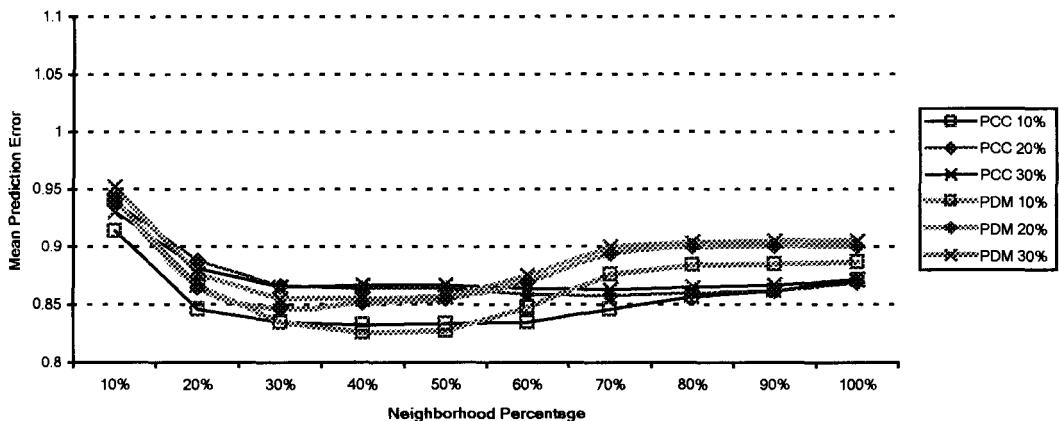
Graph 5.9: Mean error of users with 10 to 250 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.956	0.924	0.921	0.926	0.932	0.936	0.938	0.938	0.938	0.938
	PDM	0.937	0.866	0.844	0.853	0.877	0.900	0.913	0.918	0.919	0.920
20%	PCC	0.995	0.950	0.942	0.947	0.956	0.960	0.963	0.964	0.964	0.964
	PDM	0.962	0.878	0.865	0.882	0.905	0.928	0.940	0.946	0.947	0.947
30%	PCC	0.983	0.951	0.945	0.951	0.957	0.959	0.961	0.962	0.962	0.962
	PDM	0.948	0.877	0.863	0.881	0.908	0.930	0.941	0.947	0.948	0.949



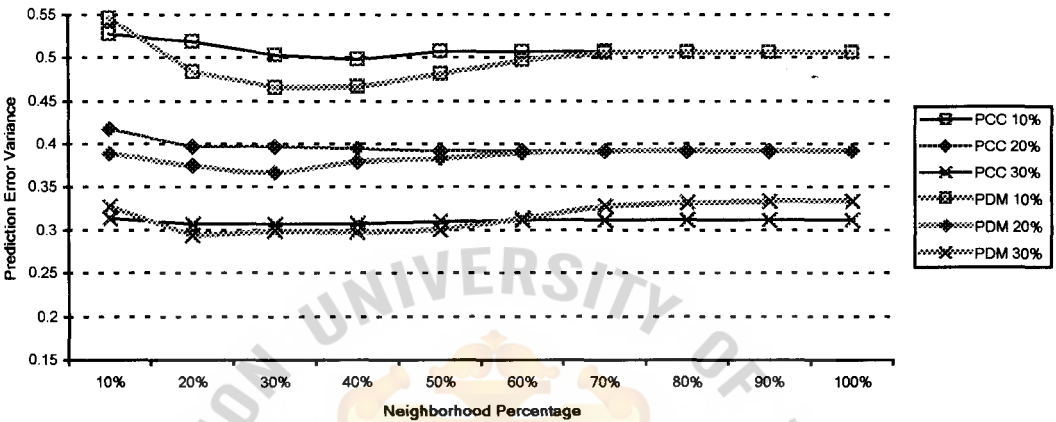
Graph 5.10: Mean error of users with 10 to 250 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.914	0.846	0.835	0.833	0.834	0.835	0.846	0.857	0.862	0.871
	PDM	0.941	0.866	0.835	0.826	0.828	0.848	0.876	0.884	0.885	0.887
20%	PCC	0.939	0.888	0.866	0.864	0.864	0.859	0.858	0.860	0.862	0.869
	PDM	0.941	0.865	0.847	0.852	0.855	0.870	0.894	0.900	0.901	0.901
30%	PCC	0.931	0.881	0.865	0.867	0.867	0.864	0.863	0.865	0.867	0.872
	PDM	0.952	0.876	0.855	0.855	0.857	0.875	0.900	0.904	0.905	0.905



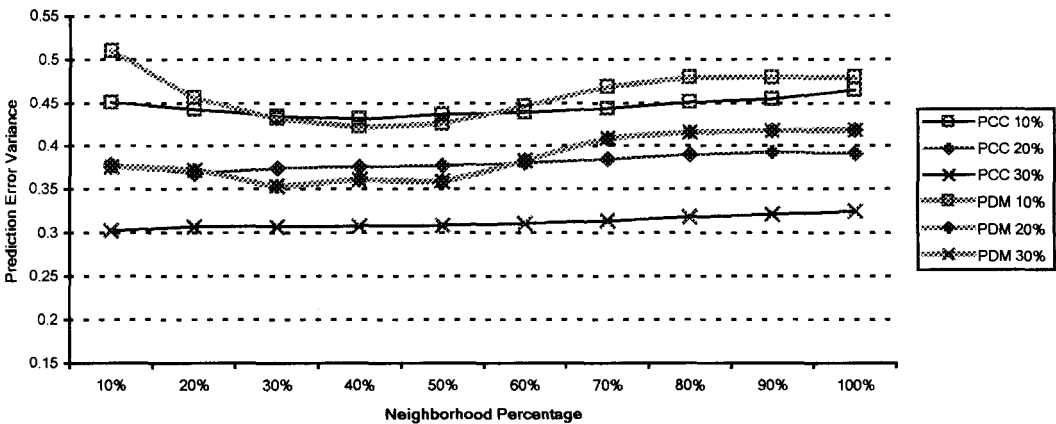
Graph 5.11: Error variance of users with 10 to 50 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.528	0.519	0.503	0.498	0.508	0.507	0.508	0.506	0.506	0.506
	PDM	0.546	0.484	0.466	0.467	0.482	0.497	0.506	0.506	0.506	0.506
20%	PCC	0.418	0.397	0.397	0.395	0.392	0.392	0.392	0.391	0.391	0.392
	PDM	0.389	0.375	0.367	0.379	0.383	0.390	0.392	0.392	0.392	0.392
30%	PCC	0.314	0.307	0.307	0.308	0.310	0.312	0.312	0.312	0.312	0.312
	PDM	0.327	0.294	0.299	0.298	0.301	0.314	0.328	0.332	0.333	0.333



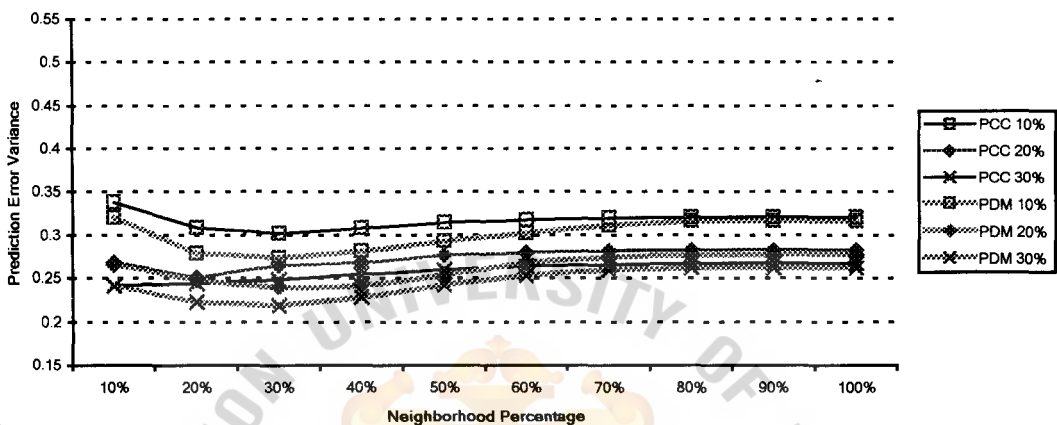
Graph 5.12: Error variance of users with 10 to 50 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.451	0.443	0.435	0.432	0.437	0.439	0.444	0.451	0.455	0.465
	PDM	0.510	0.456	0.432	0.422	0.426	0.447	0.469	0.479	0.479	0.480
20%	PCC	0.379	0.368	0.374	0.376	0.377	0.380	0.385	0.390	0.393	0.391
	PDM	0.376	0.372	0.354	0.361	0.359	0.383	0.409	0.416	0.418	0.419
30%	PCC	0.303	0.307	0.307	0.308	0.309	0.311	0.314	0.318	0.321	0.325
	PDM	0.376	0.372	0.354	0.361	0.359	0.383	0.409	0.416	0.418	0.419



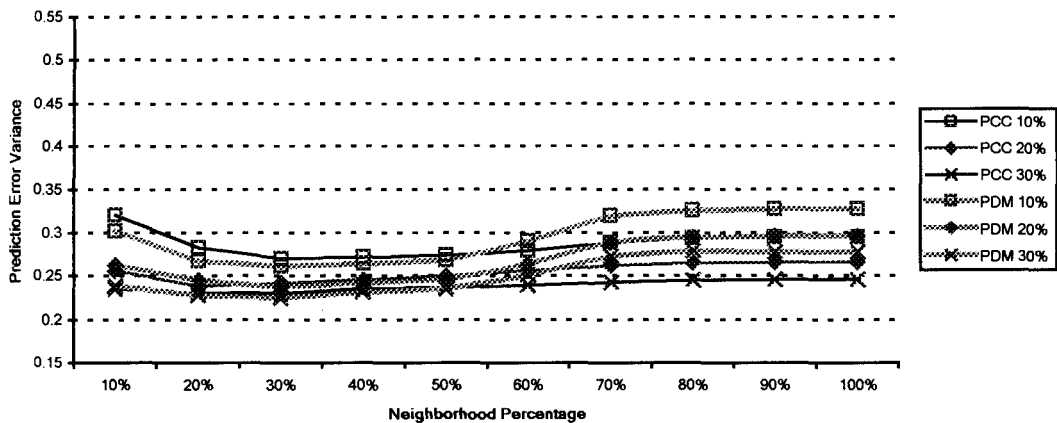
Graph 5.13: Error variance of users with 50 to 100 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.338	0.309	0.302	0.308	0.315	0.318	0.320	0.321	0.321	0.321
	PDM	0.321	0.279	0.274	0.281	0.293	0.303	0.312	0.316	0.316	0.316
20%	PCC	0.269	0.251	0.265	0.268	0.277	0.280	0.282	0.283	0.283	0.283
	PDM	0.266	0.249	0.240	0.242	0.254	0.268	0.275	0.277	0.278	0.278
30%	PCC	0.242	0.244	0.249	0.255	0.260	0.264	0.266	0.267	0.267	0.267
	PDM	0.241	0.223	0.219	0.228	0.243	0.254	0.260	0.262	0.262	0.262



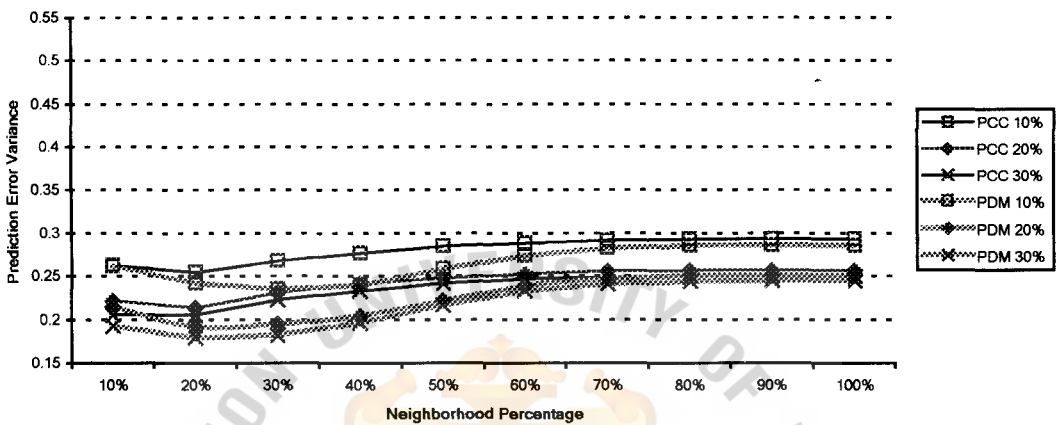
Graph 5.14: Error variance of users with 50 to 100 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.321	0.283	0.270	0.272	0.274	0.279	0.289	0.294	0.295	0.295
	PDM	0.302	0.267	0.261	0.264	0.268	0.290	0.320	0.326	0.327	0.327
20%	PCC	0.256	0.238	0.242	0.246	0.250	0.256	0.262	0.265	0.266	0.266
	PDM	0.263	0.246	0.238	0.243	0.247	0.264	0.289	0.295	0.296	0.296
30%	PCC	0.235	0.230	0.230	0.235	0.237	0.239	0.243	0.245	0.246	0.246
	PDM	0.238	0.228	0.225	0.232	0.235	0.251	0.273	0.279	0.278	0.278



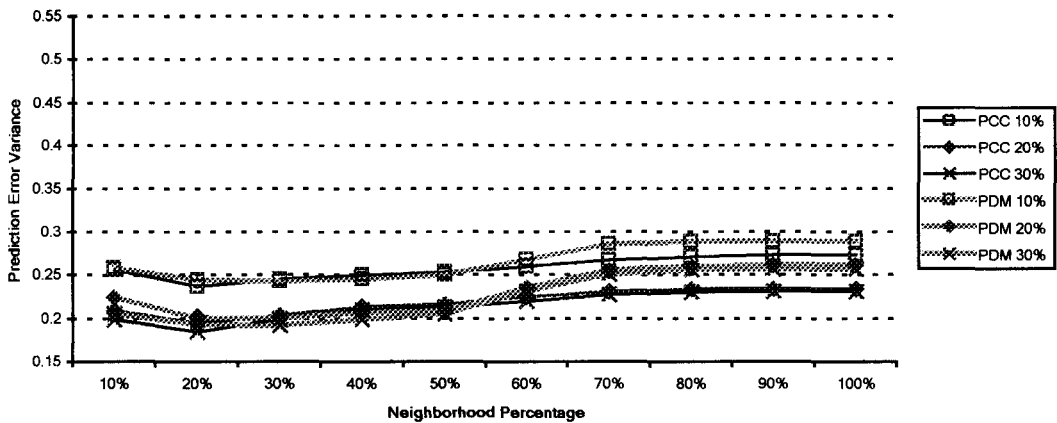
Graph 5.15: Error variance of users with 100 to 150 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.263	0.255	0.268	0.277	0.285	0.289	0.292	0.293	0.293	0.293
	PDM	0.262	0.242	0.235	0.240	0.259	0.274	0.283	0.286	0.286	0.286
20%	PCC	0.222	0.214	0.231	0.242	0.248	0.253	0.256	0.257	0.257	0.257
	PDM	0.215	0.192	0.195	0.205	0.222	0.239	0.247	0.251	0.251	0.251
30%	PCC	0.206	0.206	0.223	0.233	0.242	0.247	0.249	0.250	0.250	0.250
	PDM	0.193	0.180	0.183	0.197	0.217	0.233	0.241	0.244	0.245	0.245



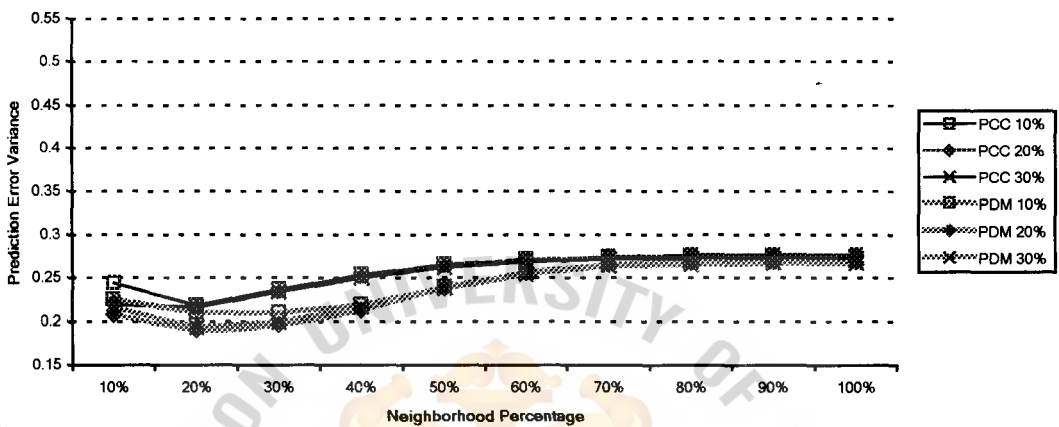
Graph 5.16: Error variance of users with 100 to 150 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.257	0.237	0.245	0.250	0.253	0.260	0.267	0.271	0.273	0.273
	PDM	0.259	0.244	0.243	0.246	0.250	0.268	0.286	0.289	0.289	0.289
20%	PCC	0.209	0.195	0.204	0.214	0.217	0.225	0.231	0.234	0.234	0.234
	PDM	0.224	0.203	0.201	0.207	0.211	0.235	0.255	0.260	0.261	0.261
30%	PCC	0.199	0.185	0.200	0.210	0.213	0.220	0.227	0.230	0.231	0.231
	PDM	0.206	0.194	0.192	0.200	0.206	0.231	0.251	0.256	0.257	0.257



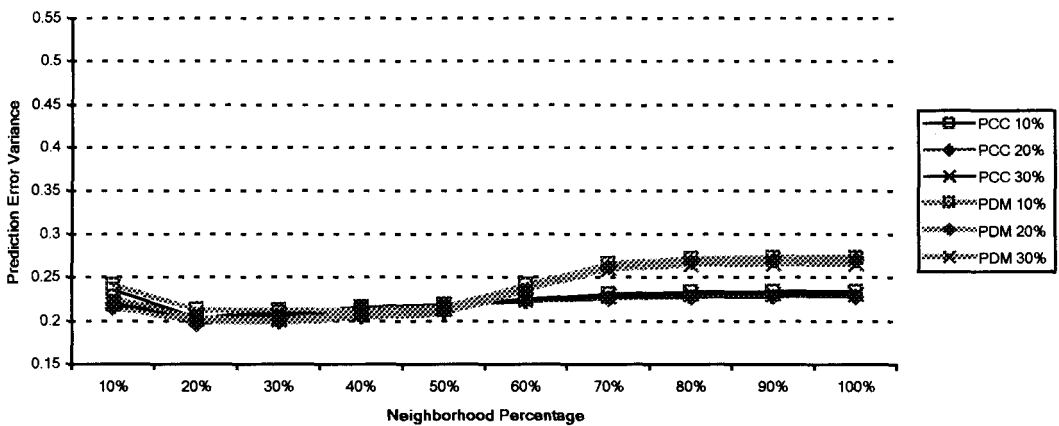
Graph 5.17: Error variance of users with 150 to 250 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.244	0.218	0.237	0.254	0.266	0.272	0.275	0.277	0.277	0.277
	PDM	0.225	0.211	0.210	0.219	0.240	0.257	0.266	0.268	0.268	0.269
20%	PCC	0.222	0.216	0.234	0.252	0.265	0.272	0.275	0.277	0.277	0.277
	PDM	0.208	0.191	0.197	0.213	0.238	0.256	0.265	0.268	0.269	0.269
30%	PCC	0.219	0.216	0.234	0.250	0.262	0.269	0.272	0.273	0.273	0.273
	PDM	0.216	0.194	0.200	0.217	0.238	0.255	0.264	0.266	0.267	0.267



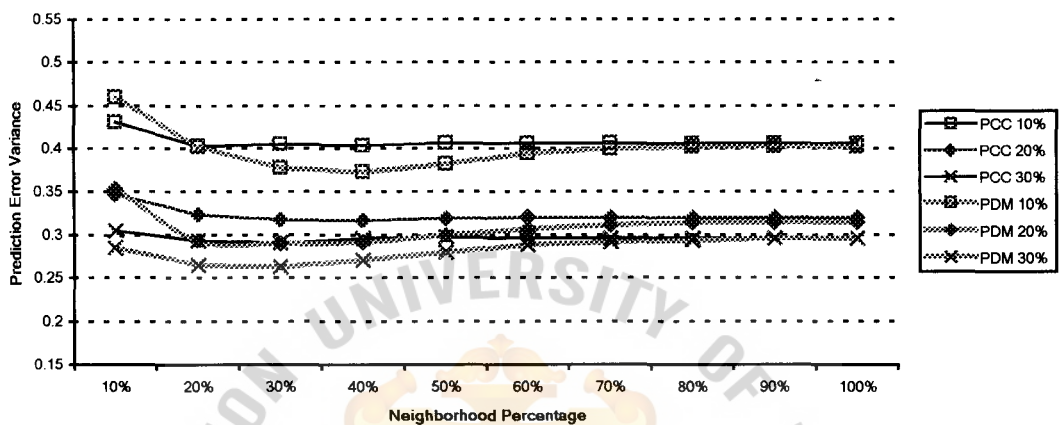
Graph 5.18: Error variance of users with 150 to 250 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.235	0.204	0.210	0.216	0.219	0.225	0.231	0.233	0.234	0.234
	PDM	0.241	0.213	0.212	0.213	0.216	0.241	0.266	0.272	0.273	0.273
20%	PCC	0.221	0.197	0.205	0.215	0.219	0.222	0.226	0.227	0.228	0.228
	PDM	0.215	0.201	0.199	0.205	0.210	0.235	0.262	0.268	0.270	0.270
30%	PCC	0.220	0.203	0.207	0.216	0.219	0.224	0.229	0.231	0.231	0.231
	PDM	0.226	0.204	0.202	0.209	0.214	0.236	0.260	0.265	0.266	0.266



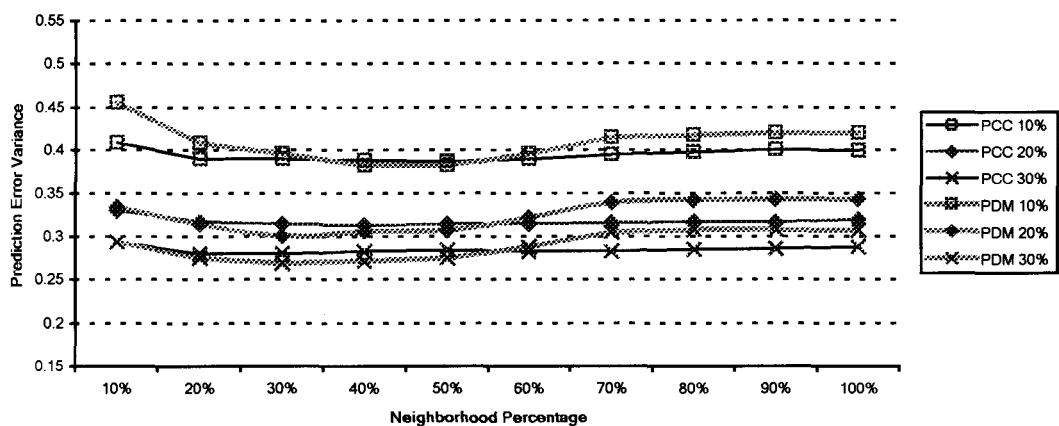
Graph 5.19: Error variance of users with 10 to 250 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.431	0.403	0.405	0.404	0.407	0.406	0.407	0.406	0.406	0.406
	PDM	0.460	0.402	0.378	0.373	0.382	0.395	0.400	0.402	0.402	0.402
20%	PCC	0.347	0.324	0.318	0.317	0.319	0.321	0.320	0.320	0.320	0.320
	PDM	0.353	0.293	0.290	0.292	0.300	0.307	0.312	0.315	0.315	0.315
30%	PCC	0.306	0.293	0.291	0.296	0.298	0.296	0.296	0.296	0.296	0.296
	PDM	0.285	0.265	0.263	0.271	0.280	0.289	0.292	0.294	0.296	0.296



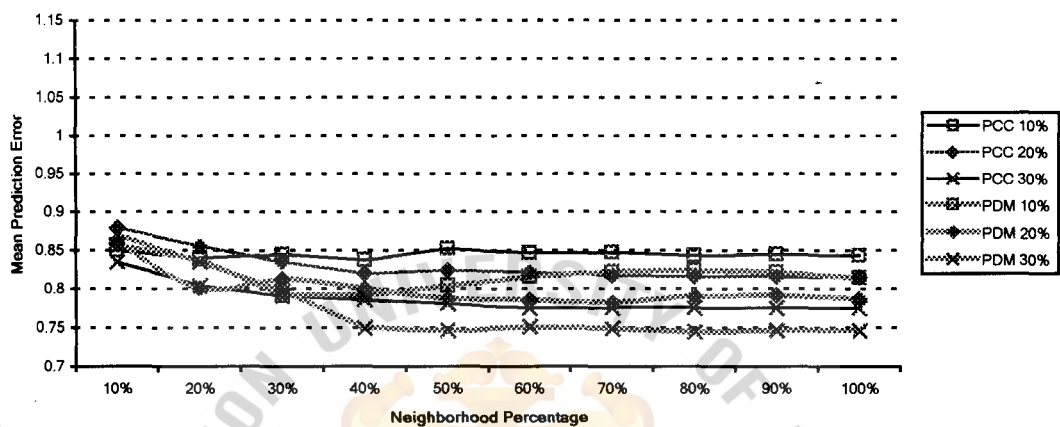
Graph 5.20: Error variance of users with 10 to 250 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.409	0.390	0.390	0.388	0.387	0.390	0.395	0.398	0.401	0.399
	PDM	0.456	0.408	0.396	0.382	0.382	0.396	0.415	0.418	0.420	0.420
20%	PCC	0.330	0.318	0.315	0.313	0.315	0.315	0.316	0.318	0.317	0.320
	PDM	0.335	0.315	0.301	0.306	0.307	0.322	0.340	0.342	0.343	0.343
30%	PCC	0.294	0.280	0.280	0.283	0.284	0.283	0.283	0.285	0.286	0.288
	PDM	0.294	0.275	0.269	0.272	0.275	0.288	0.305	0.308	0.308	0.307



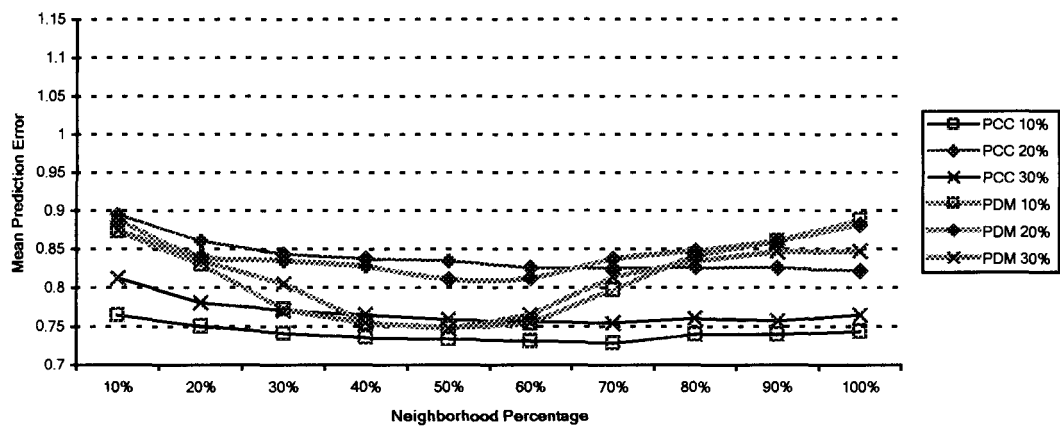
Graph 5.21: Mean error of users with 10 to 50 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.850	0.840	0.845	0.838	0.853	0.847	0.848	0.844	0.845	0.844
	PDM	0.857	0.839	0.792	0.792	0.804	0.816	0.823	0.824	0.822	0.814
20%	PCC	0.880	0.856	0.835	0.820	0.824	0.822	0.818	0.816	0.816	0.815
	PDM	0.860	0.801	0.813	0.801	0.788	0.786	0.783	0.791	0.792	0.788
30%	PCC	0.835	0.804	0.791	0.786	0.781	0.775	0.776	0.776	0.775	0.775
	PDM	0.871	0.835	0.798	0.750	0.747	0.752	0.749	0.745	0.747	0.747



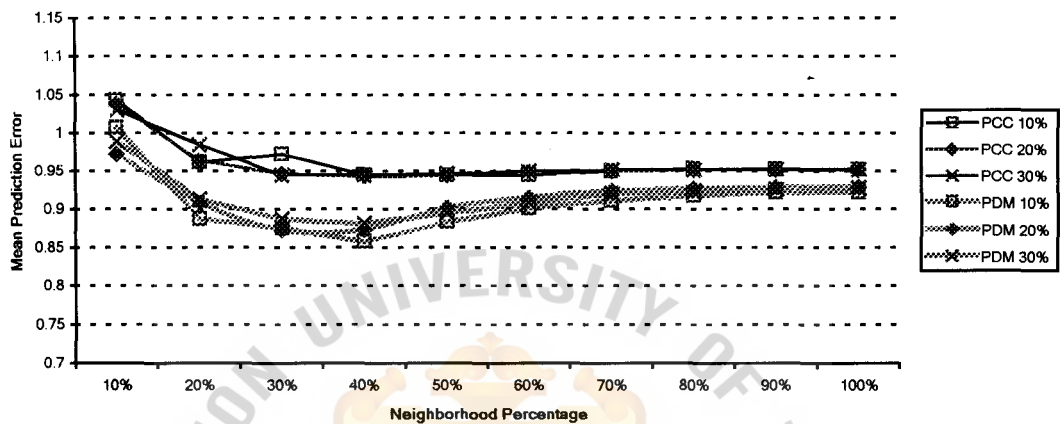
Graph 5.22: Mean error of users with 10 to 50 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.765	0.751	0.742	0.736	0.734	0.732	0.729	0.740	0.740	0.744
	PDM	0.874	0.830	0.773	0.754	0.749	0.754	0.798	0.843	0.861	0.889
20%	PCC	0.895	0.861	0.844	0.838	0.835	0.826	0.825	0.827	0.826	0.822
	PDM	0.888	0.841	0.835	0.828	0.811	0.813	0.838	0.849	0.861	0.882
30%	PCC	0.813	0.781	0.771	0.765	0.759	0.757	0.755	0.761	0.758	0.765
	PDM	0.875	0.835	0.805	0.756	0.748	0.765	0.814	0.835	0.847	0.848



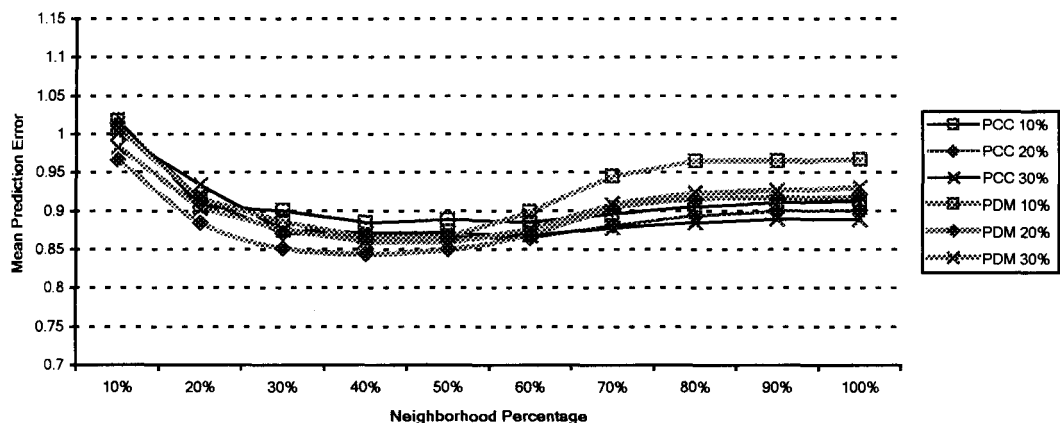
Graph 5.23: Mean error of users with 50 to 100 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.043	0.961	0.972	0.945	0.944	0.945	0.95	0.953	0.952	0.952
	PDM	1.007	0.888	0.876	0.858	0.884	0.902	0.911	0.918	0.922	0.922
20%	PCC	1.038	0.964	0.946	0.942	0.944	0.950	0.950	0.951	0.951	0.951
	PDM	0.972	0.909	0.874	0.874	0.903	0.915	0.924	0.927	0.929	0.929
30%	PCC	1.031	0.985	0.945	0.945	0.946	0.950	0.951	0.951	0.952	0.952
	PDM	0.989	0.913	0.888	0.881	0.895	0.909	0.920	0.924	0.926	0.928



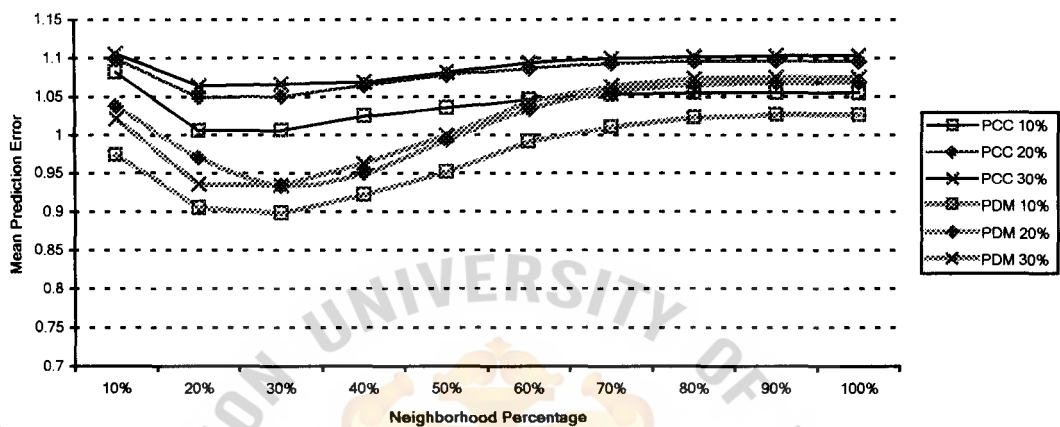
Graph 5.24: Mean error of users with 50 to 100 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.018	0.907	0.901	0.885	0.889	0.886	0.897	0.906	0.911	0.913
	PDM	1.007	0.916	0.887	0.866	0.867	0.899	0.945	0.965	0.965	0.967
20%	PCC	1.015	0.916	0.872	0.867	0.867	0.865	0.882	0.894	0.900	0.902
	PDM	0.967	0.885	0.852	0.844	0.850	0.869	0.903	0.916	0.917	0.918
30%	PCC	1.001	0.933	0.876	0.872	0.872	0.868	0.878	0.885	0.890	0.889
	PDM	0.983	0.904	0.877	0.860	0.859	0.877	0.909	0.923	0.927	0.930



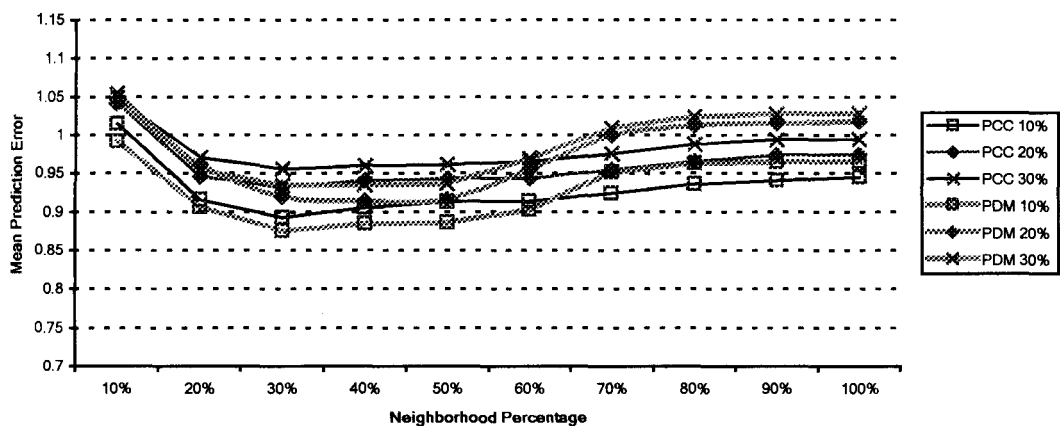
Graph 5.25: Mean error of users with 100 to 150 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.082	1.006	1.007	1.025	1.036	1.047	1.053	1.055	1.055	1.055
	PDM	0.974	0.905	0.898	0.922	0.952	0.992	1.011	1.023	1.027	1.027
20%	PCC	1.099	1.049	1.051	1.065	1.078	1.087	1.093	1.096	1.097	1.096
	PDM	1.038	0.971	0.934	0.950	0.994	1.034	1.057	1.066	1.068	1.069
30%	PCC	1.106	1.064	1.067	1.069	1.082	1.094	1.100	1.102	1.103	1.103
	PDM	1.022	0.936	0.934	0.963	1.000	1.044	1.064	1.073	1.075	1.075



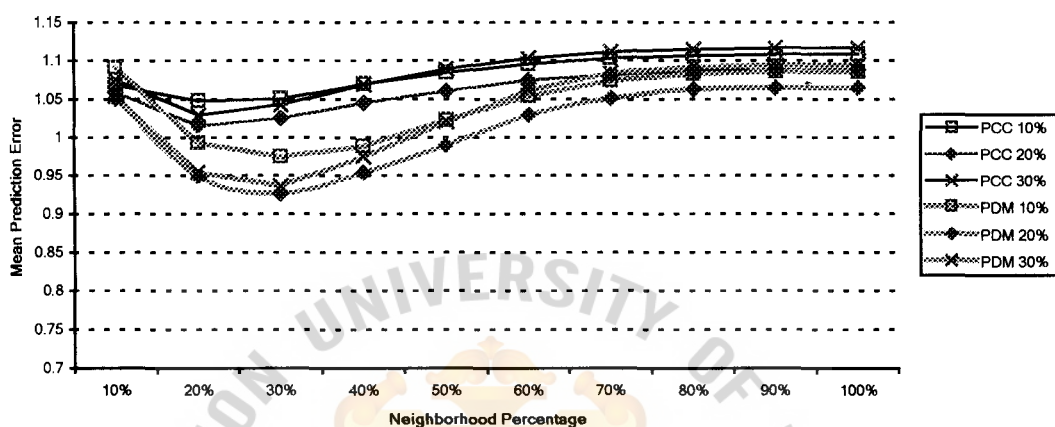
Graph 5.26: Mean error of users with 100 to 150 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.015	0.916	0.893	0.906	0.913	0.914	0.924	0.936	0.941	0.945
	PDM	0.992	0.907	0.876	0.885	0.887	0.903	0.952	0.963	0.965	0.965
20%	PCC	1.042	0.946	0.932	0.941	0.943	0.943	0.955	0.966	0.974	0.975
	PDM	1.043	0.962	0.920	0.914	0.917	0.956	1.000	1.013	1.016	1.018
30%	PCC	1.049	0.971	0.956	0.960	0.962	0.966	0.976	0.988	0.994	0.995
	PDM	1.055	0.954	0.934	0.935	0.937	0.970	1.010	1.025	1.028	1.029



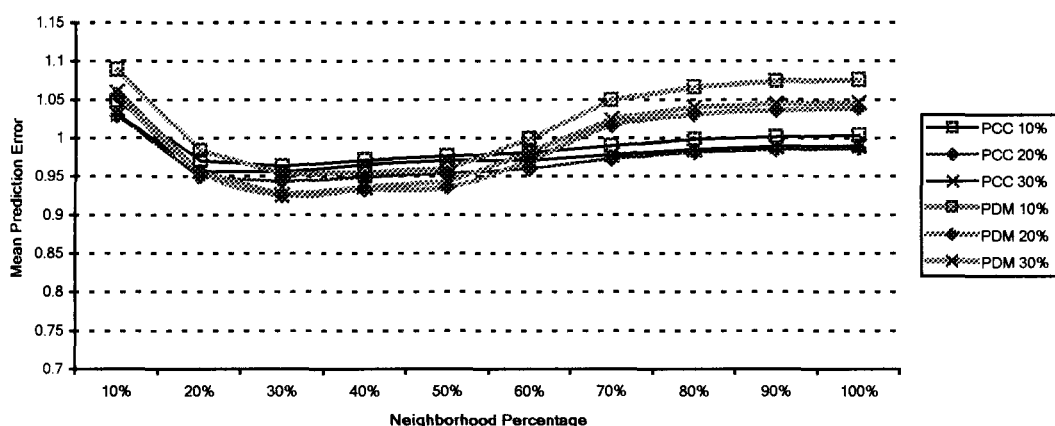
Graph 5.27: Mean error of users with 150 to 250 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.068	1.048	1.051	1.070	1.084	1.096	1.103	1.107	1.108	1.108
	PDM	1.091	0.993	0.975	0.989	1.023	1.054	1.074	1.083	1.086	1.086
20%	PCC	1.059	1.016	1.026	1.046	1.061	1.075	1.082	1.087	1.088	1.088
	PDM	1.051	0.949	0.927	0.954	0.990	1.030	1.052	1.063	1.065	1.065
30%	PCC	1.075	1.030	1.043	1.069	1.089	1.103	1.112	1.115	1.117	1.117
	PDM	1.054	0.954	0.937	0.975	1.021	1.062	1.084	1.092	1.094	1.095



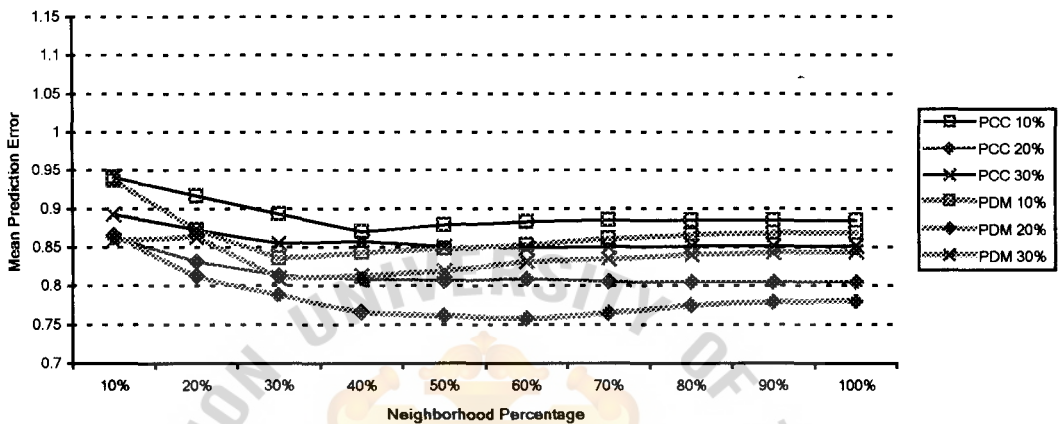
Graph 5.28: Mean error of users with 150 to 250 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	1.048	0.971	0.963	0.972	0.977	0.981	0.990	0.998	1.002	1.004
	PDM	1.089	0.984	0.953	0.955	0.960	0.999	1.050	1.067	1.074	1.076
20%	PCC	1.030	0.949	0.943	0.949	0.954	0.960	0.973	0.982	0.985	0.987
	PDM	1.057	0.957	0.926	0.933	0.936	0.971	1.017	1.032	1.037	1.039
30%	PCC	1.032	0.957	0.956	0.966	0.971	0.971	0.978	0.985	0.989	0.990
	PDM	1.061	0.957	0.926	0.936	0.943	0.982	1.025	1.041	1.045	1.047



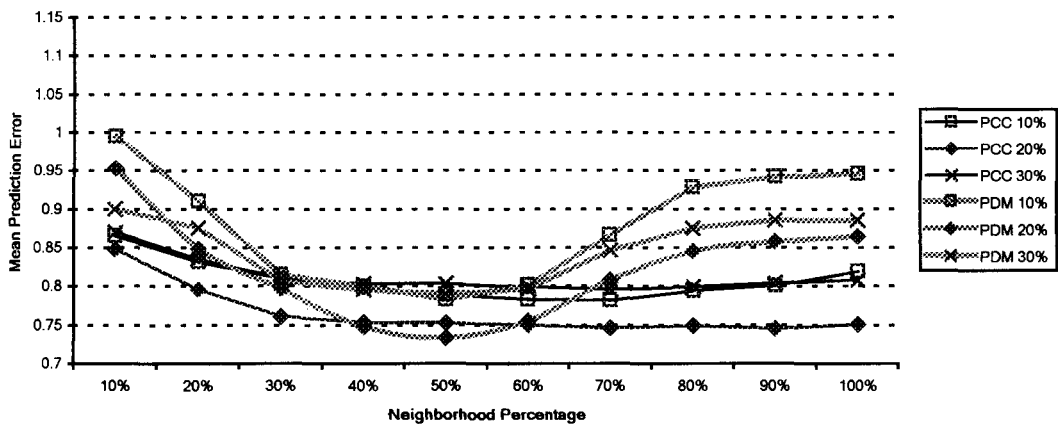
Graph 5.29: Mean error of users with 10 to 250 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.941	0.917	0.893	0.871	0.879	0.883	0.886	0.885	0.885	0.885
	PDM	0.937	0.873	0.836	0.843	0.848	0.853	0.862	0.867	0.869	0.869
20%	PCC	0.863	0.832	0.813	0.809	0.807	0.809	0.806	0.805	0.805	0.805
	PDM	0.867	0.813	0.788	0.767	0.762	0.758	0.765	0.775	0.779	0.780
30%	PCC	0.893	0.873	0.855	0.858	0.851	0.849	0.851	0.852	0.852	0.852
	PDM	0.859	0.864	0.810	0.813	0.819	0.832	0.835	0.841	0.843	0.844



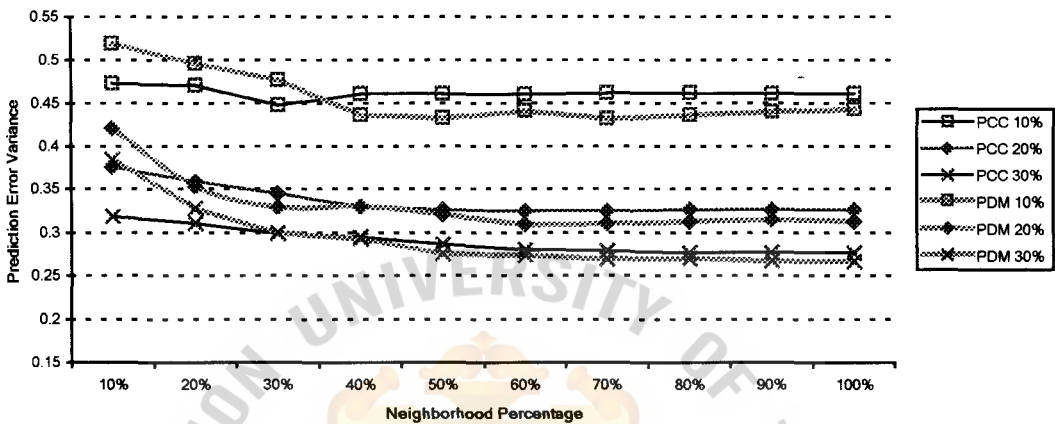
Graph 5.30: Mean error of users with 10 to 250 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.866	0.832	0.810	0.798	0.789	0.783	0.782	0.794	0.801	0.819
	PDM	0.995	0.911	0.815	0.800	0.783	0.802	0.867	0.928	0.942	0.946
20%	PCC	0.848	0.796	0.762	0.754	0.753	0.751	0.747	0.750	0.746	0.752
	PDM	0.953	0.849	0.797	0.748	0.734	0.756	0.808	0.846	0.858	0.865
30%	PCC	0.871	0.835	0.811	0.803	0.803	0.799	0.796	0.799	0.804	0.808
	PDM	0.901	0.876	0.807	0.796	0.790	0.797	0.847	0.876	0.886	0.886



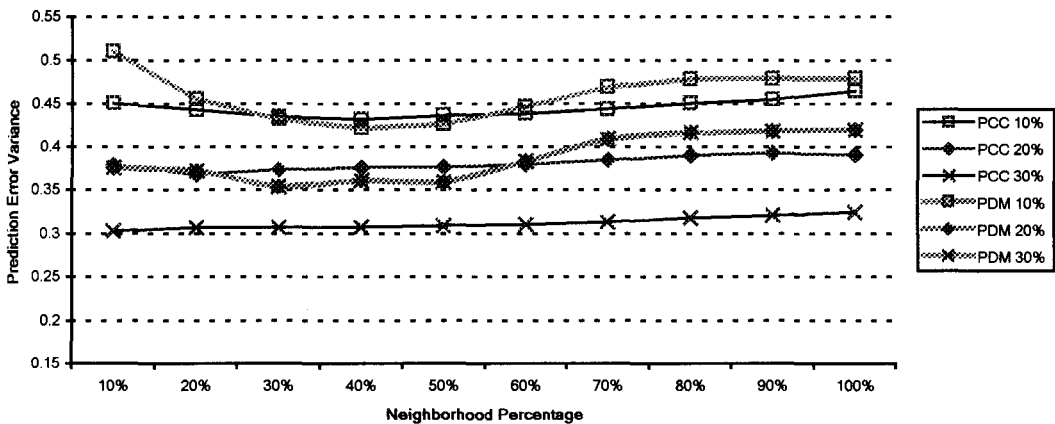
Graph 5.31: Error variance of users with 10 to 50 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.473	0.471	0.448	0.461	0.461	0.460	0.462	0.462	0.461	0.461
	PDM	0.519	0.496	0.477	0.437	0.433	0.441	0.432	0.437	0.440	0.443
20%	PCC	0.376	0.359	0.345	0.330	0.326	0.325	0.325	0.327	0.326	0.326
	PDM	0.421	0.353	0.329	0.331	0.321	0.310	0.311	0.313	0.315	0.313
30%	PCC	0.319	0.311	0.299	0.295	0.286	0.280	0.279	0.277	0.277	0.277
	PDM	0.385	0.328	0.300	0.293	0.275	0.274	0.269	0.270	0.267	0.266



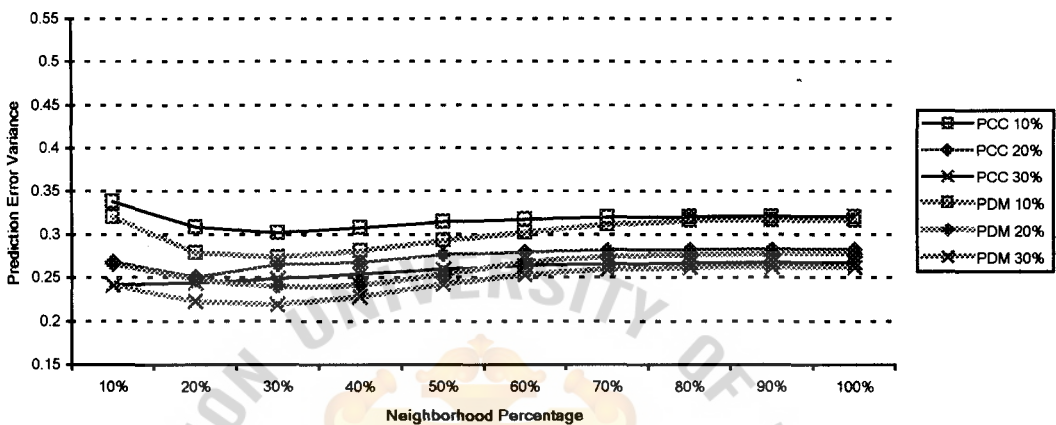
Graph 5.32: Error variance of users with 10 to 50 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.451	0.443	0.435	0.432	0.437	0.439	0.444	0.451	0.455	0.465
	PDM	0.510	0.456	0.432	0.422	0.426	0.447	0.469	0.479	0.479	0.480
20%	PCC	0.379	0.368	0.374	0.376	0.377	0.380	0.385	0.390	0.393	0.391
	PDM	0.376	0.372	0.354	0.361	0.359	0.383	0.409	0.416	0.418	0.419
30%	PCC	0.303	0.307	0.307	0.308	0.309	0.311	0.314	0.318	0.321	0.325
	PDM	0.376	0.372	0.354	0.361	0.359	0.383	0.409	0.416	0.418	0.419



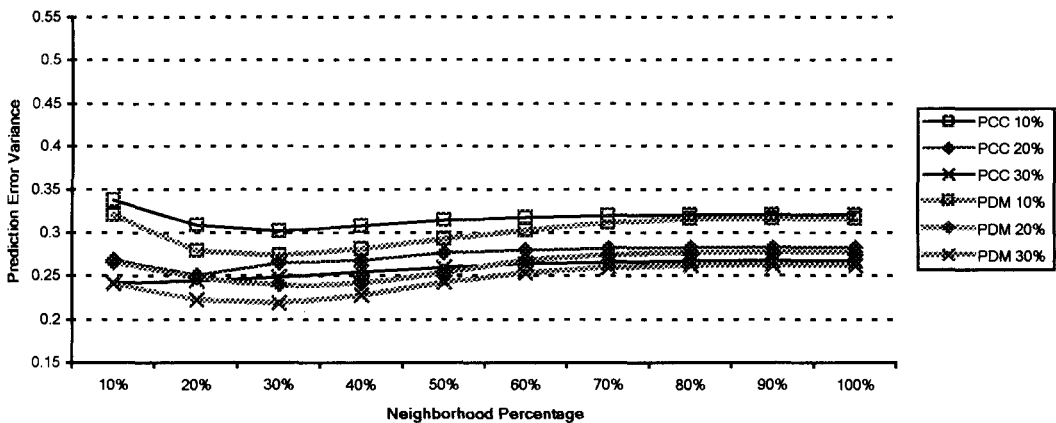
Graph 5.33: Error variance of users with 50 to 100 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.289	0.276	0.269	0.266	0.264	0.268	0.269	0.272	0.272	0.272
	PDM	0.326	0.286	0.276	0.269	0.269	0.269	0.272	0.271	0.272	0.272
20%	PCC	0.237	0.218	0.227	0.234	0.234	0.236	0.235	0.235	0.235	0.235
	PDM	0.245	0.250	0.243	0.223	0.219	0.224	0.227	0.229	0.229	0.229
30%	PCC	0.234	0.203	0.202	0.211	0.215	0.216	0.215	0.215	0.215	0.215
	PDM	0.242	0.217	0.203	0.201	0.205	0.208	0.209	0.209	0.210	0.213



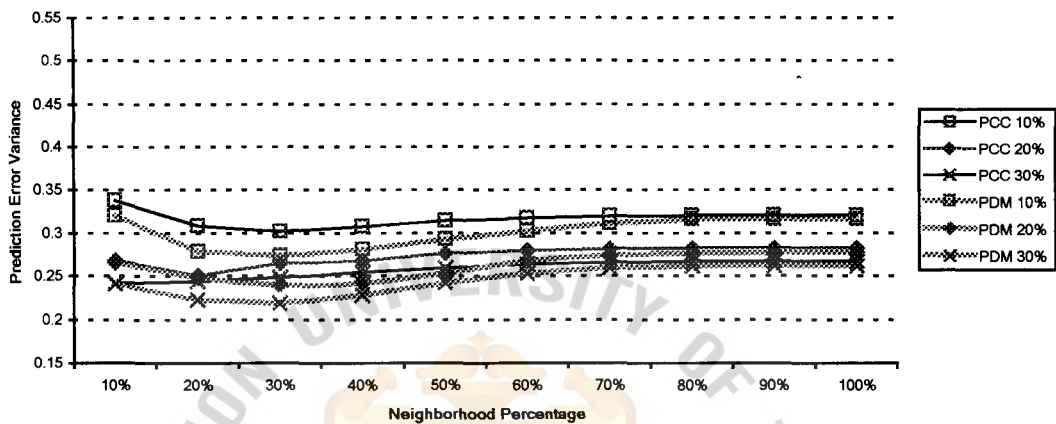
Graph 5.34: Error variance of users with 50 to 100 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.308	0.278	0.278	0.279	0.278	0.295	0.311	0.318	0.321	0.322
	PDM	0.318	0.292	0.282	0.277	0.276	0.296	0.342	0.358	0.364	0.366
20%	PCC	0.240	0.226	0.234	0.240	0.241	0.241	0.241	0.246	0.245	0.244
	PDM	0.233	0.243	0.240	0.235	0.235	0.243	0.254	0.258	0.257	0.257
30%	PCC	0.221	0.207	0.195	0.202	0.206	0.205	0.217	0.223	0.226	0.228
	PDM	0.236	0.214	0.204	0.197	0.198	0.216	0.243	0.254	0.256	0.256



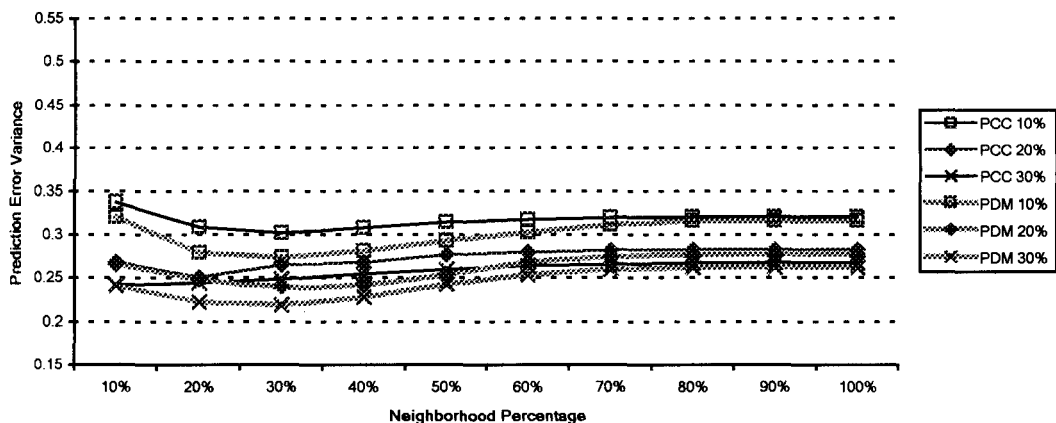
Graph 5.35: Error variance of users with 100 to 150 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.254	0.221	0.237	0.248	0.250	0.251	0.252	0.253	0.252	0.252
	PDM	0.244	0.213	0.204	0.213	0.220	0.233	0.240	0.239	0.241	0.241
20%	PCC	0.231	0.217	0.201	0.214	0.221	0.226	0.228	0.229	0.229	0.228
	PDM	0.250	0.215	0.196	0.192	0.195	0.207	0.214	0.219	0.221	0.221
30%	PCC	0.207	0.197	0.210	0.219	0.223	0.225	0.225	0.227	0.227	0.227
	PDM	0.205	0.210	0.196	0.193	0.201	0.215	0.222	0.225	0.226	0.226



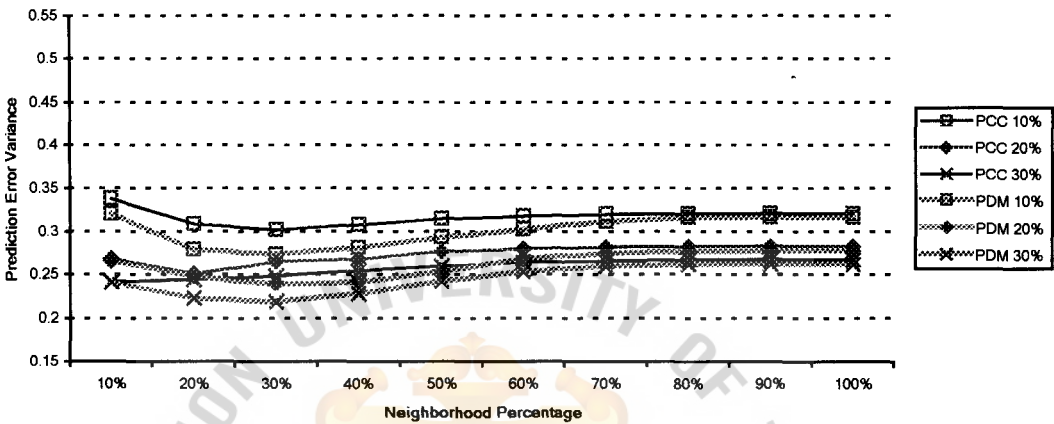
Graph 5.36: Error variance of users with 100 to 150 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.237	0.203	0.215	0.219	0.222	0.236	0.258	0.270	0.273	0.276
	PDM	0.259	0.214	0.205	0.216	0.217	0.239	0.287	0.293	0.292	0.290
20%	PCC	0.211	0.192	0.193	0.208	0.213	0.224	0.239	0.246	0.250	0.250
	PDM	0.256	0.225	0.210	0.210	0.209	0.234	0.274	0.284	0.284	0.284
30%	PCC	0.188	0.189	0.200	0.208	0.209	0.218	0.229	0.235	0.234	0.234
	PDM	0.208	0.200	0.199	0.197	0.200	0.230	0.264	0.273	0.274	0.273



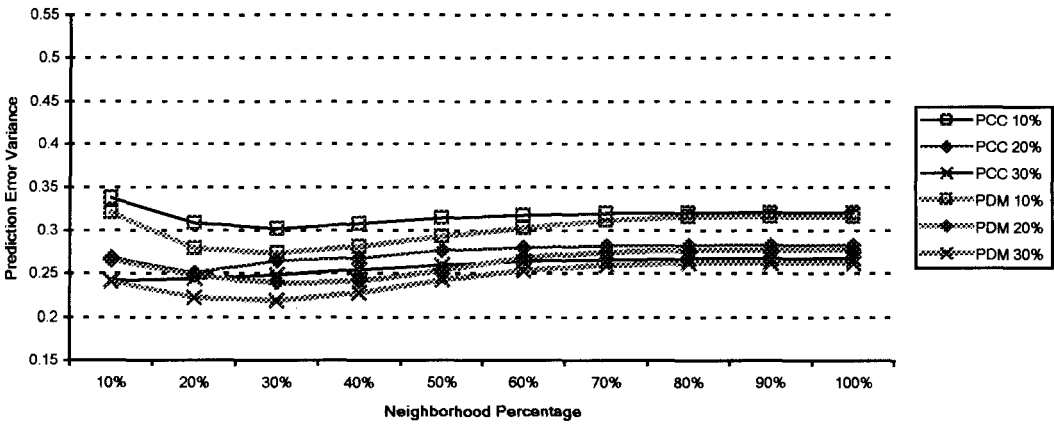
Graph 5.37: Error variance of users with 150 to 250 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.246	0.247	0.252	0.259	0.273	0.275	0.279	0.280	0.280	0.280
	PDM	0.262	0.246	0.254	0.257	0.267	0.275	0.277	0.279	0.280	0.280
20%	PCC	0.203	0.216	0.213	0.226	0.234	0.238	0.241	0.244	0.245	0.245
	PDM	0.216	0.206	0.202	0.209	0.226	0.236	0.242	0.243	0.244	0.244
30%	PCC	0.211	0.198	0.207	0.222	0.234	0.242	0.246	0.248	0.248	0.249
	PDM	0.211	0.188	0.189	0.197	0.212	0.228	0.239	0.244	0.245	0.245



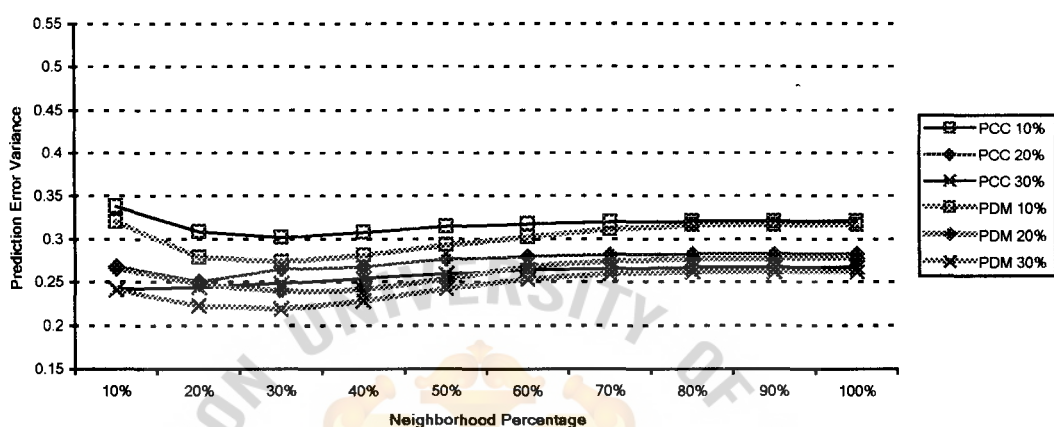
Graph 5.38: Error variance of users with 150 to 250 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.239	0.237	0.229	0.233	0.236	0.238	0.247	0.251	0.252	0.252
	PDM	0.265	0.244	0.230	0.235	0.236	0.248	0.274	0.280	0.278	0.278
20%	PCC	0.204	0.203	0.207	0.215	0.217	0.219	0.230	0.235	0.237	0.238
	PDM	0.231	0.215	0.216	0.220	0.225	0.239	0.270	0.275	0.278	0.278
30%	PCC	0.190	0.180	0.189	0.193	0.199	0.205	0.212	0.217	0.220	0.221
	PDM	0.221	0.193	0.189	0.195	0.204	0.219	0.249	0.256	0.258	0.260



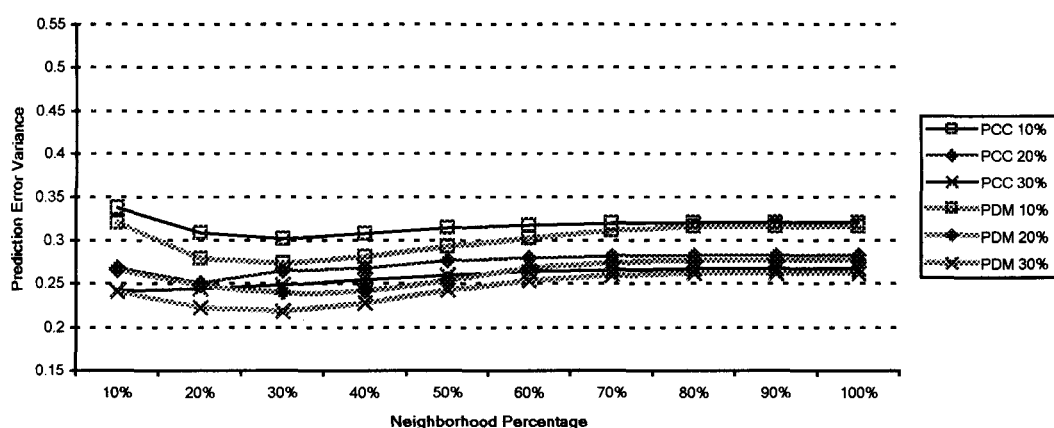
Graph 5.39: Error variance of users with 10 to 250 votes (AWS)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.437	0.399	0.425	0.417	0.414	0.421	0.424	0.424	0.425	0.425
	PDM	0.468	0.456	0.427	0.416	0.418	0.404	0.409	0.415	0.416	0.416
20%	PCC	0.294	0.287	0.276	0.283	0.277	0.280	0.277	0.277	0.277	0.277
	PDM	0.351	0.324	0.313	0.286	0.277	0.276	0.269	0.274	0.276	0.276
30%	PCC	0.265	0.255	0.257	0.246	0.245	0.249	0.249	0.249	0.249	0.249
	PDM	0.301	0.268	0.258	0.239	0.239	0.248	0.243	0.244	0.244	0.244



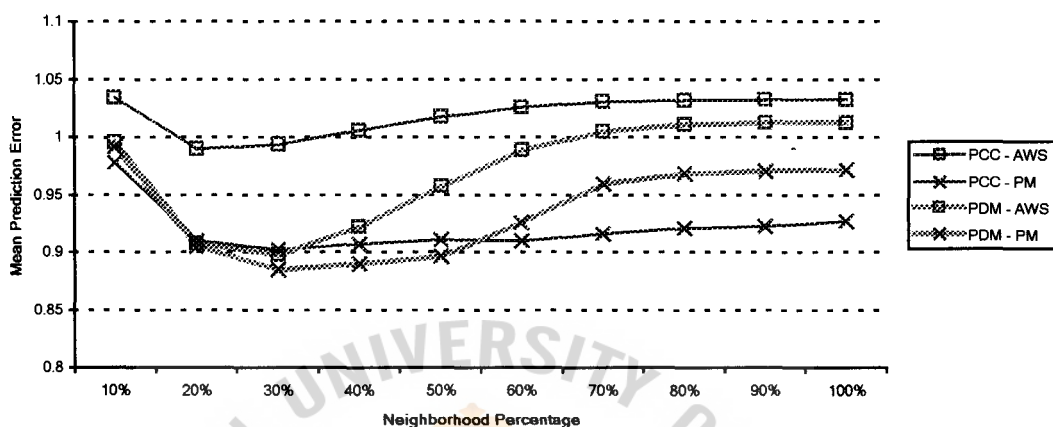
Graph 5.40: Error variance of users with 10 to 250 votes (COR)

Cut off	Similarity Measure	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
10%	PCC	0.415	0.384	0.395	0.386	0.383	0.387	0.394	0.401	0.406	0.405
	PDM	0.456	0.430	0.398	0.388	0.369	0.375	0.431	0.453	0.452	0.460
20%	PCC	0.306	0.297	0.280	0.289	0.291	0.297	0.304	0.302	0.304	0.306
	PDM	0.352	0.330	0.306	0.297	0.296	0.299	0.326	0.337	0.337	0.338
30%	PCC	0.292	0.274	0.277	0.275	0.275	0.276	0.276	0.271	0.273	0.275
	PDM	0.288	0.274	0.274	0.268	0.274	0.277	0.305	0.319	0.323	0.323



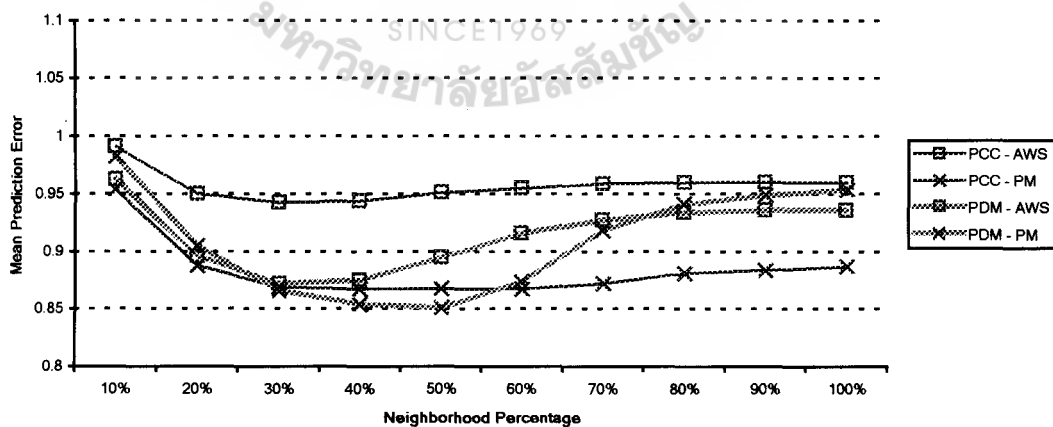
Graph 5.41: Average of Mean Prediction error (2,000 users data set)

Similarity Measure	Predict	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
PCC	AWS	1.035	0.990	0.994	1.006	1.018	1.026	1.031	1.032	1.033	1.033
	PM	0.979	0.910	0.902	0.907	0.911	0.910	0.916	0.921	0.923	0.927
PDM	AWS	0.996	0.908	0.898	0.922	0.958	0.989	1.005	1.011	1.013	1.013
	PM	0.992	0.905	0.885	0.890	0.896	0.926	0.960	0.969	0.971	0.972



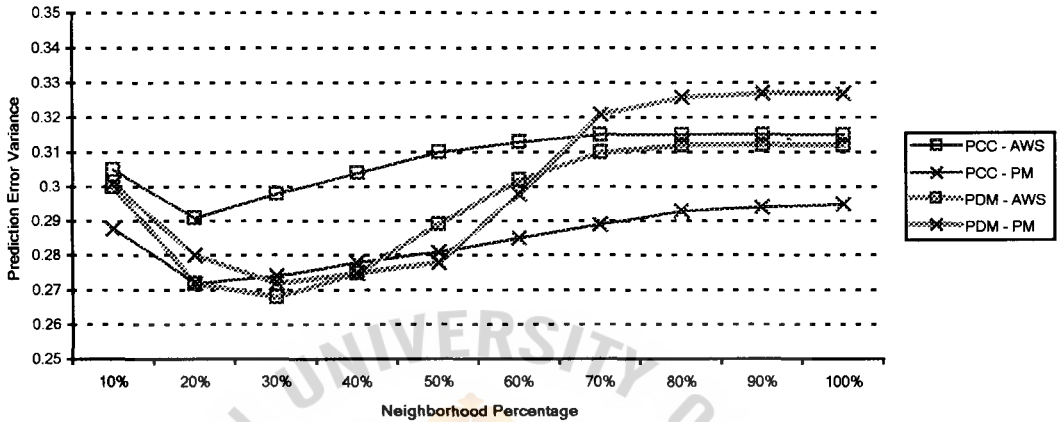
Graph 5.42: Average of Mean Prediction error (500 users data set)

Similarity Measure	Predict	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
PCC	AWS	0.991	0.950	0.943	0.944	0.951	0.955	0.959	0.960	0.960	0.960
	PM	0.954	0.888	0.869	0.867	0.868	0.867	0.872	0.881	0.884	0.887
PDM	AWS	0.963	0.896	0.872	0.875	0.895	0.916	0.928	0.934	0.936	0.936
	PM	0.983	0.905	0.866	0.854	0.851	0.874	0.919	0.941	0.949	0.954



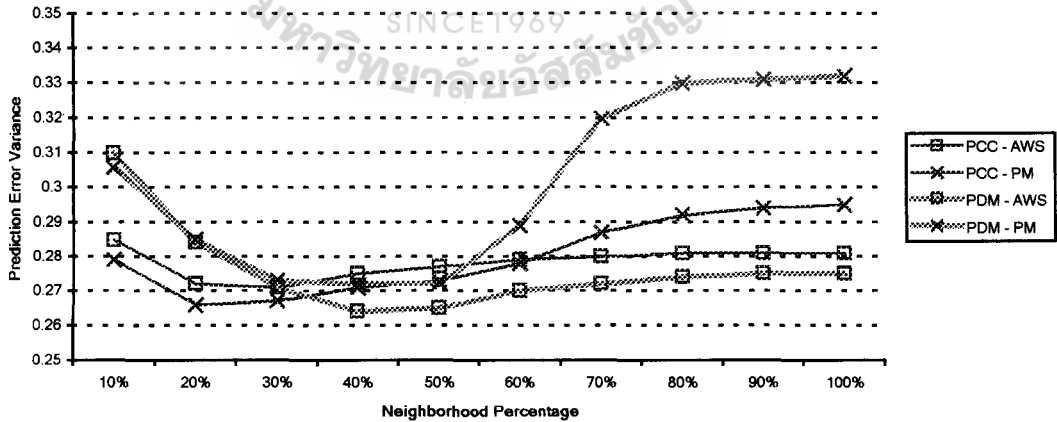
Graph 5.43: Average of Prediction Error Variance (2,000 users data set)

Similarity Measure	Predict	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
PCC	AWS	0.305	0.291	0.298	0.304	0.310	0.313	0.315	0.315	0.315	0.315
	PM	0.288	0.272	0.274	0.278	0.281	0.285	0.289	0.293	0.294	0.295
PDM	AWS	0.300	0.272	0.268	0.275	0.289	0.302	0.310	0.312	0.312	0.312
	PM	0.301	0.280	0.272	0.275	0.278	0.298	0.321	0.326	0.327	0.327



Graph 5.44: Average of Prediction Error Variance (500 users data set)

Similarity Measure	Predict	10%	20%	30%	40%	50%	60%	70%	80%	90%	All
PCC	AWS	0.285	0.272	0.271	0.275	0.277	0.279	0.280	0.281	0.281	0.281
	PM	0.279	0.266	0.267	0.271	0.273	0.278	0.287	0.292	0.294	0.295
PDM	AWS	0.310	0.284	0.271	0.264	0.265	0.270	0.272	0.274	0.275	0.275
	PM	0.306	0.285	0.273	0.272	0.272	0.289	0.320	0.330	0.331	0.332



5.3 Discussions

Both of mean prediction error and error variance have the same trend that they give reasonable result when using 10% neighborhood and gain better result if neighborhood increased to 30%. Beyond this point, the prediction performance dropped. When using average weighted sum prediction method, PDM works better than PCC in every points of neighborhood selection. When using Correlation based prediction method, PDM provides more accuracy than PCC at the optimum performance point (30%-40% neighborhood). However, when neighborhood increases, PDM performance dropped faster than PCC.

Because PDM compares all preferences of two users, higher similarity means that two users mostly prefer the same items. However, lower similarity can mean either dissimilar or not enough information. The case of dissimilar is obvious that two users prefer on different items but when two users give preferences on only a few common items, the weight will become probability of both users that will be similar, which is low. Thus, PDM works well if applied with prediction method that uses information from some similar users such as AWS. Using prediction method that applies for using all users, such as COR, PDM works well only for certain points of neighborhood selection.

From other works discussed in section 2.4, the Pearson correlation coefficient algorithm tends to give good performance on all amounts of preferences. Our evaluation also indicates that not only Pearson correlation coefficient algorithm gives good performance on all amounts of preferences, It performs better when partial users

are used in prediction, 20% to 50% of most similar users gives better result than 100% users used.

In term of ability to predict data when less information known (more percentage of movie votes cut off), both algorithms do not suffer much from increasing percentage of movie votes cut off. However, noticeably the more movie votes cut off the less prediction accuracy achieved.



5.4 Conclusions

The proposed probabilistic distance measure algorithm (PDM) can predict all missing data by using probability of agreement of two users. This algorithm satisfies this study’s goal which is it has high scalability i.e. gives good result not depending on data size. It has high accuracy and reliability that is competitive with Pearson correlation coefficient algorithm. This PDM uses relative pair-wise comparison. The user scoring bias has no effect in measuring two users i.e. it still effectively works if first user that gives score only from 1 to 4, are compared with the other user that gives score only 4 and 6 (we define score as numeric rating from 1 to 6).

The best performance of PDM is achieved when using 30% neighborhood. At this point of neighborhood selection, the measure gains more accuracy and reliability. PDM gives result that has average error at around 0.7 to 1.1, varies by neighborhood selection and known information of user. Error variance takes place around 0.2 to 0.5.

By the assumption that the mean prediction error indicates algorithm efficiency and the variance of error indicates reliability of the results. Optimum performance of mean prediction error and variance of error for PDM and PCC is compared as follows:

Table 5.1: Optimum point of the algorithms

Data Size	Measure	AWS			COR		
		Neighbor	Mean	Variance	Neighbor	Mean	Variance
500	PDM	30%	0.872	0.271	50%	0.851	0.272
	PCC	30%	0.943	0.271	40%	0.867	0.271
2,000	PDM	30%	0.898	0.268	30%	0.885	0.272
	PCC	20%	0.990	0.291	30%	0.902	0.274

(Lower value indicates better performance)

CHAPTER 6: FUTURE WORKS

1. With reference from Breese paper [Breese98] that usage of default voting from no preferences can lower the prediction performance in most of the cases, This study's probabilistic distance measure (PDM) algorithm may works better if the case of no preferences has been cut off (exclude Case 1 in Section 3.1).
2. This study's PDM algorithm has disadvantage on its speed. The main cause is to get similarity between two users the algorithm compares every combination of two movie votes on both users. This can be reduced by only compare it pair wise (after vote 1,2 of both users is compared, skip to vote 3,4). However, the performance of the algorithm is yet to be proved.
3. Improvement can be made if modification of algorithm such as default voting and case amplification are applied.
4. Optimum performance of PDM algorithm depends on how many similar users to pick and apply the prediction method. The optimum formula of similar users amount may be found if we experiment more with other database size and plot a graph between database size and amount of similar users to use.
5. In computing the similarity between two users, we have assumed a uniform distribution over missing scores. This assumption may not hold true. The PDM algorithm could be easily modified to take into consideration non-uniform distributions over movie votes. A database of user preferences could then be analyzed to determine the distribution over movie votes and that distribution could be used when computing similarity of users.

CHAPTER 7: REFERENCES

[Salton83] Gerard Salton, Michael J. McGill. Introduction to Modern Information Retrieval, McGraw-Hill Publisher.

[Rijsbergen79] van Rijsbergen, C. J., Information Retrieval (Second Edition), Butterworths (Boston, MA), 1979.

[Resnick94] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. Proceedings of the 1994 Computer Supported Collaborative Work Conference, page 175-186.

[Eachmovie] <http://www.research.compaq.com/SRC/eachmovie/>

[Swami] Danyel Fisher, Kris Hildrum, Jason Hong, Mark Newman, Megan Thomas and Rich Vuduc. A framework for collaborative filtering algorithm development and evaluation, Computer Science Division University of California, Berkeley.

[Breese98] J.S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Appears in Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, page 43-52, Madison, WI, July, 1998. Morgan Kaufmann Publisher.

[Herlocker99] J. Herlocker, J. Konstan, A. Borchers, J. Riedl. An algorithmic framework for performing collaborative filtering. Proceedings of the 1999 Conference on Research and Development in Information Retrieval, Berkeley, CA, August 1999.

[Vapnik98] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[Sims96] Berkeley Workshop on Collaborative Filtering. March 1996. Links to many collaborative filtering systems. <http://www.sims.berkeley.edu/resources/collab/>

[Hanson00] Ward Hanson, Graduate School of Business, Stanford University, Principles of Internet Marketing, South Western College Publishing.

[Sergios98] Pattern Recognition, Sergios Theodoridis, Departments of Informatics, University of Athens, Greece, Academic Press USA, 1998.

[Platt99] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods---Support Vector Learning*, Jan. 1999.

[Ha99] Vu Ha, Peter Haddawy. Toward Case-Based Preference Elicitation: Similarity Measures on Preference Structures. *Proceeding of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, page 193-201. July 1998.

