

國立臺灣大學電機資訊學院電機工程學系

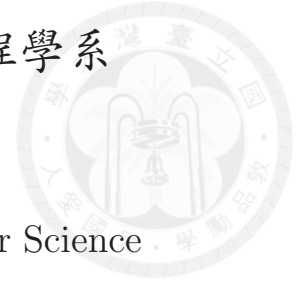
碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis



基於使用者偏好可調概念空間的推薦系統設計

User Preference Based Recommendation System Design with
Adaptive Concept Space

吳潔薇

Jie-Wei Wu

指導教授：于天立 博士

Advisor: Tian-Li Yu, Ph.D.

中華民國 103 年 7 月

July, 2014



致謝

首先謝謝我的指導教授于天立老師，老師給學生們相當大的自由，而在卡關時，又給予相當多專業和有深度的意見，謝謝老師每週咪挺的討論和建議！曾經一度想要放棄，也謝謝老師願意花時間聽我的想法和給我建議，一年前或是兩年前從沒想過我也可以畢業，衷心謝謝老師這幾年的指導！

謝謝我的口委張時中老師、陳穎平老師和鄭卜壬老師。特別謝謝鄭卜壬老師，花了相當多時間和我討論，每次討論都給了我不少建議和信心。還有謝謝台大電機系眾多可敬的教授、厲害的同學及助教，能在電機系認識這麼多好老師和好朋友，我感到自己很幸運。

感謝 TEIL 實驗室一同努力的夥伴。謝謝陳韋名學長，從我碩一開始就一直和我討論、解答我的問題，以及在我心情不好時陪我聊天，我碩二時即使畢業仍然百忙中抽空回來和我討論或幫我看 paper，謝謝學長兩年來各方面說不完的照顧。也謝謝許博竣學長，碩一時常常被我問問題，以及出國時很多小地方的照顧。謝謝邵中昱學長和周志遠學長，在卡關時的鼓勵、在緊繃時又適度讓我放鬆情緒。謝謝顏傳哲學長，碩二準備投 paper 時，常常一起在實驗室待很晚的陪伴。謝謝學長們除了解答我的疑問和互相幫忙，也讓實驗室有很多歡笑和電動，有你們在真的很歡樂。謝謝王士銘，碩一時一起合作從你身上學到不少，碩二也因為一起咪挺而有人能一起討論。謝謝許儲羽，兩位在我有問題時都給我不少建議。還有謝謝學弟許世煥、吳家輝、童宇凡，幫忙寫我檢查 paper 錯誤和一直被我詢問處理 cluster 的事情。最後謝謝蘇倚恩，跟你討論總是有收穫，也謝謝你給我的許多建議。

謝謝張瑋軒、李昆航、和女人迷的夥伴，在實習的期間我在大家身上學習到了很多，也因為在女人迷實習的緣故，也啟發了我一些想法，最後更謝謝女人迷願意將寶貴的資料給予我當成實驗來源，對這份研究幫助很大！

謝謝中華扶輪教育基金會提供的獎學金，讓我升上碩二的暑假能在沒有經濟壓力的狀況下出席國外會議，。

謝謝梁翔勝，謝謝你一直以來都的包容、鼓勵和幫助。

最後謝謝親愛的家人，父母和妹妹，尤其是媽媽，謝謝您在我很忙很少回家的時候給予體諒，對於我的意見和決定都給予尊重，但又在我疲倦時讓我知道，累了永遠都可以回家休息充電。



中文摘要

此篇研究提出了一種推薦系統設計，其中利用了以使用者為基礎的協同過濾、以項目為基礎的協同過濾和基於內容過濾的優點。不同的是，上述兩種以使用者或項目為基礎的協同過濾，其在項目或使用者空間中是高維度的。而基於內容過濾的方式，雖然可以處理協同過濾的冷啟動問題，但對於發覺使用者可能的潛在項目卻較為無效。此篇提出的推薦系統採用了「基於使用者和基於項目的概念空間」，維度的大小，或概念空間的數量，在有必要時才增加，另外該系統能利用產生另一種在文章上的向量維度，以處理冷啟動問題。更多的是，推薦能隨著時間更改演化，在快速增加的訊息下，避免重複的過程是必要的。

概念空間的為度是依照項目的特徵來設立，另外「概念」是項目概念空間分群的結果，也就是文章分群的結果。而使用者空間概念是項目概念空間根據使用者的行為調整後的。譬如說使用者看了某兩個項目，我們便假設這兩個項目之間有關係。這兩個概念空間互相演化，接著系統利用演化最後的「概念」來作推薦。這樣的系統在實驗上實作在文章推薦上面。在我們的例子中，項目是文章，項目特徵是文章的詞，而使用者行為就是讀者閱讀文章的紀錄。

在實驗中，以使用者為基礎的協同過濾和以項目為基礎的協同過濾的為度分別約是三萬和三千，這是根據實驗中的使用者和文章數量決定的。而我們提出的系統，由於使用了「概念」來作，以概念的個數視為維度，其大小從五開始，每個迭代都可能增加，最後在第十二次的迭代收斂，此時維度為八十七。除此之外，所提出的系統能動態調整代表文章的向量維度，文章的維度是用於將文章分群時的依據，文章會依照向量維度來計算相似度，最後向量的長度是四十四，新的文章則可以根據這個維度來加入分群以及被推薦。

精確度-召回率虛線顯示，我們所提出的推薦系統，使用者真正點擊推薦文章的比例，相較於以使用者為基礎的協同過濾、以項目為基礎的協同過濾和基於內容過濾，有更多的點擊比例。另外在平均精確度也可，系統的平均隨著迭代次數增長並超過其他推薦系統。我們希望概念空間的這個想法可以擴展到有可被提取的特徵與用戶之間交互關係的項目。

關鍵詞

推薦系統、協同式過濾、內容式資訊過濾、使用者閱讀行為





Abstract

This thesis proposes a recommendation system (RS) which incorporates the advantages of the user/item-based collaborative filtering (CF) and the content-based filtering. Unlike the user/item-based CF where the user/item spaces are of high dimension, the proposed RS utilizes the user-based and item-based concept spaces where dimension, or the number of concepts, is increased only necessary. In addition, the proposed system can deal with the cold start problem with producing another kind dimension of items. With modifying clustering results, it can be used to create recommendation in the rapid increasing information.

The dimension of the item-based concepts is defined by the features of the items, and concepts are the clustering result of the item-based concept space. The user-based concepts are the result of clustering adjustment from the item-based concepts with the information of users' behaviors, such as whether or not a user is interested in both items in a concept. The user-base and item-based concepts co-evolve iteratively in the above manner. At the end, the proposed RS utilizes the learned concepts combined with the reading dependence to perform recommendation. The proposed techniques are demonstrated on the article recommendation. In this case, the features of an item correspond to the segmented contents of an article, and users' behaviors correspond to users' reading preferences.

In the experiment, the item-based/user-based CF dimension is about 30,000 and 3,000 while the concept space in proposed RS articles starts from 5 and ended up merely 87 after 12 iterations. The proposed RS dynamically adjust the dimension of articles. The dimensions of articles is 44 in the end and used for clustering articles. New articles then can be clustered and recommended as well.

The precision-recall curves indicates that the proposed RS achieves more hits than user-based/item-based CF and content-based filtering. The average precision-recall curves and mean average precision of proposed system grows and exceeds others. This idea of two concept spaces can be extended to the situation with items with extractable features as

dimension and the interaction between items and users.

Keywords

Recommendation system, collaborative filtering, content-based filtering, users' reading behaviors





Contents

致謝	i
中文摘要	ii
Abstract	iv
1 Introduction	1
2 Background	6
2.1 Collaborative Filtering Based Recommendation Systems	6
2.1.1 Overview of CF Process	7
2.1.2 Memory-based CF	7
2.1.3 Model-based CF	11
2.1.4 Challenges of CF	12
2.2 Content-based Filtering	13
2.3 Hybrid Approaches Combined with CF and Content-based Filtering .	15
3 System Design	18
3.1 Symbols	18
3.2 Framework	21
3.3 Concepts	23
3.3.1 Item-based Concepts	24
3.3.2 User-based Concepts	24
3.4 Stages of Operation	24
3.4.1 Keyword Generation Stage	25
3.4.2 Concept Reinforcement Stage	26
3.5 Recommendation	28
3.5.1 Generating Reading Dependence	29
3.5.2 Ranking of Articles	29
4 Experiments and Results	32
4.1 Data Set and Configuration	32
4.2 Evaluation Metrics	33
4.3 Evaluation Process	35
4.3.1 Testing Data	36
4.3.2 Evaluation of Proposed System	36
4.3.3 Evaluation of CF	36
4.3.4 Evaluation of Content-based Filtering	37
4.4 Results and Discussion	37

5 Conclusion

Bibliography





List of Figures

2.1	The collaborative filtering process.	8
2.2	Isolation of the co-rated users in user-based CF.	10
2.3	Isolation of the co-rated items in item-based CF.	10
2.4	The structure of PRES.	15
3.1	An example for the weight w_{a_i, c_j} of a_i in c_j	20
3.2	An example of <i>article graph</i> of Figure 3.1.	21
3.3	The flow of the iterative procedure of the proposed system.	22
3.4	A situation that a concept is needed to be separated.	25
3.5	Max-cut algorithm applied to the complement of <i>article graphs</i> to separate the articles into two smaller concepts.	28
4.1	PR curves of three different prediction.	35
4.2	The average PR curves of proposed system with the interval 0.05. . .	38
4.3	Zoom section of the Figure 4.2.	39
4.4	The comparison of average PR curves.	39
4.5	Zoom section of Figure 4.4.	40
4.6	MAP of proposed systems with different iterations.	40
4.7	The comparison of MAP.	41



List of Tables

3.1	Symbols and definitions.	21
3.2	Each row shows the individual user and his reading sequence.	26
3.3	The transition probability of Tables 3.2. The reading sequence is from the row to the column.	27
4.1	Summary of parameters.	33
4.2	The example of PR curves.	34
4.3	The example of PR curves.	35
4.4	The example of PR curves.	35



Chapter 1

Introduction

Nowadays, the recommendation systems (RS) have been widely used to identify items that customers are potentially interested in. Based on the history of personal behaviors, such as purchasing and browsing, RS computes the ranks of items so that customers might select items with the help of such information. Recommendation systems are built in wide area, product, videos, books, new, articles and so on [5]. For customers, it is helpful to save the effort on searching items. For business like e-commerce sites, an effective RS finding the customers' preference of products might help increase the benefit.

Two commonly used techniques for building RS are the collaborative filtering (CF) [10, 14, 19] and content-based filtering [1, 28]. For example, the famous e-commercial platform, Amazon.com, has used the item-based CF to recommend books or other products for customers [15]. CF algorithms create recommendation based on the rating history of users. In other hand, the content-based filtering recommend items according to the content of items.

The user-based CF first finds similar customers based on the rating of the user profiles. If two customers both highly or lowly rate the same item, the similarity between these two customers are high. CF predicts the rating that a user would give to an item and recommends potential items not rated by the user if the items are highly rated by other similar customers. For the item-based CF, the similarity of particular two items are computed according to the users who both rate these two items. However, the computational burden of item-based and user-based CF is heavy since they both operate on item and user spaces, and the dimension of both spaces grow rapidly when new customers or items are emerged. For example, we assume ten thousand customers and one hundred thousand items have existed in the system. For user-based CF, when more customers are added to the system, the system have to compute the ten multiplied by ten thousand times to compute

the new customers. The effort of comparing any two users is according to the size of item space, which is one hundred thousand. Furthermore, next time ten more new customers means the more computation, which would be ten multiplied by one hundred thousand. For item-based CF, we assume that ten more items are added. The similarity of items need to be recalculated about ten multiplied by one hundred thousand times to renew the most similar items of each item. The same, when compare two items, the length of user space is according to the number of users, which is ten thousand in our example. As a result, each time when new customers or new items are included, the effort of computation, also the dimension, grows. The growth scale of computational burden of system is quite huge.

Moreover, both CF suffer from sparsity. Sparsity means the rating is less and not enough to provide strong recommendation. For example, the number of items sold on major e-commerce sites is extremely large. We assume that the number of items is about ten million. Even the most active customer, who rates about one thousand items, can only rate a small portion of the overall items. As a result, even the most popular items get very few ratings. With CF, it is difficult to compute the relations of users to find neighbors of target users and also hard to know the similarity of items.

The model-based CF can somehow resolve the problem of scalability and sparsity. It builds a model according to different methods of machine learning or data mining. However, since CF does not consider the item of any items, they suffer from the cold start. Cold start happens when a new item or new user comes into the system. Since there is no user has rated that item or the new user has not rated any item, there is not enough information to recommend or to be recommended.

Content-based filtering is another kind of RS. With comparing the content and the user profile to recommend. The user profile is built with the same space of items. For example, the same terms when documents are items. Several methods are used to learn the user profile, relevance feedback, for example. However, content-based filtering can not explore different way of items not rated by users. It often results in recommending the items similar to seen items and ignores the other kinds of items which users might be interested in. It processes without the fact that a user can be interested in more than one kind of items.

Several previous work also intends to combine CF and content of items. Various models and content are used. However, with the rapid information nowadays, the recommendation created from the whole history requires lots of time, and it might not that necessary to repeated the process. If the recommendation can be modified from a particular time stamp and then keeps going. It might be somehow save the

efforts of the unnecessary burden.

From now, we have talked about some issues of CF, content-based filtering, and hybrid approaches on the field of RS. This thesis investigates RS from the point of view of the problems mentioned before. The proposed RS combines both the advantages from both user-based/item-based CF and content-based filtering by introducing the user-based and item-based concept spaces. To deal with the issue of sparsity and scalability, the proposed RS intends to utilize the benefit of clustering. A cluster is regarded as a concept in this thesis. The potential interest of a customer on an item is translated into the interest on a concept. That is, with the help of clustering, an item is recommended from the level of interest for a customer on a concept and the weight of the item in the concept. In traditional clustering algorithms such as K-means [8, 9], each observation is represented by a vector, and the distances between observations can be computed in some way, Euclidean distance, for example. The results of clustering are the item-based concepts. The recommendation from item-based concepts intends to deal with the cold start problem.

Moreover, the approach of clustering also considers the customers' behaviors. In addition to consider the distance, or similarity, between items via vectors, the users' preference history is used to construct the relation of items and affects the clustering. The modified item-based concepts are the user-based concepts.

When talking about the issue of dimension growth when new customers or items added to system, the clustering approach intends to reduce the dimension. Compared to item-based CF, which computes the distance between new items and each existed item, the proposed RS requires that each new item only been compared to each concept with computing the similarity and then been included to each concept with different level. Different from user-based CF, when new users are included, the proposed RS computes the level of interest of each concept for a new user according to the items the user rates. The dimension of both concept spaces are considerably lower than that of user-based/item-based CF. In both spaces, a concept consists of several items, or in other words, a group of items represents a concept.

This thesis proposes two type of concept spaces, the item-based concepts and the user-based concepts. The major difference is that the clustering of item-based concepts is based on the contents of items, while that of user-based concepts is based on users' preferences. Starting with or without initial item-based concepts given by editors, our RS iteratively refines concepts by increasing the size of the concept spaces. Our RS aims to achieve better article recommendation with the better representation of items and thus forming better concepts. The recommendation is based on the following two factors: (1) *level of interest* that a user shows to a

concept, and (2) the reading dependence of articles that a user reads an article after another. These two factors are combined with cosine similarity [22] to recommend articles. Apart from initial concepts given by editors, which might be rough, the reinforced concepts with not only content-similarity but also user-similarity are more specialized for readers and articles.

Thesis Objectives

The objectives of this thesis are described as follows.

1. Proposing approaches combined with CF and content-based filtering to deal with the problems they face.
2. Using users' access patterns to refine the concepts and via the iterative process to better reinforce the concepts.
3. Extracting *keywords* as dimension of items from the iterative procedure with the content of items and clustering the new articles with *keywords* to deal with the cold start problem.
4. Shedding lights on the future developments on the RS design for items with extractable features and the user access patterns.

Roadmap

This thesis is structured as follows.

Chapter 2 presents the background knowledge about recommendation system for this thesis. Several kinds of CF are elaborated, including the memory-based and model-based CF. We first show the CF algorithm process then the detail of memory-based and model-based Cf. Some challenges associated with the memory-based approaches are presented then. Then the content filtering is briefly described and followed by the dependence modeling.

Chapter 3 describes the proposed recommendation system design and how this design is used on article recommendation. In the structure, an iterative process is designed and composed of two concept space and two operation stages. The two kind of concepts are named the item-based and user-based concepts. They are elaborated and followed by the two operation stages, keyword generation stage and concept reinforcement stage.

Chapter 4 describes the experiments and discussion. With the logs of users' reading history, the experiments show the result of RS mentioned in Chapter 2 and the system constructed with the design structure. This chapter provides details of data sets, evaluation methodology and results of different experiments and discussion of the results.

Chapter 5 first gives a summary of this thesis. At the end, several possible directions to extend the work are presented.



Chapter 2

Background

In this section, we briefly present some of the research on recommendation systems (RS) that are closely related to this thesis. RS typically produces a list of recommendations. Most recommendation systems focus on the task of information filtering, which deals with the delivery of items selected from a large collection that the user is likely to find interesting or useful. Recommendation systems are special types of information filtering systems that suggest items to users. Some of the largest e-commerce sites are using recommendation systems and apply a marketing strategy that is referred to as mass customization. There are two main approaches to information filtering: collaborative filtering (CF) and content-based filtering. Traditional CF selects items based on two kinds of similarities. One is between the preferences of different users and called user-based. The other is the similarity of items and called item-based. CF merely utilizes the rating history of user and does not consider the content of items. The content-based filtering, in opposite, recommend items for users with the comparison of the content of users and the user profile. Next, previous work combined CF and content-based are briefly introduced. These approaches intend to utilize both advantages of CF and content-based. For example, one of the challenges of CF is cold start, and it can be somehow conquered by analyzing the content of items. However, several trade-offs exist. The most common disadvantages are the increased complexity and high expense for implementation [23].

2.1 Collaborative Filtering Based Recommendation Systems

The basic intuition of CF is that it provides item recommendation for one user according to other like-minded users. Collaborative filtering builds a model from past behaviors, activities or preference of users, for example, the items previously pur-

chased or selected. In explicit data collection, the users are required to rank the items. The ratings matrix is the input of CF algorithms, and the output is a set of scores for unseen items. Two widely used categories of collaborative filtering are the memory-based CF and the model-based CF. With memory-based CF, the entire recommendation process generally is an on-line process while for model-based CF, the recommendation is performed using an aggregated model. The time-consuming part, building the model with training data, is off-line, leaving the on-line recommendation process with low time complexity.

2.1.1 Overview of CF Process

Based on the user's previous rating and the opinions of other like-minded users, a typical CF recommends new item for users [11]. In CF, a set of m users and a set of n items exist, and each user has a list of items which the user has rated or other records like pursuing goods. A user for whom the task of CF is to find items are called *active users*. Opinions of other likely-minded users can be provided explicitly through rating-scores which generally are within a certain numerical scale or implicitly from the logs by analyzing records like web mining [13,25]. The items for the active user can be one of two forms: prediction and recommendation.

Prediction is a numerical value and expressed as the predicted likeliness of a certain item for the active user. This predicted value is within the same scale.

Recommendation is a list of N items that the active user will like most. The items on the list should not have been rated by the active user. This way is also known as *Top- N recommendation*.

Figure 2.1 shows the schematic diagram of the collaborative filtering process. [19] Totally n items and m users exist and u_a represents the active user. The input is a $m \times n$ ratings matrix. Each entry represents a preference score and is within a numerical scale. The results are the prediction of item i_j or the N items recommendation.

2.1.2 Memory-based CF

Memory-based CF is the most general CF. Two major kinds are the user-based CF and the item-based CF [2,15,19]. This mechanism uses user rating data to compute similarity between users or items and generates a prediction or recommendations. The user-based and item-based CF are briefly introduced. Then, the mechanisms of

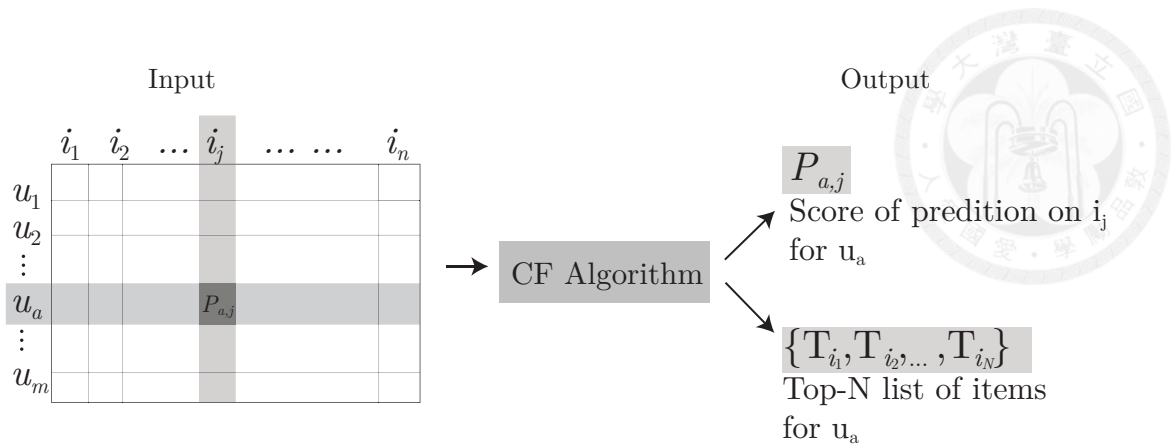


Figure 2.1: The collaborative filtering process.

similarity computation and the approaches of prediction and recommendations are described.

User-based CF

In the user-based CF, first all users are weighted respect to similarity with the active user. The similarity computation will be described lately. Generally, those users who are considered similar often like the items which are high rated by active user or rate item with low score same as active user. After similarities are computed, a subset of users are select to be the predictors, or called nearest neighbors. These predictors are applied to produce the weighted combination with normalized ratings and finally comes the prediction.

Item-based CF

In item-based CF, the main idea is to analyze the user-item matrix and identify the relations between items. The intuition behind this approach is that a user would be interested in purchasing items that are similar to the items the user liked before and would tend to avoid the items the user did not like earlier. The similarity between items depends on the co-rated score. Figure 2.3 shows a ratings matrix. When computing similarity $s_{x,y}$ of two items, only the co-rated scores are considered. The entry marked R means the user rates the item. This means that if any two items are not rated by a same user, the similarity of the two items cannot be computed. For two items which are both rated high or low from the same users, the similarity of these two items are high.

Similarity Computation

The memory-based CF processes with the help of computing the similarity of users or items. The most used two similarity computation are as follows.

Cosine-based Similarity The items or users are represented as vectors in the m -dimensional user space or n -dimension item space. The similarity of two items/users can be thought of the cosine of the angle between these two vectors. For two items/users x and y , the similarity can be given as

$$sim(x, y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|}.$$

Correlation-based Similarity Also known as Pearson correlation similarity, this method measures the similarity according to the correlation coefficient. For user-based/item-based CF, only the co-rated cases are considered. That is, for item-based CF, two items i_x and i_y , we only consider the users who both rated these two items. On the other hand, for user-based CF, the co-rated cases are the items that are both rated by the two users which are also denoted as u_x and u_y . The set of these users or items are denoted as S . Figures 2.2 and 2.3 shows the diagrammatic graph for S in user-based/item-based CF. In Figure 2.2, S is the set of i_1, i_2, \dots, i_m for user-based CF and the set of u_1, \dots, u_m for item-based CF shown in Figure 2.2. To simplify, x and y are denoted as two users or items in user-based or item-based CF respectively.

$$sim(x, y) = \frac{\sum_{s \in S} (R_{x,s} - \bar{R}_x)(R_{y,s} - \bar{R}_y)}{\sqrt{\sum_{s \in S} (R_{x,s} - \bar{R}_x)^2} \sqrt{\sum_{s \in S} (R_{y,s} - \bar{R}_y)^2}}$$

where $R_{x,s}$ means the rating of user x on item s for user-based CF or the rating of user s on item x in the case of item-based CF. \bar{R}_x is the average rating of item x in item-based CF or the average of the x user's ratings in user-based CF.

Prediction and Recommendation Computation

Prediction and recommendation are the final output of a CF algorithm. How to compute these final results are listed and described as follows.

Weighted Sum of Ratings In user-based CF, to make a prediction for the active user a on a certain item i , we can take a weighted average of all the ratings on that item according to the following formula

$$P_{a,i} = \bar{R}_a + \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_u) \cdot sim_{a,u}}{\sum_{u \in U} |sim_{a,u}|}, \quad (2.1)$$



	i_1	i_2	i_n
u_1								
u_2								
⋮								
u_x	R	R	R	-		R		R
⋮								
u_y	R	R	-	R		R		R
⋮								
u_m								

Figure 2.2: Isolation of the co-rated users in user-based CF.

	i_1	i_2	...	i_x	...	i_y	i_n
u_1				R		R			
u_2				R		-			
⋮				-		R			
⋮				R		R			
⋮									
⋮				-		R			
u_m				R		R			

Figure 2.3: Isolation of the co-rated items in item-based CF.

where $P_{a,i}$ is the prediction of user a on item i , \bar{R}_a and \bar{R}_u are the average rating for the user a and user u on all rated items, and $sim_{a,u}$ means the similarity between the users a and u . The summations are deviated all the users $u \in U$ who have rated the item i . U is the set of all users. [23]

In item-based CF, assume the similarity of two items i and j is $sim_{i,j}$, the prediction $P_{a,i}$ of user a and item i is given by

$$P_{a,i} = \frac{\sum_{n \in N} sim_{i,n} \times R_{a,n}}{\sum_{n \in N} |sim_{i,n}|}, \quad (2.2)$$

Where N is the set of all similar items of item i and $R_{a,n}$ is the rating of active user on one similar item n . Only the items that the active user has rated are considered. [19].

Top-N Recommendations

In addition to exact scores on items, another output of CF algorithms is to provide a list of N top-ranked items which have the highest potential to be liked by the active user. The rank is also the sequence of recommendation. In user-based CF, the CF first select the several nearest neighbors using the cosine-based similarity or Pearson correlation. Each user is treated as a vector in the m -dimensional item space and the similarities between the active user and other users are computed between the vectors. Then, the ratings of these selected users are aggregated as a set. The mechanism then selects the items which have been rated higher in the set from the neighbors, and the chosen items should not have been rated by the active user. In item-based CF, according to the items that the active user has rated, the items can be computed via a weighted sum according to Equation 2.2. Then the sequences of items are ranked with the scores.

2.1.3 Model-based CF

Apart from memory-based CF, model-based CF builds a model of user ratings with data mining, machine learning algorithms [24] with the training data then provide item recommendations. It takes a probabilistic approach and computes the expected value of the rating of a user's prediction on an item when given the user's ratings on other items. The model building process is performed by different machine learning algorithms such as Bayesian network, clustering, and rule-based approaches. Model-based CF algorithms have been investigated to solve the shortcomings of memory-based CF algorithms. There are several approaches about model-based CF to build model. They are briefly described and list as follows.

The Bayesian network model [2] formulates a probabilistic model for a collaborative filtering problem. In the resulted network, each item will have a set of parent items that are the best predictors of its votes.

The clustering model regards collaborative filtering as a classification problem [1, 2, 4, 27]. It works by clustering similar users in the same class and estimates the probability that a particular user belongs to a particular class. Then the conditional probability of ratings is computed from the probability of being a particular class. In most situations, clustering is an intermediate step and the resulting clusters are used for further analysis or processing to conduct classification or other tasks.

The rule-based approach applies association rule discovery algorithms to find association between co-purchased items. Then it generates the list of item recommendation based on the strength of the association between items [18].

2.1.4 Challenges of CF

The value of recommendation systems is providing fast and accurate potential items. The well-performed RS improves the customer consuming and bring business benefit. Therefore, how RS deal with the shortcoming of the system is important. For both memory-based CF and model-based CF, limitation lies. Though memory-based CF is more easier to implement compared to model-based Cf, and new data can be added easily and incrementally, it is more likely to face challenges of scalability and data sparsity. In other hand, it seems that model-based CF are able to deal with the scalability and sparsity. Since the model is built off-line, the performance of on-line prediction is better. However, the model building mechanism is often high time complexity and might lose useful information for dimensionality reduction techniques. Both CF face the challenge of cold start since none of them considers the content of an item. Overall, the challenges are discussed as follows.

Scalability

In memory-based CF, the computation of nearest neighbor algorithm grows rapidly with the number of users and the number of items. For CF applied on E-commerce, the environment faces the problem of large scale, especially for large on-line shopping companies like eBay and Amazon. Therefore, the scalability is an issue that RS must faces. More precisely, the complexity is $O(n^2m)$ on the worst case of item-based CF and $O(nm^2)$ of user-based CF. When both n and m go beyond million, the computation time is about 31 years with 10^9 times per second.

Data Sparsity

Sparsity occurs when the item set is large, which often occurs for commercial issues. The scale exceeds the tolerance ability of the system. For example, the number of items is ten million. For a user who rate one thousand items, the proportion of whole items is only 0.01%. In this systems, even the most active user would only select or rate under 1% of the items. Then the recommendation becomes not that reliable for the reason that the “nearer” neighbors or more similar items might not found.

Cold Start

When a new user or new item is just included into the system, the information is not enough. For new item, it can not be recommended to any user since no user has rated it. Then the similarities between it and other items are merely zero. It is also not in the aggregated set of the neighbors of active user. Surely the system does not recommend this item to any user. For a new user, the similarities between this user and any other users can only be zero since no co-rated items. Also, the new user has not rated any item therefore no comparison of rated items and unseen items. The direction of item-based CF also fails. New items can not be recommended until some users rate it, and new users can not have recommendation before rating some items.

2.2 Content-based Filtering

Quite different from CF, content-based filtering recommends items based on a comparison between the content of items and a user profile. A content-based filtering system selects items based on the correlation between the content of the items and the user’s preferences as opposed to a collaborative filtering system that chooses items based on the correlation between people with similar preferences. The content of each item is represented as a set of descriptors or terms, typically the words in a document. With analyzing the content of items which have been seen by the user, the user profile is represented with the same terms and built up. Several issues have to be considered when implementing a content-based filtering system. First, terms can either be assigned automatically or manually. When terms are assigned automatically, we should choose a method so that we can extract these terms from items. Second, the terms have to be represented such that both the user profile and the items can be compared in a meaningful way. Third, a learning algorithm has to be chosen that is able to learn the user profile based on seen items and can make recommendations based on this user profile.

The information sources that content-based filtering systems are mostly used with are text documents. A standard approach for term parsing selects words from documents. The vector space model (VSM) [17] and latent semantic indexing [7] are two methods that use these terms to represent documents as vectors in a multi-dimensional space. In VSM, a document D is represented as a vector, which dimension is composed of terms. The weight, or the importance, of each term can be determined by the *tf-idf* scheme. With this way, the weight of a term in a document is decided according to two factors. One is how often the term appears in this document, and the other is how often the terms appears in the whole documents. Generally, the more time the term appears in a document and the less time the term in other documents increases the weight. The formula is $w_i = tf_i \cdot \log(n/df_i)$ where tf_i is the number of occurrence of term t_i in document D , n is the number of the documents in the collection and df_i is the number of documents in which term t_i appears at least once. If document D does not contain term t_i , the weight is zero.

Relevance feedback, genetic algorithms, neural networks, and the Bayesian classifier are among the learning techniques for learning a user profile. The user profile is represented with the same terms and built up by analyzing the content of documents that the user found interesting. VSM and latent semantic indexing can both be used by these learning methods to represent documents. Some of the learning methods also represent the user profile as one or more vectors in the same multi-dimensional space which makes it easy to compare documents and profiles. For example, let $P = (u_1, \dots, u_k)$ be the profile vector, and the cosine similarity can be written as

$$sim(D, P) = \frac{D \cdot P}{\|D\| \|P\|}. \quad (2.3)$$

An example of the content-based filtering is Personalize Recommendation system (PRES) architecture [28]. It categorizes the page of a web site into three categorizations, content page, navigation page and hybrid page. Content pages include the information and items a user interested in, and the navigation pages help the user to search for the interest items. The hybrid pages both provide content as well as navigation facilities. In addition, a content page for one user might be a navigation page for another. Therefore, each user has their own structure. The structure of PRES is shown in Figure 2.4. The recommender system compares the user profile with the documents in the collection. The documents are then ranked on the basis of certain criteria such as similarity, novelty, proximity and relevancy and the best ranked documents appear as hyperlinks on the current web page. A user profile is learned from feedback provided by the user. The feedback includes the documents a user selects and reads, and this provides a strong indication that the document

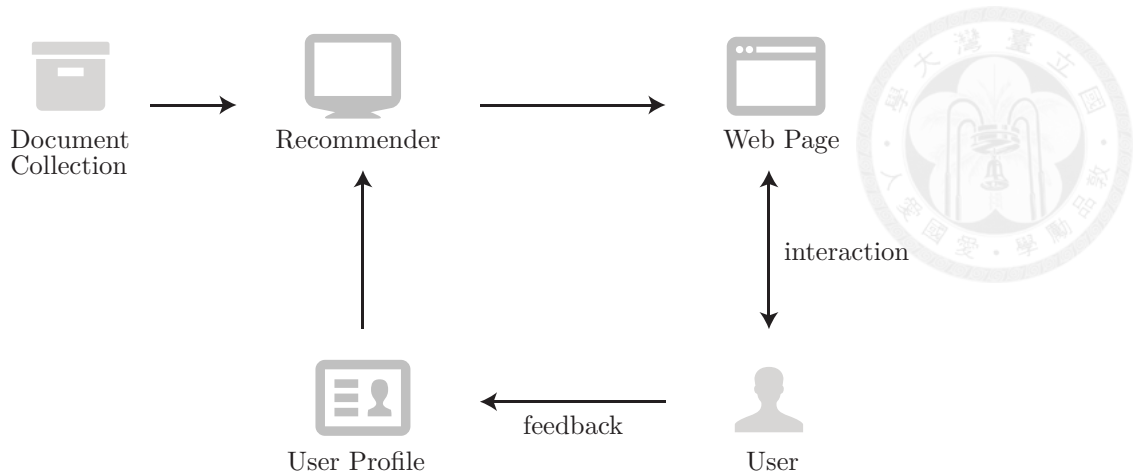


Figure 2.4: The structure of PRES.

contains information a user is interested in. What's more, negative examples are not used since it is hard to recognize the negative actions. For example, the reason for the short reading time for a page might be just the document is short or information provided less new to the user. The links the user does not click do not provide strong evidence as well. The reason is that the user might visit the document later or just does not notice the link. Another important impact is that the user is interested in a topic for a short period since once enough information has been provided the user will lose interest in that topic. In short, the user model has to be dynamic and learned from positive examples only. PRES uses relevance feedback to learn user profile model. After the user reads document D , a user profile P would be updated as $P' = \alpha P + \beta D$.

Though content-based can somehow deal with the cold start problem of CF with the content of items, several issues emerge. The major issue is that users are interested in manifold topics even those topics do not much intersect in content. Therefore, pure content-based filtering lower the exploration of potential items of which are not much related to those have been highly ranked by active user.

2.3 Hybrid Approaches Combined with CF and Content-based Filtering

In addition to Cf and content-based filtering, approaches combined with these two have been studied and reveal benefit [3]. CF has been mentioned about the challenges of cold start and content-based filtering alone can prove ineffective. Content-

based techniques have difficulty in distinguishing between high-quality and low-quality information that is on the same topic. Also, content-based filtering faces the problem of recommendation depends on too much of the active user merely. In this section several previous work combined with both CF and content-based are described.

In [3], they propose a web recommendation system in which the basis of CF and content-based filtering are kept separated. A page, or an item, is represented as a vector which attributes as dimension. The number of attributes is the length of the dimension. After defining a set of features and generating constraints for each feature, it is guaranteed that, under all the constraints, a unique distribution exists with maximum entropy [12]. Each source of knowledge can be represented as features with associated weights. In their model, two sources of knowledge about Web users' navigational behavior are namely *features*. The overall features are combined to provide the recommendation. Two sources of information are considered to be the features. One is based on item-level usage patterns and the other is based on item content associations. With the usage patterns, the condition probability of a certain page is used to decide the value of features. The probability are decided regarding to the user ration history. In other hand, the attribute selection method is modified from Latent Dirichlet Allocation. Then they use Variational Bayes technique to estimate each item's association with multiple "classes", or "topics". Since they find that each item shows strong association with one "class", they assign each item to a single class and then define the values of features.

Latent Dirichlet Allocation is also mentioned in [29] as topic model. In this work, they intend to solve the problem when searching scientific articles, finding relevant paper is difficult in the large on-line archives of scientific articles. They propose the approach combining the merits of traditional collaborative filtering and probabilistic topic modeling in the direction of utilizing the article libraries created by users. The topic modeling is used to generalize the unseen articles through providing a representation of the articles in terms of latent themes discovered from the collection. The topic representation of articles allows the algorithm to make meaningful recommendations about articles before anyone has rated them. In short, an article that has not been seen by many will be recommended based more on its content while an article that has been widely seen will be recommended based more on the other users. Both user and items share the same latent low-dimensional space and are represented by a latent vector. With latent factor models, collaborative topic regression is proposed and represents users with topic interests and assumes that documents are generated by a topic model. Each document comes with an topic

proportion. Then from the topic proportions and the interest of use, collaborative topic regression provides recommendation.

Though approaches combined with CF and content of items reveal some way to work. The rapid growth of information results in the need of recommendation changeable with time. A mechanism is needed to adapt recommendation according to the sequence of history. However, if new recommendation is just produced from more records of history, the effort is expensive because of somehow repeated effort. That is, “new” recommendation modifies from the “past” recommendation with considering only the more recent history instead of all history to save off-line computation burden.



Chapter 3

System Design

Until now we have talked about the motivation of the proposed system and some background of the previous work of RS. Considering the factors mentioned before, this thesis works on proposing a RS design to deal with the challenges of CF, content-based filtering and the dynamic recommendation. This chapter describes how we design our system, and in this thesis, the design is adapted to the reading relations. Articles are the items and readers are the users. To describe the structure of the design on detail, the symbols and their definitions are given at first. Then, the framework introduces each component of the system briefly and how this procedure processes. The details of each concept and stage are explained to further clarify the procedure. Finally, the approach of ranking for article recommendation is revealed.

3.1 Symbols

In this work, we want to find the relations between articles with reading behaviors of users and to deal with some challenges of CF through grouping articles. Therefore, three major symbols for articles, users and concepts are first revealed. A concept means a group of article. Let \mathbf{A} be the set of articles, \mathbf{U} be the set of users and \mathbf{C} be the set of concepts. The symbols $a_i \in \mathbf{A}$, $u_k \in \mathbf{U}$ and $c_j \in \mathbf{C}$ respectively indicate an article, a user and a concept.

For a particular concept, we consider that the importance of each article in this concept is not always the same. The reason is that the page views of different articles might be different. When two articles in this concept are compared, the article receiving more page views is regarded more important than the other. Therefore, the articles in a particular concept are evaluated according to the page views they receives. However, only the page views of some kind of users are cared. That is, another issue we need to talk about is the users who read any article in the

particular concept. If a user reads an article in a concept, he or she is considered to be interested in this concept. The weight of an article in a concept increases while more users who are interested in this concept read the article. Let \mathbf{W} denote the set of weight of articles and $w_{a_i, c_j} \in \mathbf{W}$ represent the weight of a_i in c_j . c_j is a concept including a_i .

To get \mathbf{W} , three factors are related. First, for a particular concept, c_j , we only consider the articles in this concept. As a result, a symbol g is used to find the articles which are contained in c_j and defined as

$$g_{a_i, c_j} = \begin{cases} 1 & \text{if article } a_i \text{ contributes to concept } c_j \\ 0 & \text{otherwise.} \end{cases} \quad (3.1)$$

Second, only the users who are interested in c_j are considered. The way is to filter the user who does not read any article included in c_j since the user does not read an article should not have impact on this article. Therefore, we define a symbol r to check whether or not a user reads article, and r is defined as

$$r_{u_k, a_i} = \begin{cases} 1 & \text{if } u_k \text{ reads } a_i \\ 0 & \text{otherwise.} \end{cases} \quad (3.2)$$

Next, if u_k reads any article in c_j , he or she is considered to be interested in this concept. The number of users interested in c_j is concluded to estimate $w_{i,j}$. A symbol b represents a user who is interested to c_j and defined as

$$b_{u_k, c_j} = \begin{cases} 1 & \sum_{a_i \in A} r_{u_k, a_i} \times g_{a_i, c_j} \geq 1 \\ 0 & \text{otherwise.} \end{cases} \quad (3.3)$$

Finally, the weight of a_i in c_j is defined as

$$w_{a_i, c_j} = \frac{\sum_{u_k \in U} r_{u_k, a_i} \times g_{a_i, c_j}}{\sum_{u_k \in U} b_{u_k, c_j}}. \quad (3.4)$$

The nominator is the number of users who read a_i , and the denominator is the number of total users who are interested in c_j . Since the number of users who read an article is no more than the summation of all users who are interested in a concept, w_{a_i, c_j} is in the range of $[0, 1]$. We can use an example to explain more (Figure 3.1). The edge between an article and a user implies the user reading the article. We can see four users are interested in the concept, c_1 , since each of them is linked to more than one article in c_1 . Obviously four users reading more than one article composes this concept. According to Equations 3.1 to 3.4, weights of each article in c_1 can be computed. We get w_{a_1, c_1} , the weight of a_i grouped into c_j is $\frac{3}{4}$. w_{a_2, c_1} , w_{a_3, c_1} and w_{a_4, c_1} are $\frac{2}{4}$, $\frac{2}{4}$ and $\frac{1}{4}$ respectively.

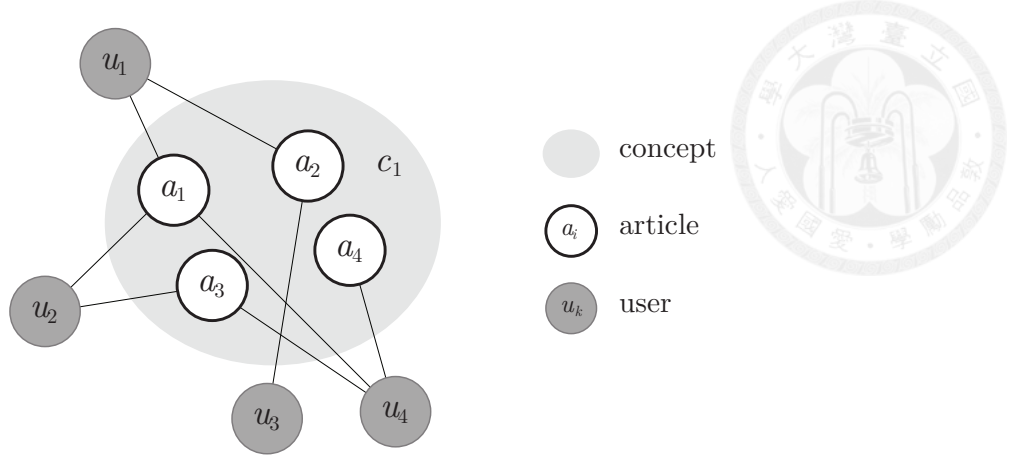


Figure 3.1: An example for the weight w_{a_i, c_j} of a_i in c_j .

After the weight of articles are decided, we need to further find the extent of user to each concept. Here we call it *level of interest* of users shown to each concept. The users reading different numbers of articles are regarded to have different influences when refining the concepts. In addition, the weight of articles also matters. *Level of interest* is defined according to two factors. One is the number of articles a user reads in the particular concept. If the user reads more articles in the concept, this user is considered to be more interested in this concept. The other is the importance, weight, of articles which is read by user. In other words, a user is considered to be more interested in a concept if he or she reads an article with higher weight than another article with lower weight in this concept. As a result, the *level of interest* for user u_k to the concept c_j is defined as

$$l_{u_k, c_j} = \frac{\sum_{a_i \in A} r_{u_k, a_i} \times w_{a_i, c_j} \times g_{a_i, c_j}}{\sum_{a_i \in A} w_{a_i, c_j} \times g_{a_i, c_j}}. \quad (3.5)$$

The denominator is the summation of weights of all articles in c_j , and the nominator is the total weight of articles in c_j . This means if the user reads more articles or reads the articles with higher weight, he or she is considered to be more interested in the concept. Since the nominator is no more than the denominator, l_{u_k, c_j} is in the range of $[0, 1]$.

Next we introduce *article graph*. *Article graph* is constructed with the nodes representing the articles and the edges denoting relations between two articles. We consider how to define the relations between articles with users' reading behaviors. We believe that two articles are connected, which means they probably form the same concept, when one or more users read both articles. If more users read both of them, the relation is considered stronger. In addition, *level of interest* of the user for the concept affects the relations. The user who is more interested in a concept

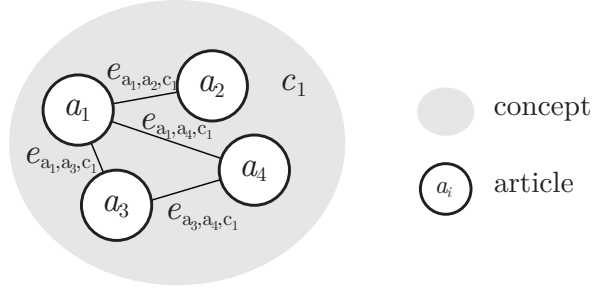


Figure 3.2: An example of *article graph* of Figure 3.1.

influences more on the relations between articles in the concept. Thus, the relation between two articles a_i and a_m is defined as

$$e_{a_i,a_m,c_j} = \sum_{a_i,a_m \in A} l_{u_k,c_j} \times g_{a_i,c_j} \times g_{a_m,c_j} \times r_{u_k,a_i} \times r_{u_k,a_m}, \quad (3.6)$$

which is the summation of *level of interest* for all users who are interested in c_j . The *article graph* of Figure 3.1 is shown in Figure 3.2. The edges between two articles exist when one or more users reading both of the two articles. In this example, $e_{a_1,a_2,c_1} = l_{u_1,c_1} = \frac{5}{8}$ and $e_{a_1,a_3,c_1} = l_{u_2,c_1} + l_{u_4,c_1} = \frac{5}{8} + \frac{6}{8}$ according to Equations 3.5 and 3.6. Table 3.1 summarizes the symbols and their definitions.

Symbol	Definition
g_{a_i,c_j}	whether or not a_i consists of c_j
r_{u_k,a_i}	whether or not u_k reads a_i
b_{u_k,c_j}	whether or not u_k is interested in c_j
w_{a_i,c_j}	the weight of a_i in c_j
l_{u_k,c_j}	<i>level of interest</i> for u_k toward c_j
e_{a_i,a_m,c_j}	the relation between two articles a_i and a_m in concept c_j

Table 3.1: Symbols and definitions.

3.2 Framework

After symbols have been described, the picture of system structure is revealed. The iterative procedure is composed of two concept spaces and two stages. Before we look into each concept space or stage, the structure and procedure are first described. The system is built on an iterative process with concepts refining in each turn. The

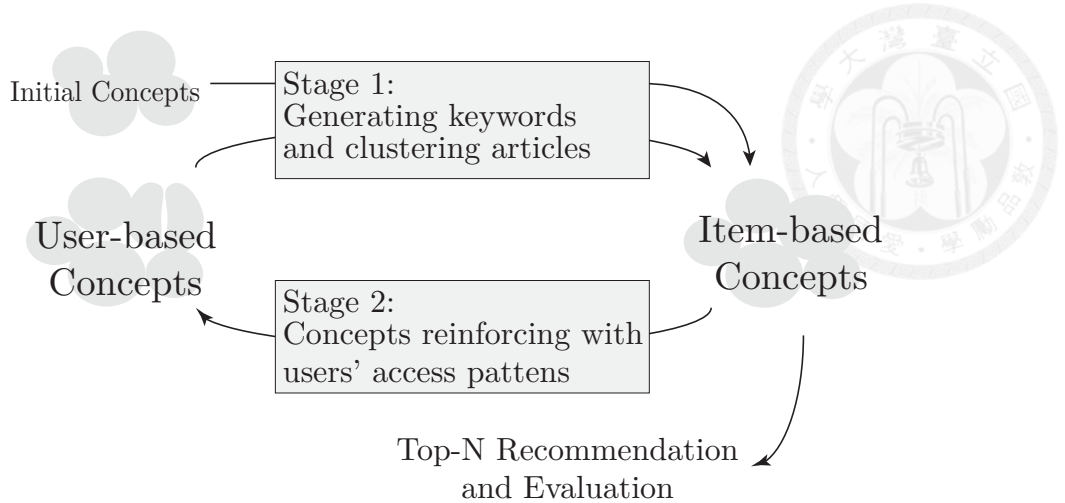


Figure 3.3: The flow of the iterative procedure of the proposed system.

procedure of iteration is shown in Figure 3.3. The two concept spaces and two stages are described briefly as follows:

- Item-based Concepts: clusters of articles with *keywords* as the dimension
- User-based Concepts: clusters of articles after separating process of item-based concepts
- Stage 1: Generating *keywords* to form item-based concepts according to user-based concepts
- Stage 2: Reinforcing item-based concepts with users' reading behaviors

At first, the initial concepts are given from the editors. The editors determine the tags and label articles with these tags. A tag means a concept in our system. This way can prevent some unexpected links between articles. The reason is that a user might be interested in several concepts which are not related to each other. The default tags prevent links between articles which have no relations. This also means that all articles are grouped in only one concepts if default tags do not exist.

Then at Stage 1, extracted from the content of the articles, *keywords* are generated according to the initial concepts. Next, item-based concepts are formed with clustering the articles. Each article is represented as a vector, $\vec{k_w}$, and *keywords* are used to be the dimension of the articles. At Stage 2, with the users' reading behaviors, item-based concepts are reinforced into user-based concepts. Then again *keywords* are regenerated to form the next item-based concepts according to user-based concepts. Until now an iteration is finished. This procedure terminates when the similarity of item-based and user-based concepts exceeds a threshold ϕ . The two

arrows in Figure 3.3 between two groups of concepts represent one iteration. The algorithm of this structure is briefly shown in Algorithm 1. The detail implementation of each function is explained lately.

Algorithm 1: The structure of system

```

1 Set keywords empty
2 for each concept in Conceptsini do
3   Features  $\leftarrow$  GETFEATURES( concept, Conceptsini excluding concepts )
   Find the feature in the set, Features, that increases maximum
   SIMILARITY(Item-based concepts, Conceptsini) then add this feature to
   the set of keywords until the set is empty or no increasing for every
   feature in Features.
4 while SIMILARITY(Item-based concepts, User-based concepts)  $<$   $\alpha$  do
5   Computing weight of articles and the level of interest for each pair of
   users and concepts
6   for each concept in Concepts do
7     cut  $\leftarrow$  GETMAXIMUMCUT(concept)
8     if cut  $\geq$  threshold then
9       group1  $\leftarrow$  one of the groups that should be separated.
10      group2  $\leftarrow$  the other one of the groups that should be separated.
11      Features  $\leftarrow$  GETFEATURES( group1, group2 )
12      repeat
13        Find a feature in the set, Features, that increase maximum
        SIMILARITY(Item-based concepts, Conceptsini)
14        This feature is added to the set, keywords and removed from
        Features.
15      until Features is empty or no increasing for every feature in
        Features;
16   Cluster the articles with keywords as dimension and form item-based
   concepts of next iteration.
17 RECOMMENDATION(Users, Item-based concepts)

```

3.3 Concepts

Two kinds of concept spaces in the structure are named item-based concepts and user-based concepts. Literally, item-based concepts are the actual groups of items.

In the other hand, user-based concepts can be thought of the modified item-based concepts but not the actual groups.



3.3.1 Item-based Concepts

Item-based concepts are groups of articles with the *keywords* as the dimension. The goal of this iterative procedure is to find the best item-based concepts to recommend articles. Item-based concepts are modified each iteration when the dimension is different, in other words, the *keywords* are modified every iteration from different user-based concepts. The length of the vectors, \vec{kw} , for each article would be increased or decreased after an iteration finished.

3.3.2 User-based Concepts

When articles are clustered into concepts, the articles in the same concept are linked according to Equation 3.6. Then each concept is determined to be separated or not. For a concept, if some users read articles only in one group while some other users read articles only in the other group, we find less connection exists in the two groups of articles. Thus the articles in the some concept are considered be separated into two concepts. Figure 3.4 shows this situation. The line between a user and an article represents that the user reads the article. Eight articles are in the concept and seven users read more than one article in the concept. The users u_1, u_2, u_3 and u_4 read one or more articles in the set of a_1, a_2, a_3 and a_4 . On other side, the users u_5, u_6 and u_7 read one or more articles in the set of a_5, a_6, a_7 and a_8 . Though u_7 reads a_2 , we still can separate the concept into two groups because the relations between the two groups are less, only existing between a_2 and a_5 . The dashed line between the two groups of articles in the concept means that this concept should be separated since the relations between the two groups are less. The concepts tend to form with users' reading behaviors is named user-based concepts.

3.4 Stages of Operation

Following the previous framework, two stages show how the two kinds of concept spaces influence each other. One stage is the definition of *keyword* and the approach of generating *keywords* from user-based concepts to item-based concepts. The other is how to reinforce item-based concepts according to the reading history of users. The modified item-based concepts are named user-based concepts. These two stages show the interaction of these two kinds of concept spaces

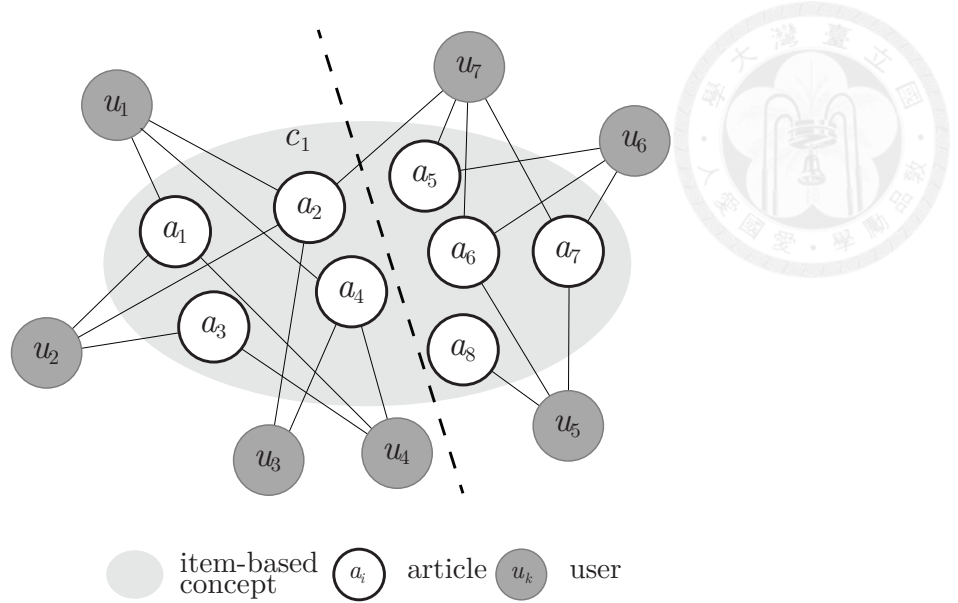


Figure 3.4: A situation that a concept is needed to be separated.

3.4.1 Keyword Generation Stage

Based on the silhouette of user-based concepts, a series of *keywords* are extracted from the contents of the articles. User-based concepts are used to increase or reduce dimension. The increase or decrease of dimension depends on the user-based concepts. When user-based concepts are designed, we look into each concept and analysis the contents of the articles in each concept. When a concept needs to be separated, the chi-square feature selection is implemented to extract features, or we call *keyword* in this paper [16, 20]. The purpose of feature selection is to select a ranked list of features which can separate two classes mostly. Here the features are the segmental contents of the articles, that is, the words tokenized from articles, and the classes are user-based concepts. For a concept needed to be separated, the two classes are the two groups after being split.

However, the features we really select are not ranked only according to chi-square feature selection. Two steps are implemented. In the first step, the top k features are selected from the scores of features, or words. Second, the similarities of user-based and different item-based concepts with different *keyword* as dimension are computed. The *keyword* which results in the minimum distance is chosen. The similarity is computed as follows. Each concept in item-based concepts greedily choose one concept of the maximum repeated articles in user-based concepts, and we compute the number of different articles. The summation of all different articles after each concept choosing is the distance. After the first chosen feature is included into *keyword*, the second feature is selected in the same way. Finally, β *keyword* are

selected to be the dimension of articles when the proportion of similarity compared to the original concepts exceeds α or all of the *keyword* are selected. Then the item-based concepts are formed through clustering articles with *keywords* as dimension.

Furthermore, the *keywords* can also be reduced to reduce the burden of computation. This means that two concepts are merged. However, this way does not refine the concepts and is not implemented in this paper.

Algorithm 2: The features of two groups

Data: Two groups of articles

Result: A set of terms which can separate the two groups most

```

1 Function GETFEATURES(group1, group2)
2   Scores is {score1, score2, ..., scores}
3   Terms ← terms in group1 ∪ terms in group2
4    $\pi$  ← number of articles
5   for each termi in Terms do
6      $\pi_1$  ← the number of articles in group1 containing termi
7      $\pi_2$  ← the number of articles in group2 containing termi
8      $\pi_3$  ← (the number of articles in group1) -  $\pi_1$ 
9      $\pi_4$  ← (the number of articles in group2) -  $\pi_2$ 
10    scorei ←  $\frac{\pi \times (\pi_1 \times \pi_4 - \pi_2 \times \pi_3)^2}{(\pi_1 + \pi_3) \times (\pi_2 + \pi_4) \times (\pi_1 + \pi_2) \times (\pi_3 + \pi_4)}$ 
11  return the best  $\beta$ termi with the highest scorei

```

user	article reading sequence
u_1	$a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_5$
u_2	$a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_1 \rightarrow a_5$
u_3	$a_4 \rightarrow a_2 \rightarrow a_3 \rightarrow a_5 \rightarrow a_1$

Table 3.2: Each row shows the individual user and his reading sequence.

3.4.2 Concept Reinforcement Stage

In the concept reinforcement stage, we add users' reading behaviors to reinforce the item-based concepts generated in the previous stage. For a certain concept, we use users' reading behaviors to decide whether this concept should be separated or not. Under some circumstances the articles in an item-based concept can be trivially divided into two smaller groups, as shown in Figure 3.4. However, in real-world scenario the boundary is not always that obvious. The size of the two separated groups



	a_1	a_2	a_3	a_4	a_5
a_1	-	1/2	0	0	1/2
a_2	0	-	1	0	0
a_3	0	0	-	2/3	1/3
a_4	1/3	1/3	0	-	1/3
a_5	1	0	0	0	-

Table 3.3: The transition probability of Tables 3.2. The reading sequence is from the row to the column.

also matters; if the sizes of the two groups are far from equal, the extracted *keywords* in the following keyword generation stage will be too specific toward a particular article. In other words, the small concepts does not with much information. However, we find the minimum cut algorithm results in the large quantity variance. The small group of split concept dose not contain much information. Therefore, in order to find a boundary that tries to evenly divide the concepts, the weighted maximum-cut algorithm is applied on the complement of the *article graph*. The resulted maximum cut is the candidate boundary between the two possible new concepts. A large cut size in the complement graph means that few connections exist between the two groups in the original *article graph*. When the cut size exceeds a certain threshold, ϕ , we split the concept into two, as shown in Figure 3.5(a). It is worth mentioning that the algorithm is expected to balance the size of the two separated groups when multiple possible boundaries exist. For example, the 5 nodes in Figure 3.5(b) could be separated into two smaller concepts arbitrarily. Although we transfer the linear min-cut problem into an NP-hard one and find the approximation of the maximum cut, we can achieve our purpose through this way.

Initially we consider the transform of Kager’s algorithm. Kager’s algorithm is originally used for minimim cut. The probability of sampling the edges is proportional to the weight. Then it merges the nodes linked with the chosen edge. The process repeats until there merely exist two groups. Thus the edges with higher weight are more likely to be chosen, and the edges with smaller weight are left. In the end, the cut between the two groups are more likely to be the minimum cut after more repeated times. This way has been proven to be the same as sampling the nodes when the probability of a node is equal to the summation of all the edges linked with the node. Here we intent to reversely process and let the edges with higher weight leave in the end. We take the unique number and reversely assign the number to be the probability of sapling nodes. For example, if the weight of nodes are 1, 3, 3, 5, 11, the unique number are 1, 3, 5, 11. We reassign the nodes as

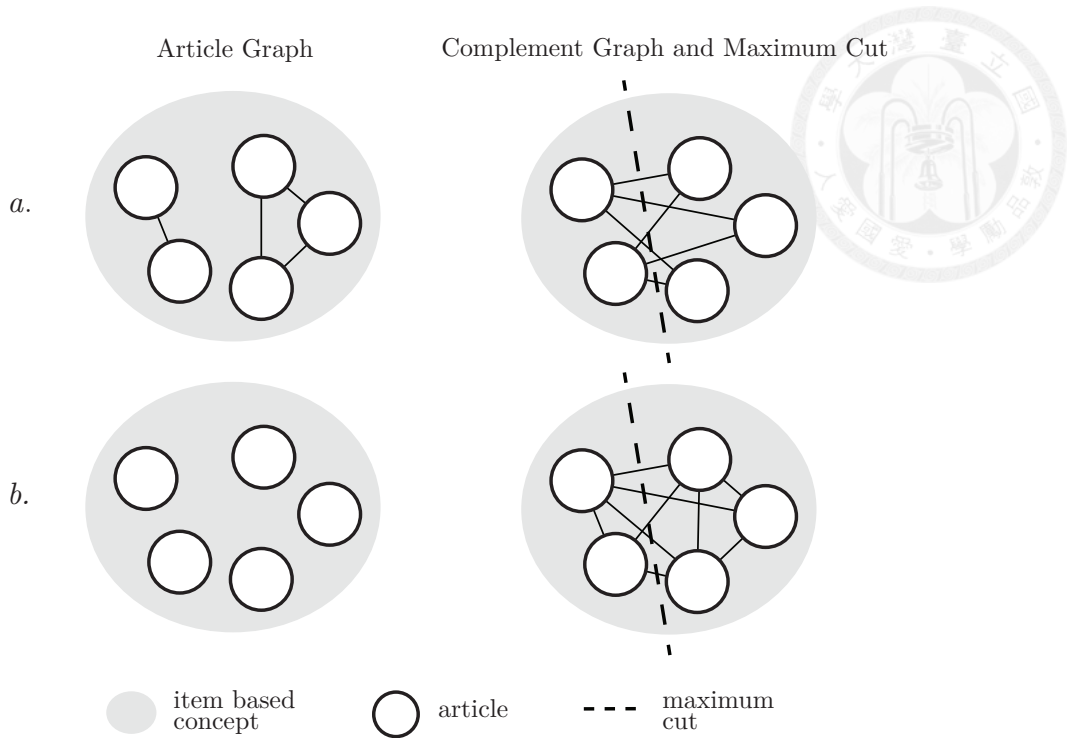


Figure 3.5: Max-cut algorithm applied to the complement of *article graphs* to separate the articles into two smaller concepts.

11, 5, 5, 3, 1, and the probability of these nodes are $\frac{11}{25}, \frac{5}{25}, \frac{5}{25}, \frac{3}{25}, \frac{1}{25}$. However, we compare greedy algorithm and the modified Karger's algorithm. From experiment, with small amount of nodes we find that the greedy algorithm can achieve higher size faster in less repeated times. Thus, we choose greedy algorithm as the approximate method.

Algorithm 3 shows the detail of approximate method. First the nodes, or items, are randomly assigned to one of two groups. Then we examine each nodes and the increase of cut after change find the the Using the max-cut algorithm described above, they can be separated almost evenly because the max-cut algorithm essentially prefers to cut through more edges. At the end of this stage, the original item-based concepts are reinforced and become user-based concepts.

3.5 Recommendation

For article recommendation in the proposed system, the reading dependence is considered. Articles with high probability of being read are probably with higher rank for recommendation. That is, when recommending, we also consider the probability for reading each article only.

3.5.1 Generating Reading Dependence

The reading dependence is also considered when the recommended article are ranked. From the record of the user's reading history, the probability of reading articles is computed. That is, from the reading sequence of the articles, the probability of each article from other articles is computed with the bigram way. We want to find the reading probability of articles themselves. For example, Table 3.2 reveals the users' reading sequence, and Table 3.3 shows the transition probability of the articles in statistics. The sequence from the row to the column means the user's reading sequence in these two articles. For example, $p(a_3, a_4)$ in third row and fourth column means the probability of reading a_4 after reading a_3 is $2/3$.

3.5.2 Ranking of Articles

In the proposed system, the recommendation list is computed according to three factors. From previous result we know that concepts are built and converged after the iterative procedure. Three factors are revealed and listed as follows.

- l_{u_k, c_j} denotes *level of the interest* that u_k shows to c_j
- w_{a_i, c_j} denotes the weight of a_i which consists to c_j
- $p(a_i, a_m)$ denotes the probability that a user reads a_m after a_i .

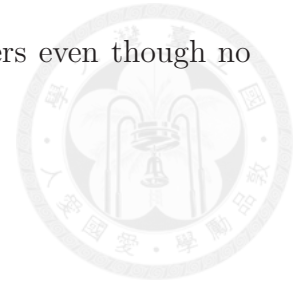
Then the recommendation articles are ranked according to the score computed with the following equation,

$$s_{u_k, a_i} = \frac{\sum_{c_j \in C} [l_{u_k, c_j} \times w_{a_i, c_j} \times \sum_{a_m \in A} p(a_i, a_m)]}{\sqrt{\sum_{c_j \in C} l_{u_k, c_j}^2} \sqrt{\sum_{c_j \in C} [w_{a_i, c_j} \times \sum_{a_m \in A} p(a_i, a_m)]^2}}. \quad (3.7)$$

From the equation we could find that s_{u_k, a_i} increase if u_k is interested more in c_j or a_i is weighted more in c_j . The article a_i with higher score is ranked higher for user u_k . This means that if a user shows more interest in a particular concept, this system recommends the articles with higher product of weight and probability in the concept. If a article shows a higher probability is ranked higher than other articles with same wight but lower probability.

When new articles are added, the *keywords* can be used to cluster these articles into existing concepts, and the reading dependence of the new articles is not considered since no users have read the article. Each new article is included in a concept with the minimum distance. The distance is computed between the new article and the center of the concept when the *keywords* are dimension of the vector

space model [17]. This way can recommends new articles to users even though no users has read them before.





Algorithm 3: The maximum cut of a concept

Data: A concept, group of articles

Result: The maximum cut of the concept

```
1 Function GETMAXIMUMCUT(concept)
2   MaxCut  $\leftarrow$  0
3   for  $i \leftarrow 1$  to  $\theta$  do
4     group1  $\leftarrow$  empty
5     group2  $\leftarrow$  empty
6     /* Randomly assign the articles into two groups          */
7     for article in concept do
8       | number  $\leftarrow$  randomly pick a integer  $\in \{0, 1\}$  if number is 0 then
9       | | group1  $\leftarrow$  article
10      | | else
11      | | | group2  $\leftarrow$  article
12      | Cut  $\leftarrow$  the summation of weight of links between the two groups
13      | /* Change articles to the other group to increase MaxCut
14      | /*
15      | repeat
16      | | MaxIncrease  $\leftarrow$  0
17      | | for article in concept do
18      | | | Increase  $\leftarrow$  increase cut of the article after changed
19      | | | if MaxIncrease < Increase then
20      | | | | MaxIncrease  $\leftarrow$  Increase
21      | | | Cut  $\leftarrow$  Cut + MaxIncrease
22      | | until MaxIncrease is 0;
23      | if MaxCut < Cut then
24      | | MaxCut  $\leftarrow$  Cut
25      | return MaxCut
```



Chapter 4

Experiments and Results

In this section, we compare the articles recommendation from (i) the proposed system, (ii) the initial concepts labeled by editors, (iii) the CF results and (iv) the content-based filtering.

4.1 Data Set and Configuration

We use the data and the browsing logs from womany.net¹, a website that aggregates articles from various sources. The data includes 3,205 articles and the reading behaviors from 31,692 users in total. Each user reads 45 articles in average. Every article is labeled with several tags by the editors of the website. An article can have multiple tags. The tags are used as the initial concepts.

According to the characteristic of data, we only get the browsing logs of users and do not know the rating of each user on items. Therefore, when consider the users' behaviors, the user access pattern is used as the implicit opinions. Also, when creating recommendation, top-N recommendation is used instead of the prediction of items.

When implementing the system, several parameters and tools are shown in this subsection. When generating the *keywords*, two parameters are set. β is set to 8 since we find no obviously increase in similarity when more *keywords* are selected as dimension. Only one or two *keywords* are needed in most concepts to make the similarity exceed α , which is empirically set to 0.9.

Now we need to decide the threshold of the size of maximum cut in Stage 2. If the size of the maximum cut exceeds the threshold, the articles separated by the cut should have few relations. Thus, the concept should be split into two smaller concepts using the calculated cut.

¹www.womany.net

When determining the threshold, we first assume the users do not have preferences for any specific concepts. We use 31,692 computer agents to simulate user reading behaviors. Each agent randomly reads 45 articles and then the articles are grouped to 100 concepts. The number of the concepts are empirically chosen. As a result, the average size of maximum cut over the 100 concepts is 468.916. Therefore, we choose 468.916 as the threshold.

During Stage 1, Stanford Word Segmenter [26] is used to segment the article contents before extracting the keywords. After that, K-means [9] algorithm is used to cluster the articles into item-based concepts. The k in K-means is changeable. We use elbow method to find the first proper k . Table 4.1 summarizes the parameters.

In addition, when estimating a concept should be separated or not, we use a method to approximate maximum cut. Though we transfer a linear problem into a NP-hard problem, with the approximated solution we could separate the concept as what we want. The implementation is described as follows. First, each article is rand assigned into one of two groups. Then we examine each article to find the one which is changed to another group can increase the maximum cut between these two groups and change the article. The process then continues to find the next article until no article changed can increase the cut size. Then on time of process is finished. We repeat the process for 10 times to find the ever existing maximum value and use the value to be the found one.

Parameter	value	Description
β	8	the number of <i>keywords</i> selection in each iteration
α	0.9	the similarity between item-based concepts and user-based concepts
γ	30	the number of neighbors in user-based CF
ϕ	468.916	threshold of maximum cut
θ	10	repeated times of approximate max-cut algorithm

Table 4.1: Summary of parameters.

4.2 Evaluation Metrics

Several common methods of evaluation for RS are used [21]. The most common is mean absolute error (MAE). MAE comes from the difference value of the prediction and the true rating. However, the characteristic of the data results in the prediction output can only be top-N recommendation. Different from exact rating of each unseen item within a numerical scale same as the rating given by users, top-N

recommendation only provides a ranking list of items. As a result, MAE is not appropriate for this kind of data. Root-mean-square deviation, a modification from MSE, is not appropriate for the same reason.

Here we use precision-recall (PR) curves [6] to evaluate the article recommendation. Drawing PR curves is another method often used to test the accuracy of recommendation systems. PR curves are a kind of curves with recall as x-axis and precision as y-axis. The definitions of precision and recall are shown as follows. Precision is the fraction of retrieved items that are relevant to the find and defined as

$$precision = \frac{|\text{relevant items} \cap \text{retrieved items}|}{|\text{retrieved items}|}. \quad (4.1)$$

The numerator is the number of items that are both relevant and retrieved, and the denominator is the number of total retrieved items at that moment. Recall is the proportion of the number of articles a user really reads from the prediction and is defined as

$$recall = \frac{|\text{relevant items} \cap \text{retrieved items}|}{|\text{relevant items}|}. \quad (4.2)$$

A spot on PR curves represents a hit, which means the current prediction is correct when examining the rank list one by one. Figure shows examples of PR curves. We assume the ground truth is a_1, a_7, a_3 and a_1 . Thus any one of these four is regarded as a hit. In this evaluation, if a user reads a recommended article in the test data, the recommendation is regarded as successful and a hit reveals. Precision and recall then are computed as a hit happens. Tables 4.2, 4.3 and 4.4 show examples for three prediction from different systems. Figure 4.1 shows the three different curves from these prediction. In the figure, we find that Prediction B is the best and Prediction C is the worst of these three. In our experiment, since there are totally 31,692 users, we average the PR curves over all users and set the x-interval to be 0.05.

Prediction A	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
Precision	1		$\frac{2}{3}$							$\frac{3}{10}$
Recall	$\frac{1}{4}$		$\frac{2}{4}$							$\frac{3}{4}$

Table 4.2: The example of PR curves.

Another evaluation method is mean average precision (MAP). This is the mean of all precision when a hit reveals. For example, we can look at Tables 4.2. Average precision (AP) is $\frac{1 + \frac{1}{3} + \frac{3}{10}}{3} = 0.544$. Then AP of Tables 4.3 and Tables 4.4 can be computed as the same way. They are 1 and 0.145 respectively. MAP is $(0.544 + 1 + 0.145)/3 = 0.563$ in the example.

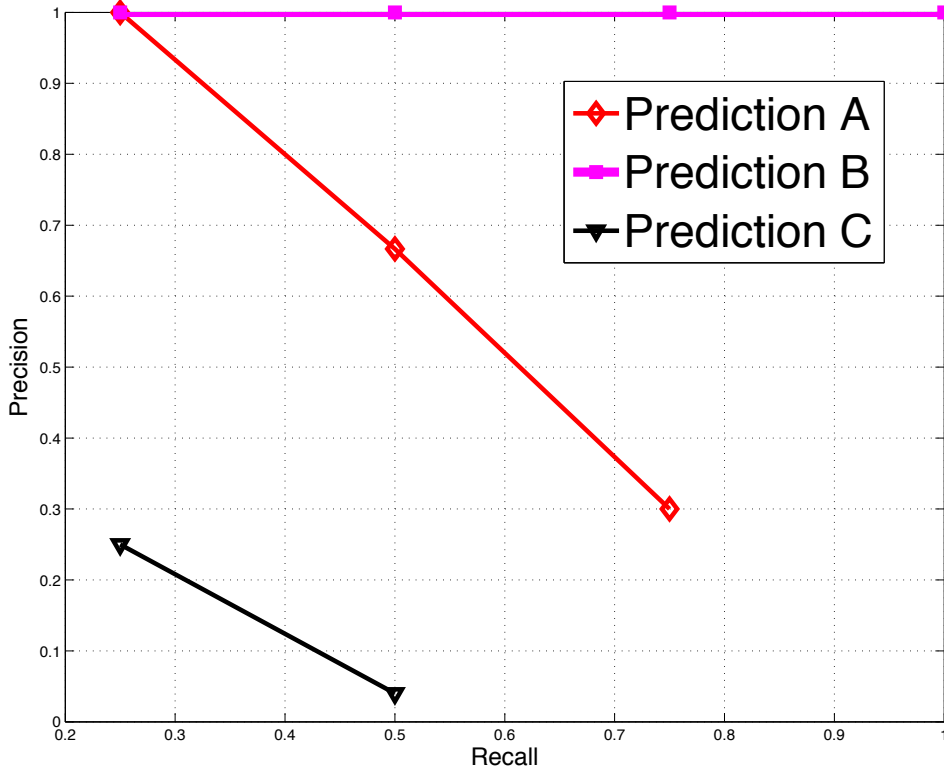


Figure 4.1: PR curves of three different prediction.

Prediction B	a_3	a_11	a_7	a_1
Precision	1	1	1	1
Recall	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{3}{4}$	$\frac{4}{4}$

Table 4.3: The example of PR curves.

Prediction C	a_12	a_7	a_4	a_11	a_5	a_6	a_13	...	a_1	a_{14}
Precision				$\frac{1}{4}$					$\frac{2}{50}$	
Recall				$\frac{1}{4}$					$\frac{2}{4}$	

Table 4.4: The example of PR curves.

4.3 Evaluation Process

Here we talk about the implementation of each RS. The testing data is introduced followed by (1) the proposed system (2) item-based/user-based CF, (3) content-based filtering. The output of these RS are a ranking list with 100 as the length. Then according to the hit, we get PR curves and average MAP.

4.3.1 Testing Data

The reading behavior records are separated into two parts: one for training, and the other for testing. The training part is the reading history from Dec 10, 2012 to Oct 6, 2013, containing about 1.09 million reading records. The testing part is the data from Oct 7, 2012 to March 7, 2014, containing 439 thousand reading record as the ground truth. Reading records of 556 new articles published after Oct 6, 2013 are also included in the testing part. However, we exclude the reading records from the new readers registered after Oct 6, 2013 because the system only recommends articles for users who have reading history in the training part. The sequence of articles in testing data does not matters. If the ranking list of RS refers to any article the user read in testing data, it is regarded successful equally.

4.3.2 Evaluation of Proposed System

Before testing, the 556 new articles is not belong to any concepts. The set of *keywords* generated in the training phase are used as dimension to cluster the new articles into the concepts. The value of each dimension is decided with tf-idf. The distance is compared using the vector space model. For k-means, the new articles are clustered into the nearest clusters according to the distance of the centers. Afterwards, the ranking of all articles can be calculated for each user using Equation 3.7. The weight and reading dependence of article are set average of existing articles to prevent influence.

4.3.3 Evaluation of CF

As for CF, both item-based CF and user-based CF are applied. In user-based CF, cosine-based similarity [22] is applied to search the nearest neighbors. Similarity value between two users are computed according to the articles they both read. The more articles the users read in common, the more similar they are. γ nearest neighbors are selected, and then we compute the articles mostly read from the neighbors. In experiments γ is set to 30, which is about 1/100 users.

In item-based CF, the similarity between items are also computed with cosine-based similarity. Since the data only provides the reading history between users and articles, we do not know the rating of the articles from the users. Here we assume that if a user reads an article, the user is interested in the article. This means no negative feedback is considered. Two articles are regarded more similar when more users read both of them. In the experiments, the same 100 articles of proposed

system are chosen. In addition, the newly added articles cannot be recommended since the CF system does not have any reading history of the new articles.

4.3.4 Evaluation of Content-based Filtering

The user profile is constructed according to the articles read. In this experiment we use relevance feedback mentioned in [28] to build the user profile. When a user has spent a certain amount of time reading a document a_i the user profile P is updated with the following equation: $P' = \lambda P + \mu \vec{a}_i$. P is the original profile, P' is the new profile, \vec{a}_i is the vector of document a_i and Weight μ determines the relative importance of a document to the user. In our experiment, since we do not know the rating of users, we set the article read means that the user is interested in it. This also implies that μ is always larger than zero and only positive feedback is considered. The profile vector is adjusted to a diminishing of the user's interests by λ , a weight between 0 and 1 that reduces the term weights in the profile. This weight is determined via experimentation. The different values of λ with fixed μ or different μ with fixed λ affects less. As a result, we set λ to be 0.8 and μ as 0.2.

4.4 Results and Discussion

All RS are set to provide 100 articles of prediction for each user, and the articles of test data are the original 3,205 articles and 556 new articles. Then we check the recommended articles one by one and compute the precision and recall. The average PR curves over all users with interval as 0.05 from proposed systems of different iterations are shown in Figure 4.2. It starts with the initial concepts and converge after twelve iterations. We can find the curve generally higher with the more iterations. Figure 4.3 is part of the Figure 4.2, and we can find the highest line is the last iteration. Figure 4.4 compares average PR curves from different systems: (1) user-based/(2) item-based CF, (3) the content-based filtering, (4) initial concepts from editors and (5) item-based concepts with different iterations. From the figure we can find the proposed system exceeds other RS at about after the 3 iterations.

The MAP of all users with each iteration is shown in Figure 4.6. The MAP generally increases with iteration increasing. The comparison of MAP for item-based/user-based CF and content-based filtering is shown in Figure 4.7. After 4 iterations, the MAP of proposed system exceeds all other RS.

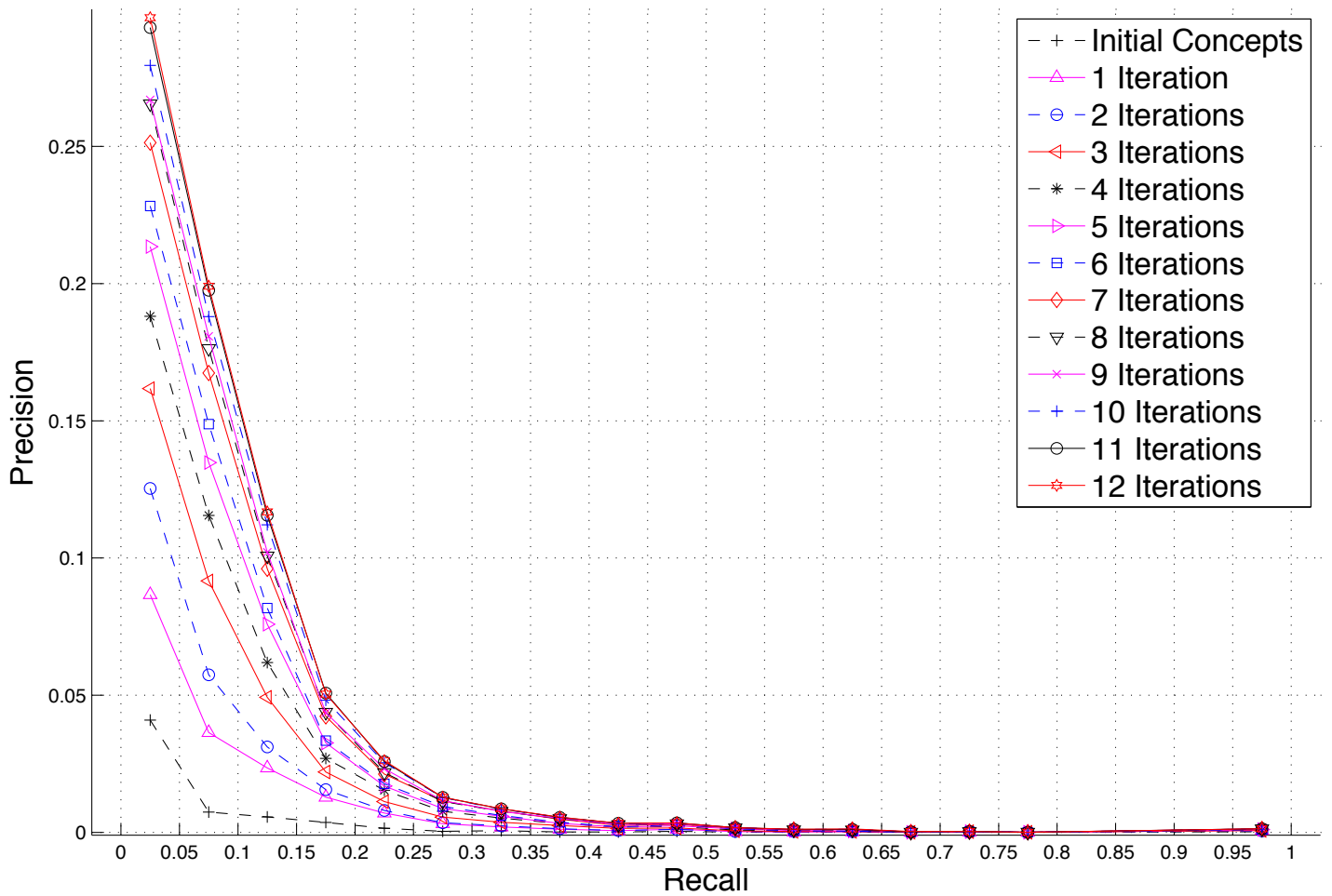


Figure 4.2: The average PR curves of proposed system with the interval 0.05.

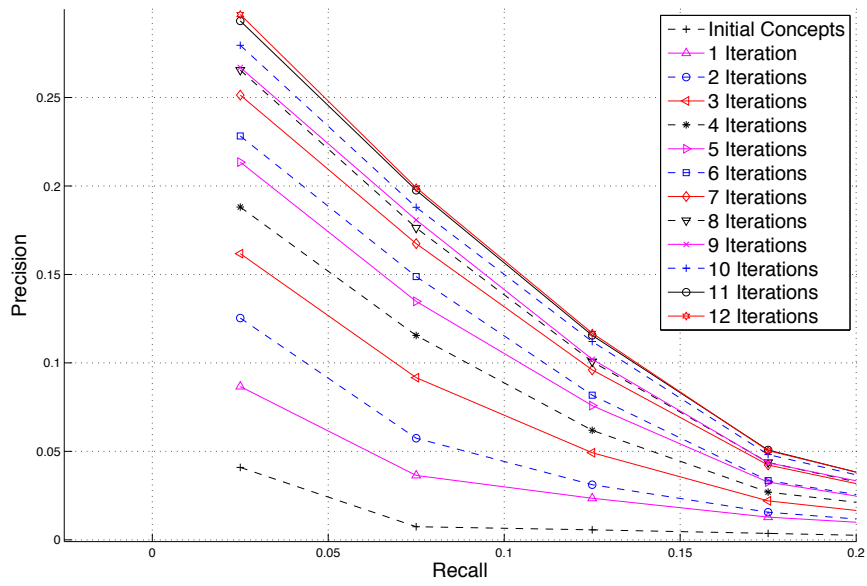


Figure 4.3: Zoom section of the Figure 4.2.

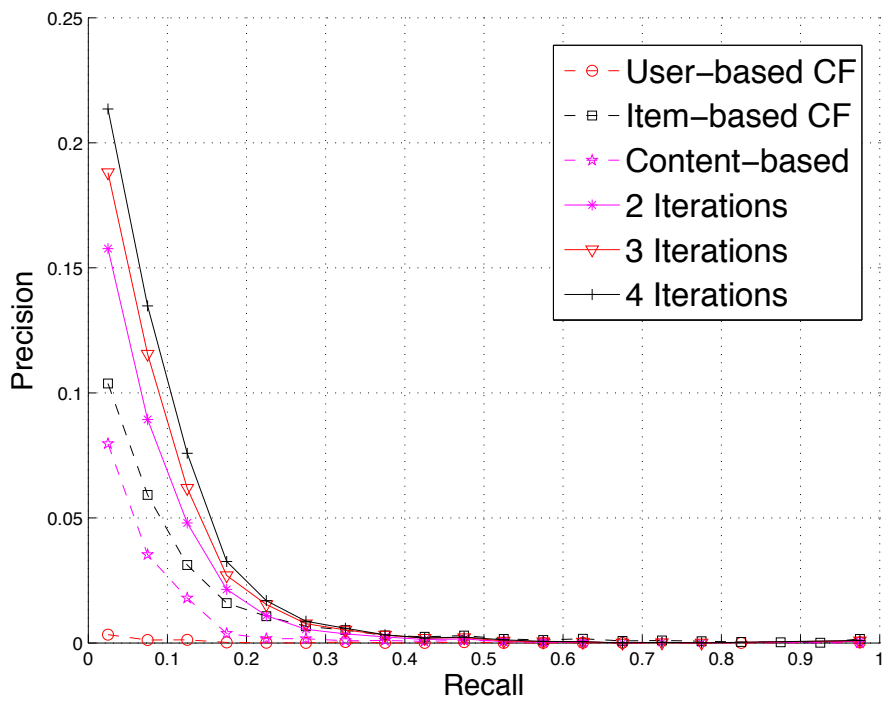


Figure 4.4: The comparison of average PR curves.

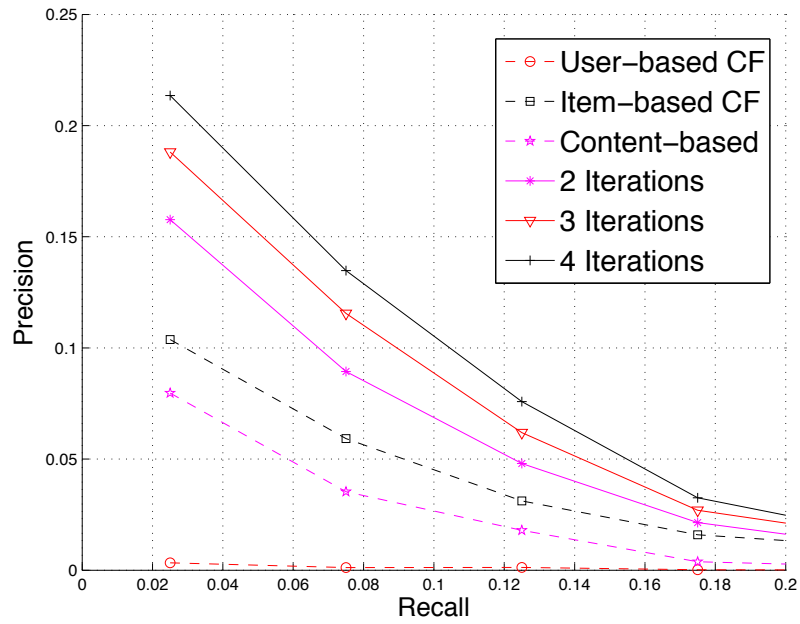


Figure 4.5: Zoom section of Figure 4.4.

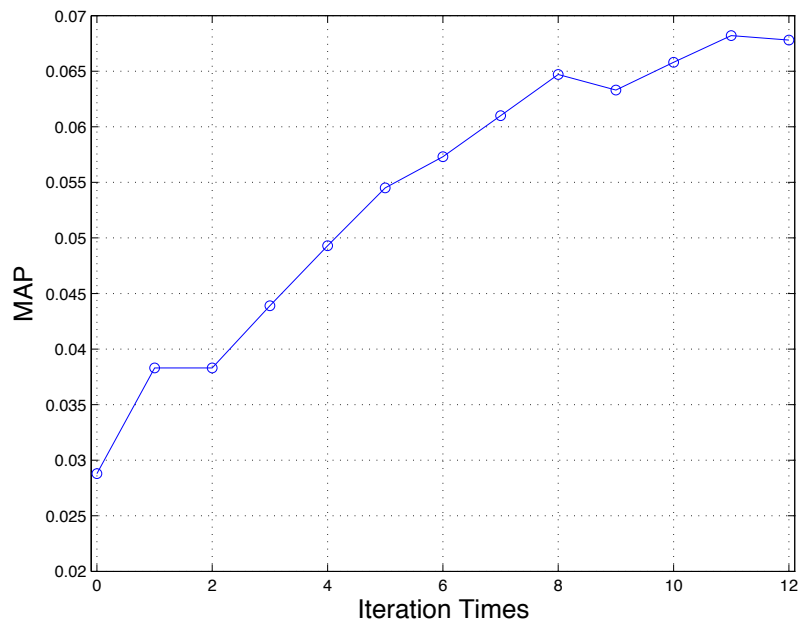


Figure 4.6: MAP of proposed systems with different iterations.

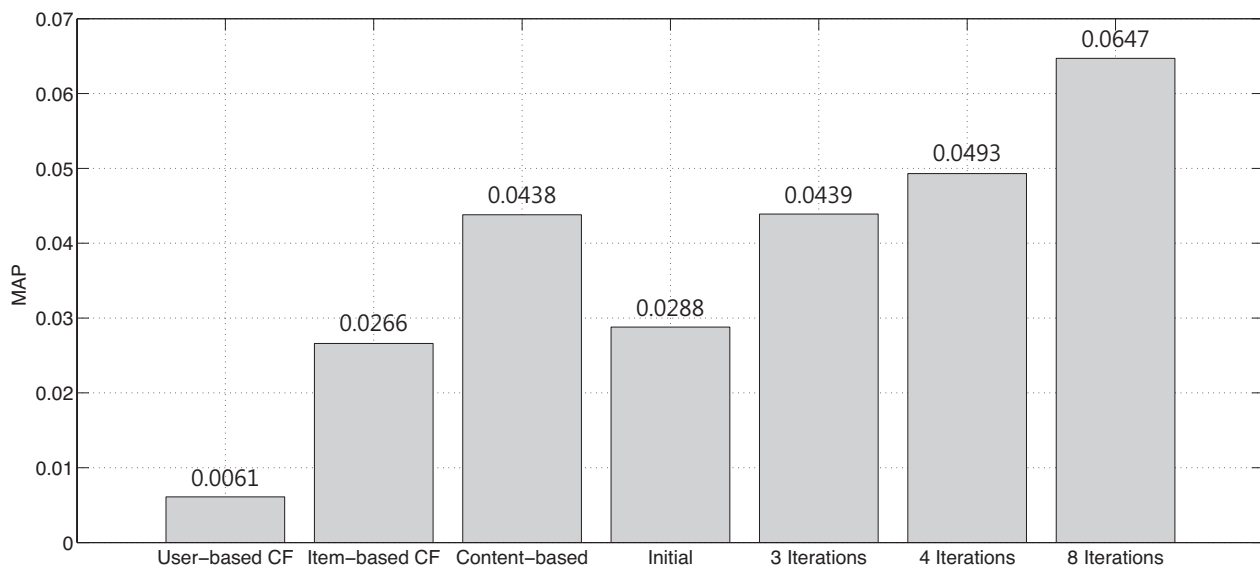


Figure 4.7: The comparison of MAP.



Chapter 5

Conclusion

This paper proposes a recommendation system (RS) with two concept spaces, item-based and user-based concepts, in an iterative procedure. The actual groups of articles are named item-based concepts and the expected groups of articles are named user-based concepts. The user-based concepts are reinforced from item-based concepts with readers' reading behaviors. When two articles are both read by one or more users, the two articles are related. The more users read both of them, the stronger the relations. A series of *keywords* are generated from the contents of articles and used as the dimension to form item-based concepts in the next iteration. Reading dependence is also considered when generating recommendation. We place both readers and articles in concept space and compute the association. In our case study, the average precision-recall curves indicate that the proposed RS produces more hits with more iterations. In the perspective of average MAP, we find that it generally increases. To some extent, it is higher than user-based/item-based CF and content-based filtering.

Our proposed RS deal with scalability and sparsity with clustering methods. Each cluster is a group of articles and named concept. Our proposed RS can dynamically scale the scale of clusters up and down in the iterative procedure. The size increases as necessary and merely increase to 87 from 5 while the dimension of user-based CF is 3,200 and 30,000 for item-based CF. In other hand, the proposed RS cluster articles with *keywords* as dimension. Thus the new articles can be clustered and recommended. The size of *keywords* increases as necessary in each iteration to separated concepts. According to the results of PR curves and average MAP, the iterative procedure can further increase the accuracy of the recommendation.

The proposed RS can be applied to other field in such a manner. Just as we extract *keywords* from textual articles, we can extract video and audio features to apply the RS in multimedia recommendations. As long as the items to be recom-

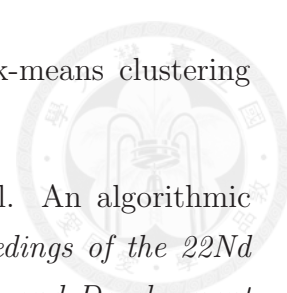
mended contain extractable features, we believe that the idea of reinforcing item-based concepts to user-based concepts can be extended to any situation with regard to interactions between users and items.

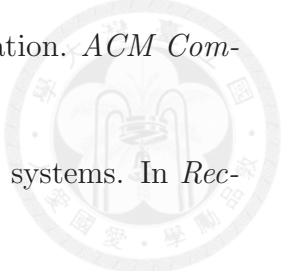




Bibliography

- [1] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720. AAAI Press, 1998.
- [2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI'98*, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.
- [3] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer, 1999.
- [4] M. Connor and J. Herlocker. Clustering items for collaborative filtering, 2001.
- [5] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10*, pages 293–296, New York, NY, USA, 2010. ACM.
- [6] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.
- [7] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.
- [8] J. A. Hartigan. Clustering. *Annual review of biophysics and bioengineering*, 2(1):81–102, 1973.

- 
- [9] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.
- [10] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99*, pages 230–237, New York, NY, USA, 1999. ACM.
- [11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.
- [12] F. Jelinek. *Statistical methods for speech recognition*. 1997.
- [13] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
- [14] G. Linden, J. Jacobi, and E. Benson. Collaborative recommendations using item-to-item similarity mappings, July 24 2001. US Patent 6,266,649.
- [15] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, Jan 2003.
- [16] H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. *SIGIR Forum*, 31(SI):67–73, July 1997.
- [17] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, Nov. 1975.
- [18] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2Nd ACM Conference on Electronic Commerce, EC '00*, pages 158–167, New York, NY, USA, 2000. ACM.
- [19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web, WWW '01*, pages 285–295, New York, NY, USA, 2001. ACM.

- 
- [20] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, Mar. 2002.
- [21] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.
- [22] A. Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–43, 2001.
- [23] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.
- [24] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, Jan. 2009.
- [25] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. Phoaks: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.
- [26] H. Tseng. A conditional random field word segmenter. In *In Fourth SIGHAN Workshop on Chinese Language Processing*, 2005.
- [27] L. Ungar, D. Foster, E. Andre, S. Wars, F. S. Wars, D. S. Wars, and J. H. Whispers. Clustering methods for collaborative filtering. AAAI Press, 1998.
- [28] R. Van Meteren and M. Van Someren. Using content-based filtering for recommendation. In *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*, 2000.
- [29] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456, New York, NY, USA, 2011. ACM.