DOCTORAL DISSERTATION

THREE ESSAYS ON ENTERPRISE INFORMATION SYSTEM MINING FOR BUSINESS INTELLIGENCE

A dissertation submitted to the

HEINZ COLLEGE, CARNEGIE MELLON UNIVERSITY

in partial fulfillment for the requirements for the degree of

DOCTORAL OF PHILOSOPHY

in

INFORMATION SYSTEMS

by

Nachiketa Sahoo

ADVISERS
Dr Ramayya Krishnan (Chair)
Dr Jamie Callan
Dr Christos Faloutsos

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

September 24, 2009

Abstract

This dissertation consists of three essays on data mining in the context of enterprise information system.

The first essay develops a clustering algorithm to discover topic hierarchies in text document streams. The key property of this method is that it processes each text documents only once and assigns it to the appropriate place in the topic hierarchy as they arrive. It is done by making a distributional assumption of the word occurrences and by storing the sufficient statistics at each topic node. The algorithm is evaluated using two standard datasets: Reuters newswire data (RCV1) and MEDLINE journal abstracts data (OHSUMED). The results show that by using Katz's distribution to model word occurrences we can improve the cluster quality in majority of the cases over using the Normal distribution assumption that is often used.

The second essay develops a collaborative filter for recommender systems using ratings by users on multiple aspects of an item. The key challenge in developing this method was the correlated nature of the component ratings due to Halo effect. This challenge is overcome by identifying the dependency structure between the component ratings using dependency tree search algorithm and modeling for it in a mixture model. The algorithm is evaluated using a multicomponent rating dataset collected from Yahoo! Movies. The results show that we can improve the retrieval performance of the collaborative filter by using multi-component ratings. We also find that when our goal is to accurately predict the rating of an unseen user-item pair, using multiple components lead to better performance when the training data is sparse, but, when there is a more than a certain amount of training data using only one component rating leads to more accurate rating prediction.

The third essay develops a framework for analyzing conversation taking place at online social networks. Conversation data is inherently multi-modal. It consists of text, author, recipient, keywords etc. Therefore matrix based representation is inadequate for analyzing such data. This chapter proposes a tensor based framework for representing such multi-modal data. With the help of blog data collected from a large IT services firm it shows that by tensor factorization one can identify significant topics of conversation, the trend of the topic over time, and the important actors in each topic. In addition it also shows that hubs identified by such tensor factorization are closer to the topic of discussion in their response than hubs identified by other content free approaches such as performing HITS on just the reply network.

Table of Contents

Abstract 2

Table of Contents 3

List of Tables 5

List of Figures 7

1 Introduction 9

2 Discovering topic hierarchy in document streams 12
- 2.1 Introduction . . . 12
  - 2.1.1 Contribution of this research . . . 13
- 2.2 Literature review . . . 14
- 2.3 Text Documents and word distributions . . . 23
  - 2.3.1 Models based on Poisson distribution . . . 24
  - 2.3.2 Katz's K-mixture model . . . 25
  - 2.3.3 Fitness comparison . . . 28
- 2.4 Algorithms for text . . . 29
  - 2.4.1 COBWEB: when attribute values follow Katz's distribution . . . 29
  - 2.4.2 COBWEB: when attribute values follow Negative Binomial distribution . . . 31
- 2.5 Cluster Evaluation Methods . . . 32
  - 2.5.1 Evaluating the clusters . . . 32
  - 2.5.2 Evaluating the hierarchy . . . 33
- 2.6 Experiment setup and results . . . 35
  - 2.6.1 Reuters-RCV1 . . . 35
  - 2.6.2 OHSUMED (88-91) . . . 39
- 2.7 Conclusion . . . 43

3 Multi-component Rating Collaborative Filtering 45
- 3.1 Introduction . . . 46
- 3.2 Multi-component rating recommender system . . . 51
  - 3.2.1 Data description and preliminary analysis . . . 51
  - 3.2.2 Modeling component ratings for collaborative filtering . . . 53
  - 3.2.3 Parallels . . . 55
  - 3.2.4 Model estimation using EM algorithm . . . 56
  - 3.2.5 Predicting the Overall rating . . . 61
  - 3.2.6 Instance based approaches . . . 62
- 3.3 Results and discussion . . . 63
  - 3.3.1 Experiments with Random Training Sample . . . 63
  - 3.3.2 Experiments with time ordered data . . . 70
  - 3.3.3 Filling-in missing component ratings . . . 72

3.4 Conclusions . . . 78

4 Socio-temporal analysis of conversations in intra-organizational blogs 81
4.1 Introduction . . . 81
4.2 Importance of entities in intra-organizational blogs . . . 84
4.2.1 Summary of notations . . . 86
4.2.2 Importance definition for multi-modal data . . . 86
4.2.3 Blog post developments . . . 87
4.2.4 Blog conversation development . . . 89
4.2.5 Comparison with the existing methods . . . 89
4.3 Dataset . . . 89
4.4 Application of tensor factorization . . . 91
4.4.1 Data preparation . . . 91
4.4.2 Illustrative results . . . 93
4.4.3 Comparison with content independent hub and authority . . . 95
4.4.4 "On topic" quality of the top hubs' response . . . 101
4.4.5 Community discovery . . . 102
4.5 Conclusion . . . 105

5 Conclusion 107

A Document Clustering 110
A.1 MLE of Katz's distribution parameters . . . 110

B Multi-component Rating Collaborative Filtering 112
B.1 Derivation of marginal distributions . . . 112
B.1.1 $P(U, I, O)$ for the model with dependency among the ratings . . . 112
B.1.2 $P(U, I, O)$ for the model with independent component ratings . . . 113
B.1.3 $P(U, I, O)$ for the model with only the overall ratings . . . 113
B.1.4 $P(U, I, S, O)$ from the complete joint distribution . . . 114
B.2 Halo in multi-criteria movie rating . . . 114
B.2.1 Halo Effect . . . 114
B.2.2 Halo in movie rating data . . . 118

References 122

List of Tables

2.1 Likelihood comparison . . . 30
2.2 RCV1 dataset . . . 36
2.3 Cluster quality comparison on RCV1 data . . . 37
2.4 Evaluation of the cluster hierarchy using RCV1 data . . . 39
2.5 OHSUMED dataset (88-91) . . . 39
2.6 Cluster quality comparison on OHSUMED dataset . . . 41
2.7 Evaluation of the cluster hierarchy using OHSUMED data . . . 42

3.1 Examples of Collaborative Filtering based recommender systems . . . 46
3.2 An example of multi-component rating. Ratings are on a scale of 0 to 4. . . . 51
3.3 Correlation among components of rating . . . 53
3.4 Principal components . . . 53
3.5 Factor loadings after quartimax rotation . . . 53
3.6 Partial correlations controlling for Overall rating . . . 56
3.7 Set of metrics compared . . . 64
3.8 Increase in dataset size after filling in missing components . . . 74
3.9 Comparison of different methods filling in missing rating components. . . . 75

4.1 Blog data description . . . 89
4.2 Posts in different communities . . . 90
4.3 Different communities . . . 90
4.4 Blog reading counts obtained from the web access logs. The raw data often contains one person accessing a post multiple times. For some purposes it might be more meaningful to consider only the unique instances of a person reading a blog post. . . . 90
4.5 Average number of people who read posts of the most frequent bloggers. Average over the entire blog dataset is 22 . . . 91
4.6 Topics of conversation . . . 92
4.7 Descriptive statistics of the reply network . . . 93
4.8 Comparison of hubs. . . . 98
4.9 Comparison of authorities. . . . 99
4.10 Reader writer adjacency matrix . . . 99
4.11 Correlation of authorities identified by HITS and PARAFAC with centralities of the writers (A_R) on the reading network. . . . 100
4.12 Precision recall of community discovery . . . 105

5.1 Thesis summary . . . 108

B.1 Correlation among rating components . . . 118
B.2 Partial correlation given Overall. $R_i, R_j \in \{S, A, V, D\}$ . . . 118
B.3 Eigen values of the correlation matrix . . . 119
B.4 Eigen values of the partial correlation matrix . . . 120

B.5 Factor loadings after quartimax rotation 120
B.6 Principal component analysis of rating components 121

List of Figures

2.1 COBWEB control structure. . . . 19
2.2 COBWEB illustrated . . . 20
2.3 COBWEB: After first two items are added. . . . 20
2.4 COBWEB: Two partitions of the root cluster. . . . 21
2.5 Merge and split operations illustrated. . . . 22
2.6 Word distribution . . . 26
2.7 Hierarchical evaluation illustrated . . . 34
2.8 Three classification hierarchies. . . . 36
2.9 Cluster quality comparison on RCV1 data . . . 38
2.10 MeSH labels file . . . 40
2.11 Sibling precision: using 25 and 75 words . . . 42
2.12 Sibling precision: using 125 and 175 . . . 43

3.1 Aspect model in Hofmann and Puzicha, 1999. Latent variable nodes are shaded and observed variable nodes are not shaded. . . . 49
3.2 Flexible Mixture model of Luo Si, Rong Jin, 2003. Latent variable nodes are shaded and observed variable nodes are not shaded. . . . 49
3.3 *log-log* plot of frequency of users who have rated a certain number of movies. *log*s are calculated with base e. . . . 52
3.4 Extreme ratings are more frequent . . . 52
3.5 Flexible Mixture model for component rating collaborative filtering . . . 54
3.6 Discovered structure in the sub ratings . . . 55
3.7 Flexible Mixture Model with component rating dependency structure . . . 57
3.8 Flexible mixture model with independent component ratings . . . 59
3.9 Plot of errors by amount of training data used, for different models. . . . 66
3.10 Precision-Recall curve for Three methods . . . 68
3.11 Mean precision at top-5 and Mean reciprocal rank . . . 69
3.12 MAE comparison using randomly selected training data, vs., using training data as they are received. . . . 71
3.13 Three algorithms compared by using data in time order for training . . . 72
3.14 Precision-Recall curves using prior data for training and later data for testing . 73
3.15 Records with partial information . . . 74
3.16 Error distributions . . . 76
3.17 Accuracy of collaborative filtering after filling in missing components . . . 76
3.18 Precision recall curves for filled in data . . . 77

4.1 Tensor representation of the conversation . . . 85

4.2 Trends of topics and importance of top bloggers in each topic. Four identified topics are illustrated. The top-left one is about software testing. The top-right one is about Indian politics. The bottom-left one is about physical exercise. The bottom-right one is about new technology companies. The histograms of the actor importances show different degree of participation of actors in the topic. For example, the topic of physical exercise can be seen to be driven primarily by one person. 94
4.3 Trends of topics and importance of topic specific hubs and authorities in each. The plots from the top-left to bottom-right can be roughly described to be on topics "Free Linux open source community", "Cast system in India", "Mythology", and "Hinduism". The histogram of source bloggers show the extent to which the top bloggers have posted in the topic. The histogram of target bloggers show the extent to which they are the recipient of messages in the topic—presumably because of their prior participation in the topic. 96
4.4 Analysis of messages exchanged in the Sports community reveals Cricket and Soccer topics 97
4.5 Discussion of file systems and Linux medial players in FLOSS community . . . 97
4.6 Average KL divergences of the top hubs identified by HITS, keyword specific HITS, and Tensor factorization 103

CHAPTER 1

Introduction

Mining large scale corporate data to gather business intelligence has become a part of doing competitive business today. This involves identifying patterns in data, collected from a variety of sources, which can lead to economic gains. Developing methods to do so provides strategic advantages. Under these new realities of business operation this dissertation makes a contribution by developing data mining methods in three areas of enterprise information systems.

Text documents, such as reports, news articles, emails etc., are one of the most common forms of information in any organization. They contain data in unstructured form. The data variables and their values are not as well defined as data gathered from some other sources such as, data collected from surveys or those recorded from experiments. Therefore, text data presents unique challenges to the analysts. In last two decades text data mining and information retrieval literature has seen rapid progress. Some of the topics in the literature include document clustering, classification, document indexing and retrieval, natural language processing, etc. However, there are opportunities for further development in each area especially with relation to their application in corporate data mining. The first essay focuses on one such important application. One of the common nature of corporate document databases is that documents arrive in a streaming manner rather than in batches, e.g., from newswires, emails, blogs and electronic message boards etc. Given a large collections of documents one of the task of interest is to automatically identify topics and subtopics that exist in it (See Section 2.2 for a review). Although, document clustering literature addresses this task, most of the proposed methods work in a batch manner, i.e., they wait for certain set of documents to be collected and then process them all together. I argue in Chapter 2 that batch clustering can be unsatisfactory in an enterprise setting where the time sensitive nature of decision making requires real time processing of the documents. I propose an algorithm to identify topic hierarchies in document streams by processing the documents as they arrive. This algorithm exploits the skewness in occurrence distribution of a word across documents. I evaluate the proposed algorithm using two large document datasets collected from Reuters news wire and from MEDLINE medical journals database.

The second essay focuses on mining customer ratings on items to identify promising customer-item match. This is a task of collaborative filtering based recommender systems. Such recommender systems are used by many online businesses, such as

Amazon and Netflix, to help customers discover interesting items and to target advertise items in inventory by finding potential customers. There is a large literature in collaborative filtering (See Section 3.1 for a review). However, they have been mostly designed to work with unidimensional ratings, e.g. ratings indicating how much a person likes an item on a scale of 1–5. However, often times there are multiple aspects of customer experience that are not captured by one dimensional ratings, e.g. restaurants can be rated for their food, ambience, service; movies can be rated for the story, acting, directorial style etc. Collaborative filtering for multi-component rating data has not been actively researched due to lack of large scale datasets. However, recently Yahoo! movies have started collecting multi-component ratings from its users on movies they have watched. The users rate each movie indicating how much they liked its story, acting, visuals, direction, and how much they liked it overall. More detailed rating data at the matching of user and item should enable better recommendations. Based on this idea I develop one of the first multi-component rating collaborative filtering algorithms. One of the major challenges in this development was that component ratings are correlated because of Halo effect. I discovered and modeled for the dependency structure among the component ratings in a probabilistic graphical models framework. Then I evaluate the proposed algorithm using multi-component rating data from Yahoo! movies and compare it with existing methods.

The third essay focuses on the emergent phenomenon of online social conversation occurring at intra-organizational blogs. Such blogs provide the employees media for expressing themselves and channels for connecting to other bloggers of similar interest. Thus blogs act as a online social network among the employees that is auxiliary to the traditional social network. However, unlike in the traditional social network conversations taking place at the online blog social network are easier to monitor and analyze. Such analysis of blog conversation can enable us to identify experts in different topics and monitor developments of online conversations. There has been a lot of work in Information Retrieval literature in topic detection and tracking. In social network analysis methods have been developed for identifying important and influential actors in a network. The contribution of this essay is an integrated analysis of the multi-modal blog data that consists of blog authors, post text and timestamps. For this study we collected data from the internal blogs of a large IT services firm. The data includes text and timestamps of the posts and the replies by the employee bloggers in the firm, along with the employment information of those bloggers. We have also collected author provided community labels on the posts and user provided tags indicating the topic of the post. In addition we have also collected data on who is reading whose blogs and posts along with the timestamp from the webserver logs of the blog system. This essay starts with a brief study of the influence of different online and off-line ties on formation of two important ties in the blog social network: citation and reply. Then it proceeds to develop a framework for analyzing the conversation in the blogs. It shows that by encoding the actors, the text, and the time stamps of a conversation in a higher order tensor and performing a factorization we can identify dominant topics of conversation, important people behind each conversation and how

the topics have evolved over time. This work brings together ideas from the text data mining literature and the social network analysis literature. Initial analysis considers the ties between bloggers encoded in their replies to each others posts. I propose to extend the analysis by including reading ties between people which is the basis of other ties in the blog social network. In addition, number of times a blog is read also provides a measure of influence of the blog. Therefore, it can shed a different light on which bloggers are important. The other endeavour of this essay is to evaluate the results of the tensor factorization in a task based manner. For this evaluation we use the author provided community labels on the blog posts as "gold standard". The hypothesis is that since we take into account the people involved in each conversation in addition to the text in the conversation, the tensor factorization should enable us to better identify communities in the posts than if we used only the text in the document as is done in document clustering for topic discovery. Our initial tests show that it is indeed the case. The second task based evaluation I propose to do is to compare the accuracy of communities discovered by tensor factorization with the communities discovered by graph partitioning methods. The hypothesis here is that by using the text as the label of the tie between bloggers, we should be able to do a better job of identifying communities than by using just the network information.

The proposal is organized as follows. Chapter 2 describes the hierarchical topic discovery in document streams. Chapter 3 describes the multi-component rating collaborative filtering. Chapter 4 describes the data collected, tensor factorization methods developed and results obtained so far in the intra-organizational blog data analysis.

CHAPTER 2

Discovering topic hierarchy in document streams

Abstract

Incremental hierarchical text document clustering algorithms are important in organizing documents generated from streaming on-line sources, such as, Newswire and Blogs. However, this is a relatively unexplored area in the text document clustering literature. Popular incremental hierarchical clustering algorithms, namely COBWEB and CLASSIT, have not been applied to text document data. We discuss why, in the current form, these algorithms are not suitable for text clustering and propose an alternative formulation for the same. This includes changes to the underlying distributional assumption of the algorithm in order to conform with the empirical data. Both the original CLASSIT algorithm and our proposed algorithm are evaluated using Reuters newswire articles and OHSUMED dataset, and the gain from using a more appropriate distribution is demonstrated.

2.1 Introduction

Document clustering is an effective tool to manage information overload. By grouping similar documents together, we enable a human observer to quickly browse large document collections (Cutting et al. 1992), make it possible to easily grasp the distinct topics and subtopics (concept hierarchies) in them, allow search engines to efficiently query large document collections (Liu and Croft 2004) among many other applications. Hence, it has been widely studied as a part of the broad literature of data clustering. One survey of existing clustering literature can be found in Jain et al(Jain et al. 1999).

The often studied document clustering algorithms are batch clustering algorithms, which require all the documents to be present at the start of the exercise and cluster the document collection by making multiple iterations over them. But, with the advent of online publishing in the World Wide Web, the number of documents being generated everyday has increased considerably. Popular sources of informational text documents such as Newswire and Blogs are continuous in nature. To organize such documents naively using existing batch clustering algorithms one might attempt

to perform clustering on the documents collected so far. But, this is extremely time consuming, if not impossible, due to the sheer volume of documents. One might be tempted to convert the existing batch clustering algorithms into incremental clustering algorithms by performing batch clustering on periodically collected small batches of documents and then merge the generated clusters. However, ignoring for the moment the problem of deciding on an appropriate time window to collect documents, there will always be a wait time before a newly generated document can appear in the cluster hierarchy. This delay would be unacceptable in several important scenarios, e.g., financial services, where trading decisions depend on breaking news, and quick access to appropriately classified news documents is important. A clustering algorithm in such a setting needs to process the documents as soon as they arrive. This calls for the use of an *incremental* clustering algorithm.

There has been some work in incremental clustering of text documents as a part of Topic Detection and Tracking initiative (Allan et al. 1998, Yang et al. 1998, Franz et al. 2001, Doddington et al. 2000) to detect a new event from a stream of news articles. But, the clusters generated by this task are not hierarchical in nature. Although, that was adequate for the purpose of new event detection, we believe this is a limitation. The benefits of using a hierarchy of clusters instead of clusters residing at the same level of granularity is twofold. First, by describing the relationship between groups of documents one makes it possible to quickly browse to the specific topic of interest. The second reason is a technical one. Finding the right number of clusters in a set of documents is an ill-formed problem when one does not know the information needs of the end user. But, if we present the user with a topic hierarchy populated with documents, which she can browse at her desired level of specificity, we would circumvent the problem of finding the right number of clusters while generating a solution that would satisfy users with different needs.

In spite of potential benefits of an incremental algorithm that can cluster text documents as they arrive into a informative cluster hierarchy, this is a relatively unexplored area in text document clustering literature. In this work we examine a well known incremental hierarchical clustering algorithm COBWEB that has been used in non-text domain and its variant CLASSIT. We discuss why they are not suitable to be directly applied to text clustering and propose a variant of these algorithm that is based on the properties of text document data. Then we evaluate both the algorithm using real world data and show the gains obtained by our proposed algorithm.

2.1.1 Contribution of this research

In this chapter we demonstrate methods to carry out incremental hierarchical clustering of text documents. Specifically, the contributions of this work are:

1. A COBWEB-based algorithm for text document clustering where word occurrence attributes follow Katz's distribution.

2. Evaluation of the existing algorithms and our proposed algorithm on large real world document datasets.

In Section 2.2 we briefly review the text clustering literature. In Section 2.3 we describe key properties of text documents that are central to this work. In Section 2.4 we explain the contributions of our work. In Section 2.5 we describe the cluster quality metrics that we have used to evaluate the results obtained. In Section 2.6 we explain the setup of the experiment and discuss the results. In Section 2.7 we conclude with scope for future research.

2.2 Literature review

Clustering is a widely studied problem in the Machine Learning literature (Jain et al. 1999). The prevalent clustering algorithms have been categorized in different ways depending on different criteria, such as hierarchical vs. non-hierarchical, partitional vs. agglomerative algorithms, deterministic vs. probabilistic algorithms, incremental vs. batch algorithms, etc. Hierarchical clustering algorithms and non hierarchical clustering algorithms are categorized based on whether they produce a cluster hierarchy or a set of clusters all belonging to the same level. Different hierarchical and non-hierarchical clustering algorithms for text documents have been discussed by Manning and Schutze(Manning and Schütze 2000). Clustering algorithms can be partitional or agglomerative in nature. In a partitional algorithm one starts with one large cluster containing all the documents in the dataset and divides it into smaller clusters. On the other hand, an agglomerative clustering algorithm starts with all documents belonging to their individual clusters and combines the most similar clusters until the desired number of clusters are obtained. Deterministic clustering algorithms assign each document to only one cluster, while probabilistic clustering algorithms produce the probabilities of each item belonging to each cluster. The former is said to make "hard" assignment while the later is said to make "soft" assignments. Incremental clustering algorithms make one or very few passes over the entire dataset and they decide the cluster of an item as they see it. But, the batch clustering algorithms iterate over the entire dataset many times and gradually change the assignments of the items to the cluster so that a clustering criterion function is improved. One such criterion function is the average similarity among documents inside the clusters formed. Another criterion function is the average similarity between a document in a cluster and documents outside the cluster. The first criterion is called *average internal similarity* and the second criterion is called *average external similarity*. In a clustering solution we would want high average internal similarity, because that would mean that our clusters are composed of similar items. We would also want low average external similarity because that would mean our clusters are dissimilar, i.e., they do not overlap. The final set of clusters is produced after many iterations when no further improvement of the cluster assignment is possible.

Clustering to browser large document collections (Scatter/Gather) Cutting et al. is one of the first to suggest a cluster aided approach, called Scatter/Gather, to browse large document collections(Cutting et al. 1992). It describes two fast routines named

Buckshot and Fractionation to find the centroids of the clusters to be formed. Then it assigns the documents in the collection to the nearest centroid and recomputes the centroids iteratively until very little or no improvement is observed. The last step is similar to the Simple K-means clustering except that in Simple K-means initially one randomly assigns k items as centroids of k clusters (Manning and Schütze 2000). Note that k is a fixed user provided number. Buckshot finds the k centers in the document datasets by drawing a sample of $\sqrt{kn}$ documents and clustering them into k clusters using an agglomerative hierarchical clustering routine. The agglomerative hierarchical clustering algorithms have a time complexity of $O\left(n^2\right)$. By drawing a random sample of size $\sqrt{kn}$, the time complexity is reduced to $O(kn)$. Fractionation, on the other hand, finds k centroids in the following manner. It divides the set of documents into buckets of size m, where $m > k$. Then it clusters each bucket into ρm clusters, where $\rho < 1$ and is a constant. Then it repeats the process of partitioning the data and clustering them treating each of the formed cluster as a one data item, until k clusters are obtained. Cutting et al. have shown that Fractionation has a time complexity of $O\left(mn\right)$. The center of the clusters formed by the two methods are returned as the starting points for the Simple K-means clustering routine. With the help of these two routines they have proposed a cluster aided approach to browse document collections in which the program presents the user with a set of clusters for the document dataset (*Scatter*) along with their descriptive labels. Then the user can select the clusters which interest her and submit them to the program. The program merges the documents contained in those clusters (*Gather*) and clusters them again. This process is repeated until the user's information need is met or the user decides to stop the process. The recursive clustering idea proposed in Scatter/Gather can be effective in browsing large document sets, especially when one does not know enough about the documents to query a deployed search engine using key words. This concept loosely parallels the idea of organizing documents into a hierarchy of topics and subtopics, except that the organization in this case is guided by the user and executed by a clustering routine. However, Scatter/Gather has its limitations. It is a batch clustering routine, hence it cannot be used in some important scenarios as described in subsection. Another limitation that Scatter/Gather shares with many other clustering algorithms is that it requires the input of k, the number of clusters to present the user. A value of k different from the number of subtopics in the collection might lead to meaningless clusters.

Right number of clusters Finding the right number of clusters in a non-hierarchical clustering exercise is often a difficult problem (Smyth 1996). The approaches suggested in the literature can, in general, be divided into two groups (Chakrabarti 2002). The first approach is a multi-fold cross validation one with likelihood as the objective function, in which one fits a series of mixture models with different numbers of components to a subset of the data called *training data* and computes the likelihood of each model given the remaining subset of the data called *testing data*. The model that results in the highest likelihood is selected. The second approach also fits a mixture model to the data and computes the likelihood of the model given the *entire* dataset

using different number of clusters, but it penalizes a model with a higher number of clusters for increased complexity. Observe that a higher number of clusters can be made to fit any dataset better than a lower number of clusters. Hence, by penalizing a clustering solution for its complexity one can achieve a trade off between fitness, or likelihood, of the model and its complexity, which is optimized at the right number of clusters. One such work has been done by Cheeseman and Stutz in their AUTOCLASS algorithm(Cheeseman and Stutz 1996). Other such works include Bayesian Information Criteria and Minimum Descriptor Length criteria (Figueiredo and Jain 2002). A different approach has been suggested in Liu et al.(Liu and Croft 2004) for clustering text documents. It uses stability of clustering solutions over multiple runs at each of a set of cluster counts to decide the right number of clusters for the document dataset.

Even when the "right" number of clusters can be determined by an algorithm based on some criterion, human observers often differ from each other about the clusters existing in the dataset and what should be the right number of clusters. One alternative solution is to generate a hierarchy of clusters, also called a *dendrogram*, with all the documents belonging to a single cluster at the top of the hierarchy, each document in its individual cluster at the lowest level of the hierarchy and intermediate number of clusters at levels between the two. Thus, the user can look at the desired level in the hierarchy and find a number of clusters that meets her requirement (Manning and Schütze 2000, Jain et al. 1999).

Incremental document clustering

As part of Topic Detection and Tracking (TDT) initiative (Allan et al. 1998, Yang et al. 1998, Franz et al. 2001, Doddington et al. 2000) some experiments have been done in incrementally clustering text documents. The TDT initiative is a DARPA sponsored project started to study and advance the state of the art in detection and tracking of new events in stream of news broadcast and intelligence reports. The identified tasks of TDT are Story Segmentation, Retrospective Topic Detection, On-line New Event Detection, Topic Tracking and Link Detection. The Story Segmentation task involves breaking a stream of text or audio data without story delimiters into its constituent stories. Retrospective topic detection involves detecting new events in the already collected set of documents. On-line new event detection involves identifying a new event, e.g., an earthquake or a road accident, in a new document. Tracking involves keeping track of evolution of an event by assigning the incoming news stories to their corresponding events. Among these tasks the on-line new event detection task involves incremental clustering. In this task a decision is made, after observing a new item, whether it belongs to one of the existing clusters, or it belongs to a new cluster of its own.

The TDT team at the Carnegie Mellon University (CMU) uses a threshold-based rule to decide whether a new document is another story of one of the detected events or it belongs to a new event of its own. If the maximum similarity between the new document and any of the existing clusters is more than a threshold (t_c) the new document is said to belong to the cluster to which it is most similar and it is merged to

the cluster. If the maximum similarity is less than t_c but more than another threshold, t_n, then the document is assumed to be an old story but it is not merged to any cluster. If the maximum similarity is less than t_n, then the document is accepted to be about a new event and a new cluster is formed. They have also investigated adding a time component to the incremental clustering. In this experiment, similarities of a new document to each of the past m documents are computed but they are weighted down linearly depending on how old the past documents are. If the similarity scores computed in this manner are less than a preset threshold then the new document is presumed to be about a new event. This work finds that use of time component improves the performance of new event detection task.

TDT team at the University of Massachusetts Amherst (UMASS) takes a variable thresholding approach to the on line event detection task(Allan et al. 1998). For each document that initiates a new cluster the top n words are extracted and called a *query vector*. The similarity of the query vector to the document from which the query was extracted defines an upper bound on the threshold required to be met by a document to match the query. A time dependent component is also used in the variable threshold that makes it harder for a new documents to match an older query. When a new document d_j is compared to a past query q_i the threshold is computed as $0.4 + p \times (\mathrm{sim}(q_i, d_i) - 0.4) + \mathrm{tp} \times (j - i)$, where $0 < p < 1$ and tp, a time penalty factor, are tunable parameters. q_i is the query generated from document d_i. Such threshold is computed for all existing queries q_is. If the similarity of the new document d_j does not exceed any of the thresholds then the document is assigned to a new cluster and a query is computed for the document, else it is added to the clusters assigned to the queries it triggers. The newly generated cluster is said to have detected a new news event.

Outside the TDT initiative, Zhang and Liu in a recent study have proposed a competitive learning algorithm, which is incremental in nature and does not need to be supplied with the correct number of clusters (Zhang and Liu 2004). The algorithm, called *Self Splitting Competitive Learning*, starts with a prototype vector that is a property of the only cluster present initially. During the execution of the algorithm the prototype vector is *split* and updated to approximate the centroids of the clusters in the dataset. The update of the property vector is controlled, i.e., when a new data point is added to the cluster the prototype vector is updated only if the data point is near enough to the prototype. This determined by another *property* vector that starts away from the prototype and zeroes on to it as more and more data points are added. Time for splitting the cluster associated with the prototype is determined based on a threshold condition. When there are more than one prototype a new data point is added to the prototype nearest to it. They have demonstrated their algorithm over text snippets returned from search engines as a response to a query. However, the success of this algorithm on datasets with longer text documents is yet to be demonstrated.

Yet another on-line algorithm called *frequency sensitive competitive learning* has been proposed and evaluated on text datasets by Banerjee and Ghosh(Banerjee 2003), which is designed to produce clusters of items of approximately equal sizes. In this work a

version of the K-means clustering algorithm called *spherical K-means* has been modified so that the dispersion of the distributions associated with the clusters reduces as more and more data points are added to them. This makes larger clusters less likely candidates for a new data point than the smaller clusters. Thus, the algorithm is tailored to produce clusters which are more or less equal in size.

All of these algorithms produce non-hierarchical clustering solutions, which foregoes the opportunity to use clustering as an aid to detect topic and subtopic structure within a large document collection. Also, TDT experiments effectively exploit the information in the time stamp available with news stories, i.e., assumes that news stories that describe the same event will occur within a brief span of time. Such information may not always be available.

Incremental Hierarchical Clustering: Nominal Attributes

Methods have been proposed in the non-text domain to cluster items in an incremental manner into hierarchies. Most notable among them is the COBWEB algorithm by Fisher (Fisher 1987) and its derivative CLASSIT (Gennari et al. 1989). COBWEB is an algorithm to incrementally cluster data points with nominal attributes into cluster hierarchies. The problem of incremental hierarchical clustering can be defined as follows:

Problem Definition 1. Given an item with N attributes and an existing cluster hierarchy of items which the same attributes, how do we assign the item to the cluster in the cluster tree that maximizes a predefined cluster quality measure.

At the heart of COBWEB is a cluster quality measure called Category Utility.

Let $C_1, \ldots, C_K$ be the child clusters of a cluster C_p. The Category Utility of $C_1, \ldots, C_K$ is computed as

$$CU_p[C_1, \ldots, C_K] = \frac{\sum_{k=1}^{K} P(C_k) \sum_i \sum_j [P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij} \mid C_p)^2]}{K}, \tag{2.1}$$

where,

$P(C_k)$ =Probability of a document belonging to the parent cluster C_p belongs to the child cluster C_k.

A_i = The ith attribute of the items being clustered (say $A_1 \in$ {male, female}, $A_2 \in$ {Red, Green, Blue}; assumed to be a multinomial variable),

V_{ij} = jth value of the ith attribute (say, V_{12} indicates "female"),

The $P(A_i = V_{ij} \mid C_k)^2$ is the expected number of times we can correctly guess of the value of multinomial variable A_i to be V_{ij} for an item in the cluster k when one follows a probability matching guessing strategy. For example, if we have a variable that takes values A, B and C with probabilities 0.3, 0.5 and 0.2, and we randomly predict that the variable takes value A 0.3 fraction of the time, B 0.5 fraction of the time and C 0.2 fraction of the time, we would be correct in predicting A $0.3 \times 0.3 = 0.09$ fraction of the time, B 0.25 fraction of the time and C 0.04 fraction of the time. A good cluster, in

Algorithm `CobWeb` (Adapted from Fisher's original work)
function COBWEB(item, root)
Update the attribute value statistics at the root
If root is a leaf node
then
Return the expanded node that accommodates the new object
else
Find the best child of the root to host the item and perform the qualifying step (if any) among the following:

1. Create a new node for the item instead of adding it to the best host, if that leads to improved Category Utility.
2. Merge nodes if it leads to improved Category Utility and call COBWEB(item, Merged Node)
3. Split node if it leads to improved Category Utility and call COBWEB(item, root)

If none of the above steps are performed **then**
Call COBWEB(item, best child of root)
end if
end if

Figure 2.1: COBWEB control structure.

which the attributes of the items take similar values, will have high $P(A_i = V_{ij}|C_k)$ values, hence high score of $\sum_j P(A_i = V_{ij} \mid C_k)^2$. COBWEB maximizes sum of $P(A_i = V_{ij} \mid C_k)^2$ scores over all possible assignment of a document to children clusters. When the algorithm assigns a new item to a child node of the node p, it assigns the item in such a manner that the total *gain* in expected number of correct guesses by moving an item from p to its child node, $\sum_i \sum_j [P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij} \mid C_p)^2]$, is maximized. In this manner the algorithm maximizes the utility function for each node to which a new item is added.

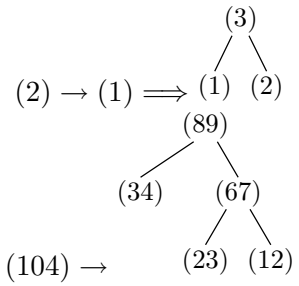
The COBWEB control structure is shown in Fig 2.1.

An illustration of the clustering process is given in Figure 2.2.

Assume that there is only one attribute of interest called t and it takes values in $\{A, B, C\}$. Also assume that we have three items a, b and c with t value A, B and C respectively. Further assume that the objects are presented in the order specified, i.e. first a followed by b which is followed by c.

After the first two items are presented the following cluster configuration is arrived without any computation of category utility (First part of Figure 2.2).

C_3 is the root cluster and C_1 and C_2 are two child clusters each containing one item. $P(C_1)$ is the probability that a document randomly picked from its parent cluster of C_1, i.e., C_3, belongs to C_1. Similarly for C_2.



Addition of a new item (2) to a leaf node (1)

Let (104) be a new item.
(@89) Which node should the new item be added to? (34) or (67)
or should it belong to a cluster of its own next to (34) and (67)?
Use Category Utility comparison as described in Fig 2.1. *Let the answer be (67)*
(@67) Which node should the new item be added to? (23) or (12)
⋮

Figure 2.2: COBWEB illustrated

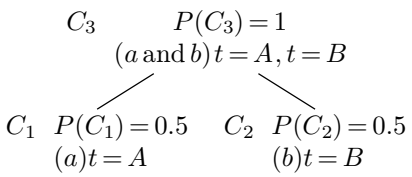


Figure 2.3: COBWEB: After first two items are added.

Let's add the third item c to the root node. We can add it at the level of C_1 and C_2 (level 2) as another cluster C_3, or we can add it *in* C_1 or C_2 that will delegate the item c to the third (a new) level. So, our options are (omitting the c within (b, c) configuration that is analogous to the c within (a, c) configuration described below):

At this point Category Utilities of the two configurations let us decide which configuration to choose. Note that we need to compute category utility of the two partitions of the root clusters. They can be computed using expression (2.1) as described below.

For the first configuration in Figure 2.4 the parent cluster is C_3 and the child clus-

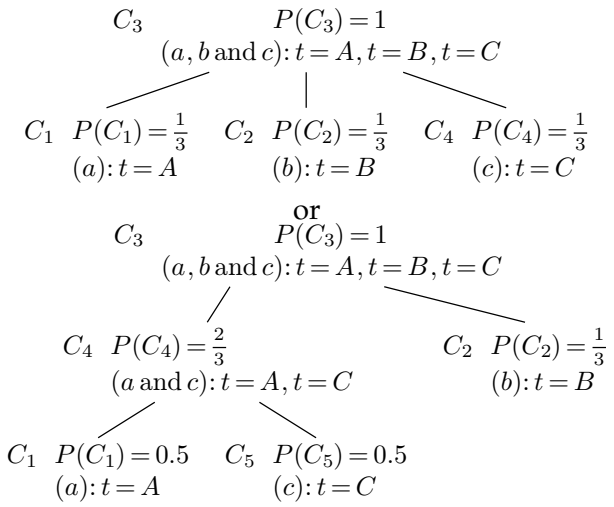


Figure 2.4: COBWEB: Two partitions of the root cluster.

ters are C_1, C_2 and C_4. The category utility of this configuration is:

$$\begin{aligned}
\text{CU}^1 &= \frac{\sum_{k=\{1,2,4\}} P(C_k) \left[\sum_{A_i=t} \sum_{t=\{A,B,C\}} P(t|C_k)^2 - \sum_{A_i=t} \sum_{t=\{A,B,C\}} P(t|C_3)^2\right]}{3} \\
&= \frac{1}{3}\left[\frac{1}{3}\left\{1^2 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)\right\}\right. \\
&\quad + \frac{1}{3}\left\{1^2 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)\right\} \\
&\quad \left. + \frac{1}{3}\left\{1^2 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)\right\}\right] \\
&= \frac{2}{9}
\end{aligned}$$

For the second configuration in Figure 2.4 the parent cluster is C_3 and the child clusters are C_4 and C_2.

$$\begin{aligned}
\text{CU}^2 &= \frac{\sum_{k=\{4,2\}} P(C_k) \left[\sum_{A_i=t} \sum_{t=\{A,B,C\}} P(t|C_k)^2 - \sum_{A_i=t} \sum_{t=\{A,B,C\}} P(t|C_3)^2\right]}{2} \\
&= \frac{1}{2}\left[\frac{2}{3}\left\{\left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right) - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)\right\}\right. \\
&\quad \left. + \frac{1}{3}\left\{1^2 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)\right\}\right] \\
&= \frac{1}{6}
\end{aligned}$$

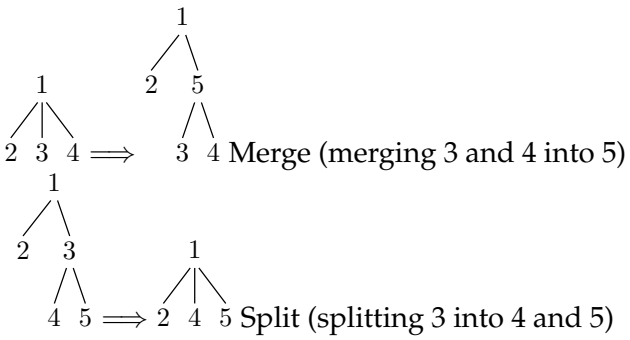


Figure 2.5: Merge and split operations illustrated.

Since, $\mathrm{CU}^1 > \mathrm{CU}^2$ we select configuration 1 over configuration 2. Looking at the Figure 2.4, it is intuitive to make a new cluster for the third item, because, it has an attribute value not seen in any of the existing categories.

There is one more possible configuration, where c is added below C_2 instead of C_1, but that is symmetrical to the second configuration in Figure 2.4. So, the analysis will be identical to the one shown in previous paragraph.

Incremental clustering algorithms, such as COBWEB, are sensitive to the order in which items are presented (Fisher 1987). COBWEB makes use of *split* and *merge* operations to correct this problem. In the merge operation the child nodes with highest and second highest Category Utility are removed from the original node and made child nodes of a new node, which takes their place under the parent node. In the split operation the best node is removed and its child nodes are made children of the parent of the removed node. Merge and split operations are only carried out if they lead to a better Category Utility than obtainable by either assigning the item to existing best node or to a new cluster of its own. By using these two operators, the algorithm remains flexible on the face of change in property of data items in the subsequent observations.

Incremental Hierarchical Clustering: Numerical Attributes

We now consider an extension of the COBWEB from nominal attributes to numerical attributes. Gennari et al.(Gennari et al. 1989) has shown that in order to use COBWEB for data items with numeric, rather than nominal, attribute values we need to make some assumption about the distribution of attribute values. When the values of each attribute follow a normal distribution, they have shown that the Category Utility function can be written as

$$CU_p[C_1, \ldots, C_k] = \frac{\sum_k P(C_k) \sum_i \left(\frac{1}{\sigma_{ik}} - \frac{1}{\sigma_{ip}}\right)}{K}$$

where,

σ_{ip} = standard deviation of the value of the attribute i in parent node p, and

σ_{ik} = standard deviation of the value of the attribute i in the child node k.

This algorithm is known as the CLASSIT algorithm.

We have not seen any prior application of either of these algorithms to text clustering. Hence, their performance on text document data is uncertain at the time of this work. Further, word occurrence counts, attributes of text documents that are commonly used to represent a document, follow a skewed distribution—unlike the Normal distribution (Figure 2.6). Also, Normal distribution assumes that the attributes are Real numbers, but, word occurrence counts are Nonnegative Integers. They can not be treated as nominal attributes either, because the occurrence counts are not contained in a bounded set, which one would have to assume while treating them as nominal attributes. A more suitable distribution for such count data is Negative Binomial, or Katz's distribution (Katz 1996).

Our work proposes to improve upon the original COBWEB algorithm using distributional assumptions that are more appropriate for word count data.

2.3 Text Documents and word distributions

Text, as we commonly know it, is available in the form of unstructured documents. Before we can use such documents for classification or clustering, we need to convert them to items with attributes and values. A popular way of converting the document to such a form is to use the words[1] in a document as attributes and the number of times the word occurs in the document, or some function of it, as the value of the attribute. This is called the "Bag of Words" approach. One consequence of using such a method to convert documents to an actionable form is that one foregoes information contained in the order of the word. Despite this drawback, the bag-of-words approach is one of the most successful and widely used method of converting text documents into actionable form.

Several attempts has been made to characterize the distribution of words across documents. This is useful in judging the information content of a word. For instance a word that occurs uniformly in every document of the corpus, e.g., "the" is not as informative as a word that occurs frequently in only a few, e.g., "Zipf".

Occurrence statistics of a word in a document can be used along with the information content of the word to infer the topic of the document and cluster documents of similar topic into same group—as is done in this work. Manning and Schutze have discussed several models to characterize the occurrence of words across different documents (Manning and Schütze 2000).

[1]Through out this chapter we shall use *word* and *term* interchangeably to refer to the same thing, i.e., a contiguous sequence of alphanumeric characters delimited by non-alphanumeric character(s). E.g. the first *word* or *term* in this footnote is "Through".

2.3.1 Models based on Poisson distribution

Poisson

The Poisson distribution has been used to model number of times a word occurs in a document. The probability of a word occurring k times in a document is given by

$$P(k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{2.2}$$

where, λ is a rate parameter. However, from empirical observations, it has been found that Poisson distribution tends to over estimate the frequency of informative words (content words) (Manning and Schütze 2000).

Two Poisson Model

There have been attempts to characterize the occurrence of a word across documents using a mixture of Poisson distributions. One such attempts uses two Poisson distributions to model the probability of a word occurring a certain number of times in a document. One of the distributions captures the rate of the word occurrence when the word occurs because it is topically relevant to the document. The second distribution captures the rate of the word occurrence when the word occurs without being topically relevant to the document. This mixture of two probability distributions has the probability density function:

$$P(k) = \alpha \frac{\lambda_1^k e^{-\lambda_1}}{k!} + (1-\alpha)\frac{\lambda_2^k e^{-\lambda_2}}{k!} \tag{2.3}$$

where, α is the probability of the word being topically relevant and $1-\alpha$ is the probability of the word being topically unrelated to the document.

It has been empirically observed that, although the two Poisson model fits the data better than single Poisson model(Bookstein and Swanson 1975), a spurious drop is seen for the probability of a word occurring twice in a document(Katz 1996). The fitted distribution has lower probability for a word occurring twice in a document than it occurring three times, i.e., it predicts that there are fewer documents that contain a word twice than there are documents that contain the same word three times. But, empirically it has been observed that document count monotonically decreases for increasing number of occurrences of a word (see Figure 2.6).

Negative Binomial

A proposed solution to the above problem is to use a mixture of more than two Poisson distributions to model the word occurrences. A natural extension of this idea is to use a Negative Binomial distribution, which is a gamma mixture of infinite number of Poisson distributions(Frederick Mosteller 1983). The probability density functions of a Negative Binomial distribution is given below,

$$P(k) = \binom{k+r-1}{r-1} p^r (1-p)^k, \tag{2.4}$$

where p and r are parameters of the distributions.

Although the Negative Binomial distribution fits the word occurrence data very well it can be hard to work with because it often involves computing a large number of coefficients(Manning and Schütze 2000). This has been confirmed in our analysis (see Expressions (2.28) and (2.29) in Section 2.4.2).

Zero inflated Poisson

When we observe the word occurrence counts in documents, we find that most words occurs in only a few documents in the corpus. So, for most of the words, the count of documents where they occur zero times is very large (see Figure 2.6). Looking at the shape of the empirical probability density function we attempt to model the occurrence counts using a Zero Inflated Poisson distribution, which assigns a large probability mass at the variable value 0 and distributes the remaining probability mass over rest of the occurrence counts according to a Poisson distribution.

The probability density function of Zero Inflated Poisson distribution is given by

$$P(k) = (1-\alpha)\delta_k + \alpha\frac{\lambda^k e^{\lambda}}{\lambda!}, k = 0, 1, 2\ldots \tag{2.5}$$

where,

$$\delta_k = \begin{cases} 1, \text{iffk} = 0 \\ 0, \text{otherwise} \end{cases}$$

As we shall demonstrate in Section 2.3.3, this distribution does not fit text data as well as the Negative Binomial or the Katz's distribution.

2.3.2 Katz's K-mixture model

This distribution, proposed by Katz(Katz 1996), although simple to work with, has been shown to model the occurrences of words in the documents better than many other distributions such as Poisson and Two Poisson, and about as well as the more complex Negative Binomial distribution(Manning and Schütze 2000). Katz's distribution assigns the following probability to the event that word i occurs k times in a document[2].

$$P(k) = (1-\alpha)\delta_k + \frac{\alpha}{\beta+1}\left(\frac{\beta}{\beta+1}\right)^k \tag{2.6}$$

$\delta_k = 1 \,\text{iffk} = 0 \,\text{and}\, 0 \,\text{otherwise}\,.$

The MLE estimates of parameters α and β are:

$$\beta = \frac{\text{cf} - \text{df}}{\text{df}} \tag{2.7}$$

[2]In this section we shall discuss the case of one word, the ith word. Hence, we shall drop the subscript i from the equations and expressions.

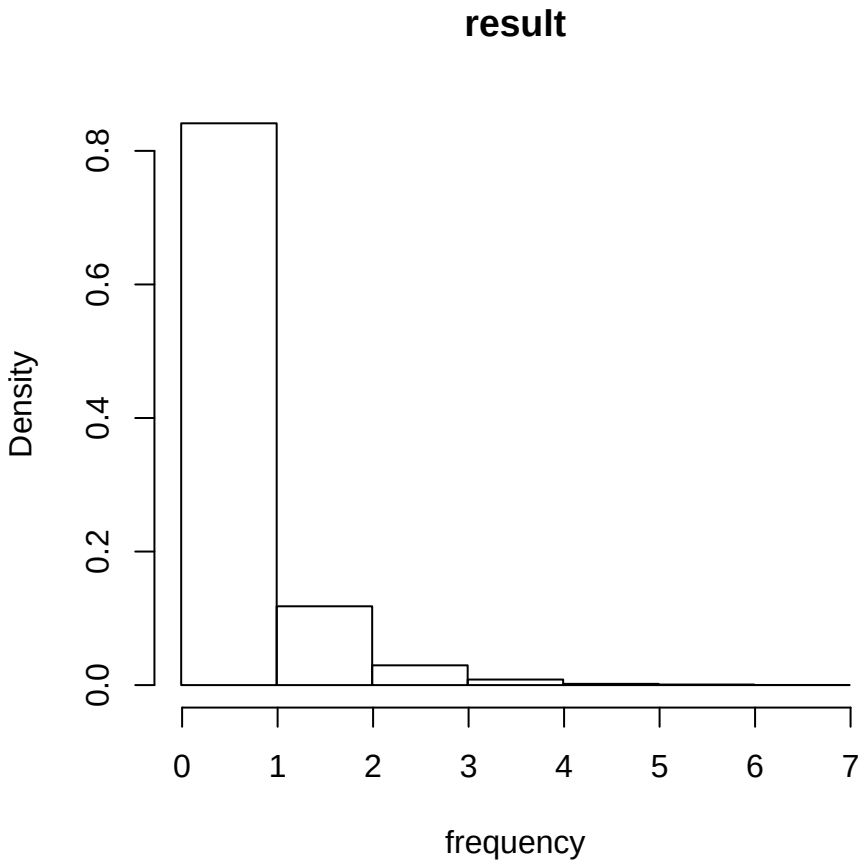


Figure 2.6: The occurrence of a typical word ("result") across different documents in our test collection.

$$\alpha = \frac{1}{\beta} \times \frac{\text{cf}}{N} \tag{2.8}$$

cf = *collection frequency* = number of times word i occurred in the document collection obtained by adding up the times the word occurred in each document. Here, a collection can be whatever we deem our universe of documents to be. It can be the entire corpus of documents or a subset of it.

df = *document frequency* = number of documents in the entire collection that contain the word i.

From (2.6) it follows that

$$\begin{aligned} P(0) &= 1 - \alpha + \frac{\alpha}{\beta + 1} \\ &= 1 - \frac{\text{df}}{N} \\ &= 1 - \Pr(\text{the word occurs in a document}) \\ &= \Pr(\text{the word does not occur in a document}) \end{aligned} \tag{2.9}$$

Also, it follows that

$$P(k) = \frac{\alpha}{\beta + 1} \left(\frac{\beta}{\beta + 1} \right)^k, k = 1, 2, \ldots \tag{2.10}$$

Substituting p for $\frac{\beta}{\beta+1}$, we have

$$P(k) = \alpha(1 - p)p^k \tag{2.11}$$

Let's define a parameter p_0 as

$$p_0 = P(0) \tag{2.12}$$

using (2.7) we find that

$$\begin{aligned} p &= \frac{\frac{\text{cf} - \text{df}}{\text{df}}}{\frac{\text{cf}}{\text{df}}} \\ &= \frac{\text{cf} - \text{df}}{\text{cf}} \\ &= \frac{\Pr(\text{the word repeats in a document})}{\Pr(\text{the word occurs in a document})} \\ &= \frac{\Pr(\text{the word repeats} \bigcap \text{the word occurs})}{\Pr(\text{the word occurs})} \\ &= \Pr(\text{the word repeats} — \text{the word occurs}) \end{aligned} \tag{2.13}$$

Hence, $1 - p$ can be interpreted as the probability of the word occurring only once. Or, it can be thought of as a scaling factor used to make (2.11) and (2.12) together a valid probability density function.

We can write Expression (2.6) for $k = 0$, using p as

$$\begin{aligned} P(0) &= (1-\alpha) + \alpha(1-p) \\ &= 1 - \alpha + \alpha - \alpha p \end{aligned}$$

Hence, α in terms of p_0 and p is

$$\begin{aligned} p_0 &= 1 - \alpha p \\ \Rightarrow \alpha p &= 1 - p_0 \\ \Rightarrow \alpha &= \frac{1-p_0}{p} \end{aligned} \tag{2.14}$$

Expression (2.11) can now be written as

$$P(k) = (1-p_0)\,(1-p)\,p^{k-1} \tag{2.15}$$

when $k > 0$.

Using Expressions (2.12) and (2.15), we can fully specify the Katz's distribution. The two parameters are p_0 and p, which can be estimated as (see Expressions 2.9 and 2.13)

$$\widehat{p_0} = 1 - \frac{\mathrm{df}}{N} \tag{2.16}$$

and

$$\hat{p} = \frac{\mathrm{cf} - \mathrm{df}}{\mathrm{cf}} \tag{2.17}$$

It can be shown that if a distribution is defined by Expressions (2.12) and (2.15), then the estimates (2.16) and (2.17) are the MLE of the parameters p_0 and p (see Appendix A.1).

2.3.3 Fitness comparison

We estimated the parameters of Zero Inflated Poisson and Negative Binomial using the method of moment, and parameters for Katz's distribution using the Maximum Likelihood Estimate (MLE) method. The reason for using the method of moments and not the MLE is that for the Negative Binomial and the Zero Inflated Poisson distributions the MLE can only be found numerically, which is computationally complex for our task of incremental clustering. One can still use numerical methods to determine MLEs of the parameters of the distribution, which admittedly have better properties, if one is willing to pay the cost in terms of delay. In this work we shall limit ourselves to the estimates that have closed form expressions and can be computed efficiently, because our goal is to carry out the incremental document clustering in real time.

Zero Inflated Poisson

If the probability density function of a Zero Inflated Poisson distribution is given in the form of Expression (2.5), then the method of moment estimates of its parameters α and λ are

$$\hat{\lambda} = \frac{\text{Var}(X)}{\overline{X}} + \overline{X} - 1 \tag{2.18}$$

and

$$\hat{\alpha} = \frac{\overline{X}}{\lambda} \tag{2.19}$$

Negative Binomial

For the Negative Binomial distribution, parameters p and r can be estimated as

$$\hat{r} = \frac{\bar{X}^2}{\text{Var}(X) - \bar{X}} \tag{2.20}$$

$$\hat{p} = \frac{\bar{X}}{\text{Var}(X)} \tag{2.21}$$

For the Katz's distribution we used Expressions (2.16) and (2.17) to estimate the parameters p_0 and p.

We evaluated the fitness of these three distributions by computing the probabilities of the word occurrences using the estimated parameters, on three different datasets. For each dataset we selected the top 100 terms by their $\text{cf} \times \log(N/df)$ score. The distribution that has a higher likelihood than another can be considered a better fit to the data. For each term a pairwise comparison of fitness of different distributions is carried out in this manner. The results are shown in the form of three dominance matrices in Table 2.1. Each cell records the number of terms for which distribution for the row has 10% or higher likelihood than the distribution for the column.

It can be observed from the table that Katz's distribution, is not only easier to work with as we will see in Section 2.4, it also fits better than Zero Inflated Poisson (ZIP) and gives fitness comparable to Negative Binomial (NB) distribution.

2.4 Algorithms for text

2.4.1 COBWEB: when attribute values follow Katz's distribution

Category utility

Using words as attributes, we can derive the Category Utility function assuming that word occurrences follow Katz's distribution. For reference, the *Category Utility* formula as given in COBWEB is

$$\frac{1}{K}\sum_k P(C_k)\left[\sum_i\sum_j \left(P(A_i = V_{i,j}|C_k)^2 - P(A_i = V_{i,j}|C_p)^2\right)\right]$$

dataset		dominance table		
		NB	Katz's	ZIP
classic	NB	0	55	92
	Katz's	41	0	96
	ZIP	7	4	0
		NB	Katz's	ZIP
tr41	NB	0	41	98
	Katz's	58	0	98
	ZIP	2	2	0
		NB	Katz's	ZIP
k1a	NB	0	63	98
	Katz's	35	0	98
	ZIP	2	2	0

Table 2.1: Likelihood comparisons, count of likelihood of row distribution $>$ likelihood of col distribution$\times$1.1

Notice that for each attribute indexed i we need to compute

$$\sum_j \left(P(A_i = V_{i,j}|C_k)^2 - P(A_i = V_{i,j}|C_p)^2\right) \tag{2.22}$$

where, j is an index of value of the attribute i. In this case $V_{i,j}$ would take values 0, 1, 2 ... because we are working with count data.

Hence, the first part of Expression (2.22) can be written as

$$\mathrm{CU}_{i,k} = \sum_{f=0}^{\infty} P(A_i = f|C_k)^2 \tag{2.23}$$

Let's use $\mathrm{CU}_{i,k}$ to refer to the contribution of the attribute i towards the *Category Utility* of the cluster k.

Substituting Expressions (2.12) and (2.15) in Expression (2.23), we obtain

$$\mathrm{CU}_{i,k} = \sum_{f=0}^{\infty} P(A_i = f|C_k)^2 = \frac{1 - 2p_0(1-p_0) - p(1-2p_0)}{1+p} \tag{2.24}$$

Substituting estimates of p_0 and p from Expressions (2.16) and (2.17) in Expression (2.24), and simplifying, we get

$$\mathrm{CU}_{i,k} = \sum_{f=0}^{\infty} P(A_i = f|C_k)^2 = 1 - \frac{2 \times \mathrm{df}\left(N - \frac{\mathrm{cf} \times \mathrm{df}}{2 \times \mathrm{cf} - \mathrm{df}}\right)}{N^2} \tag{2.25}$$

where, df, cf, and N are counted in the category k.

Expression (2.25) specifies how to calculate the *Category Utility* contribution of an attribute in a category. Hence, the *Category Utility* of the CLASSIT algorithm, when the distribution of attributes follows Katz's model, is given by

$$\mathrm{CU}_p = \frac{1}{K}\sum_k P(C_k)\left[\sum_i \mathrm{CU}_{i,k} - \sum_i \mathrm{CU}_{i,p}\right] \tag{2.26}$$

where, $\mathrm{CU}_{i,k}$ is given by Expression (2.25).

2.4.2 COBWEB: when attribute values follow Negative Binomial distribution

The probability density function of the Negative Binomial distribution is

$$P(x) = \begin{pmatrix} x+r-1 \\ r-1 \end{pmatrix} p^r(1-p)^x \tag{2.27}$$

p and r are the parameters of the distribution, which are to be estimated from the data.

Category utility

Substituting Expression (2.27) in (2.23), we obtain the contribution of a word in a child cluster towards Category Utility

$$\mathrm{CU}_{i,k} = \sum_{x=0}^{\infty}\left(\frac{(x+r-1)!}{x!(r-1)!}p^r(1-p)^{x-1}\right)^2 \tag{2.28}$$

This expression cannot be reduced to any simpler form, although, it can be written using a hyper-geometric function in the following manner.

$$\mathrm{CU}_{i,k} = \frac{p_2^{2r}F_1\left(r,r,1,(1-p)^2\right)}{(1-p)^2} \tag{2.29}$$

One can use a library, such as the one available with Mathematica, to numerically evaluate ${}_2F_1(r,r,1,(1-p)^2)$. In our experience this computation is three orders of magnitudes more resource intensive than computing (2.25), the equivalent expression for Katz's distribution. As we described in Section 2.3.3, in this work we shall limit ourselves to the methods that will let us carry out incremental clustering in real time, i.e., in the time available between arrival of two documents.

For this reason and the reasons cited in Section 2.3.1 and 2.3.3, we shall fully explore only Katz's distribution and original CLASSIT algorithm based on Normal distribution in our work.

2.5 Cluster Evaluation Methods

2.5.1 Evaluating the clusters

One commonly used cluster quality measure is the purity of clustering solution. Purity of a cluster is defined as

$$p_k = \frac{\max_c\{\mathrm{CF}_k(c)\}}{N_k} \tag{2.30}$$

where,

- c is the index of classes
  - *class* is a pre-specified group of items
- k is the index of clusters
  - *cluster* is an algorithm generated group of items

$\mathrm{CF}_k(c)$ = number of items from class c occurring in cluster k. Or, the frequency of class c in cluster k.

N_k = number of items in class k.

Purity of the entire collection of clusters can be found by taking the average of the cluster qualities. Here, there are two kinds of averages one might consider: weighted or unweighted. If we assign a weight to each cluster proportional to the size of the cluster and take the weighted average then it is called *micro average*, since each of the documents get equal weight. If we instead want to give equal weight to each cluster, we compute the arithmetic average instead. This is called *macro average*. The first one is a document level evaluation, while the second one is a cluster level evaluation. Both these purity are greater than 0 and less than 1.

The drawback of relying only on purity to evaluate the quality of a set of clusters, becomes apparent in hierarchical clustering. When we collect clusters occurring at or near the lowest level of the hierarchy, we get clusters with very few documents in them. Hence, we obtain clusters with high purity score. In the limit, at the lowest level there are N clusters each containing only one item. Hence, $\max_c\{\mathrm{CF}_k(c)\}$ is 1 for each $k \in \{1, \ldots, N\}$ resulting in purity score of 1. We get larger clusters at a higher level in the hierarchy, which are more likely to contain documents belonging to different classes, leading to a lower purity score. This illustrates how purity score can be misleading when the number of clusters formed is different than the number of classes in the dataset. If we make more number of clusters than there are in the dataset we bias the purity score up. If we make less number of clusters than there are in the dataset we bias the purity score down.

To correct this problem, we define another score of the clustering solution in the following manner.

$$r_c = \frac{\max_k\{\mathrm{CF}_k(c)\}}{N_c}$$

where, N_c is the size of the class c. The other variables are as defined for the expression of the purity score in Expression (2.30). Here, also we can compute the micro average or the macro average to compute the score for the entire solution.

This is a purity computation with the clustering solution treated as the true classes of the data items and the human generated clusters as the solutions to be evaluated. Using this measure we evaluate how well the "true" *classes* in the datasets are represented in the clusters formed.

These metrics, p_k and r_c, have interpretations that parallel the *precision* and *recall* metrics, respectively, in information retrieval literature. Precision is the fraction of the retrieved documents that are relevant. Our p_k has the precision interpretation when we think of a cluster to retrieve documents from the class to which majority of its elements belong. On the other hand recall is the fraction of all the relevant documents that are retrieved. In the framework we described for p_k, our metric r_c has the recall interpretation.

Taking a cue from the F measure commonly used in IR literature to combine precision and recall, we computed the F score as the harmonic mean of the $\mathrm{P and R}$ values:

$$\frac{1}{F} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right) \tag{2.31}$$

The F score is the metric by which we shall measure the quality of our clusters.

2.5.2 Evaluating the hierarchy

Another question of interest when evaluating a hierarchical clustering algorithm is "To what extent the generated cluster hierarchy agree with the class hierarchy present in the data?". As we shall describe in Section 2.6, the datasets we have used in our experiments have a hierarchy of classes and provide us a rare opportunity to evaluate our generated cluster hierarchy for correctness. As a reminder, a *class* is a document category that has been provided to us as a part of the dataset. It is what the documents have been labeled with by an external entity and help us in evaluating how good our algorithm is. On the other hand, a *cluster* is a grouping of documents that our algorithm generates. It does so by grouping together the documents it considers similar.

Matching the generated cluster hierarchy with the existing class hierarchy is a non-trivial task. In stead, in this work we focus on measuring how often the sibling clusters in the generated hierarchy have sibling classes, i.e, how often children clusters of a parent cluster have children classes of the class that is assigned to the parent cluster. For instance, consider the generated cluster subtree shown in Figure 2.7.

In this case we have already determined the classes of child clusters[3]. To be able to measure if they are filed under the correct class, we need to find the class of the parent cluster. To do this we tabulate the parent classes of the child clusters and assign the

[3]At the lowest level each cluster has only one document and its class can be read from the data directly.

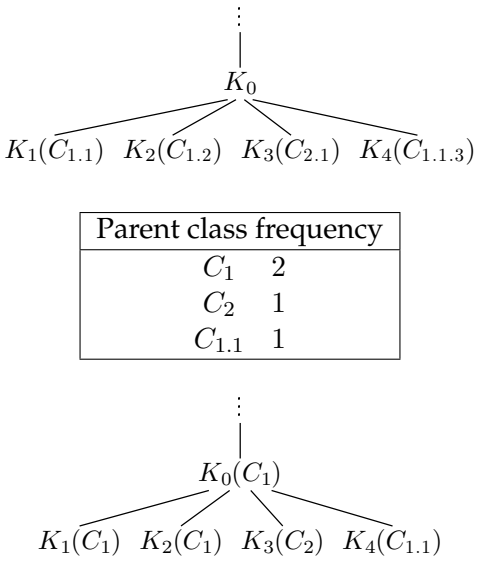


Figure 2.7: A sample subtree with the children nodes. Class labels of the children node are given in parenthesis.

most frequent parent class to the parent cluster K_0. So, in this case the parent cluster K_0 gets the label C_1. Then we evaluate this cluster configuration as if K_0 is merely a cluster of four other smaller entities, each of which has a class label same as the parent class of what they really have. This is equivalent of saying that as long as the children clusters of K_0 have children classes of the class of K_0, i.e., C_1 in this case, they are correct. Clusters with all other class labels that occur under that parent cluster are incorrect classifications by the algorithm. They should have been somewhere else.

So, in the above example the precision of K_0 would be $\frac{2}{4} = 0.5$. We compute this precision for all the *internal nodes* of the cluster tree and take their average (both micro average and macro average) to compute the overall precision of the hierarchy. This gives us a measure of how much the generated cluster hierarchy agree with the class hierarchy present in the data. We call it sibling precision score of the cluster hierarchy.

We needed to make a few decisions while evaluating the hierarchy in this manner. For instance, we used only the internal nodes to compute the precision of any node. This is because, often times leaf nodes co-exist with internal nodes as children of another internal node. In this case if we compute precision based on leaf nodes, i.e., single documents, then we are mixing the precision of the kind we described in Section 2.5.1 with the precision of the hierarchy and it is not clear how we should interpret the resulting number. Another decision that needed to be made was, what should we do if a child cluster has the broadest class label assigned to it? Since, we can not find a parent class for these classes, we explored the possibility of

1. dropping such child clusters from our evaluation and
2. treating them as their own parent cluster since, they are the broadest level classes.

In our experiments the results do not change much if we take either of these strategy. So, we shall report only the results we got by treating the broadest classes as their own parent classes.

2.6 Experiment setup and results

We evaluate our algorithm over two text document collections, i.e., Reuters-RCV1 and OHSUMED (88-91). These datasets were picked because of the presence of human labeled hierarchical class labels and reasonably large number of documents in them. They are described in more detail in the following section.

2.6.1 Reuters-RCV1

Incremental clustering algorithms process the data points only once and in the order in which they are presented and the order in which data points are present in the dataset influences the clusters produced[4]. Therefore, it is imperative that we test the incremental clustering algorithms with an ordering of data points that is similar to the what they are expected to receive during their deployment. As we envision the two algorithms in this work to be used to process streams of text documents from newswire, newsgroups, Blogs, etc., the natural ordering among the documents is determined by the time at which they are received. Therefore, we need a document dataset in which the time order of the documents is preserved. Reuters-RCV1(Lewis et al. 2004) is one such dataset.

Reuters-RCV1 dataset is a collection of over 800,000 English newswire articles collected from Reuters over a period of one year(20th Aug 1996 to 19th Aug 1997). These documents have been classified by editors at Reuters simultaneously under three category hierarchies: "Topic" hierarchy, "Industry" hierarchy and "Region" hierarchy. The Topic hierarchy contains four categories at the depth one of the tree, namely "Corporate/Industrial", "Economics", "Government/Social" and "Market". There are ten such categories in the Industry hierarchy. Some of them are "Metals and Minerals", "Construction", etc. The Region hierarchy has geographical locations, such as country names, and economic/political groups as categories. There are no finer sub-categories in the Region hierarchy.

The classification policy, also called The Coding Policy, requires that each document must have at least one Topic category and at least one Region category assigned to it. It also requires that each document be assigned to the most specific possible subcategory in a classification hierarchy. A document might be, and often is, assigned more than one categories from any one of the three category hierarchies. The documents are present in the dataset in the order in time in which they were collected.

[4]However, the ideal incremental clustering algorithm is expected to be insensitive to the order in which it encounters the data points. Such, characteristic is partly achieved by the COBWEB algorithm by its *split* and *merge* operators.

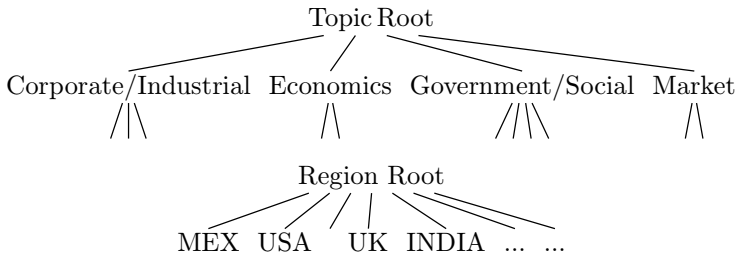


Figure 2.8: Three classification hierarchies.

number of documents	62935
number of unique words	93792
average document length	222
number of classes	259

Table 2.2: RCV1 dataset (First 30 days). Classes are the region classes

Evaluating clusters

Experiment setup For our experiments articles from the first 30 days of the Reuters-RCV1 dataset were used. There were 62935 articles. Stop words were removed from the documents and the terms were stemmed. Then the most informative terms were selected by their $\mathrm{cf} \times \log(N/df)$ scores to represent the documents. We repeated the experiments using 100 to 800 terms at step size of 100.

We have evaluated the clustering solutions for the correctness of assignment of documents to the clusters using the region categories, because (i) in the region class hierarchy all the assigned classes belong to one level and (ii) fewer articles are assigned multiple region class labels than they are assigned other class labels, suggesting that the region classes in the dataset do not overlap a lot. This allows us to evaluate out algorithm on a dataset with well defined classes. There were 259 region categories present in the selected documents. So, we have extracted 259 clusters from the dendrogram constructed by the clustering algorithms and measured their quality using the Region categories of the documents.

Results and Discussion The results of the clustering exercise is given in Table 2.3. We can see that Katz's distribution based CLASSIT algorithm dominates Normal distribution based CLASSIT algorithm across varying vocabulary sizes in both the micro and macro average of F scores.

As we can see Katz based CLASSIT algorithm consistently performs better than the Normal based CLASSIT algorithm on this dataset. However, we are cautious in interpreting the micro averaged-F score. Both of these algorithms produce clusters of widely different sizes, i.e., a few big clusters, a few more clusters of intermediate size and a lot of smaller clusters. The micro-averaged F score, is affected by it. Because, performance over a few good clusters dominates the entire performance metric. This

V	100		K	N
		micro	0.46	0.31
		macro	0.83	0.60
	200		K	N
		micro	0.45	0.43
		macro	0.81	0.74
	300		K	N
		micro	0.45	0.33
		macro	0.85	0.67
	400		K	N
		micro	0.45	0.42
		macro	0.79	0.74
	500		K	N
		micro	0.45	0.36
		macro	0.84	0.69
	600		K	N
		micro	0.45	0.42
		macro	0.82	0.76
	700		K	N
		micro	0.45	0.39
		macro	0.81	0.74
	800		K	N
		micro	0.45	0.30
		macro	0.83	0.61

Table 2.3: Cluster quality comparison on RCV1 data

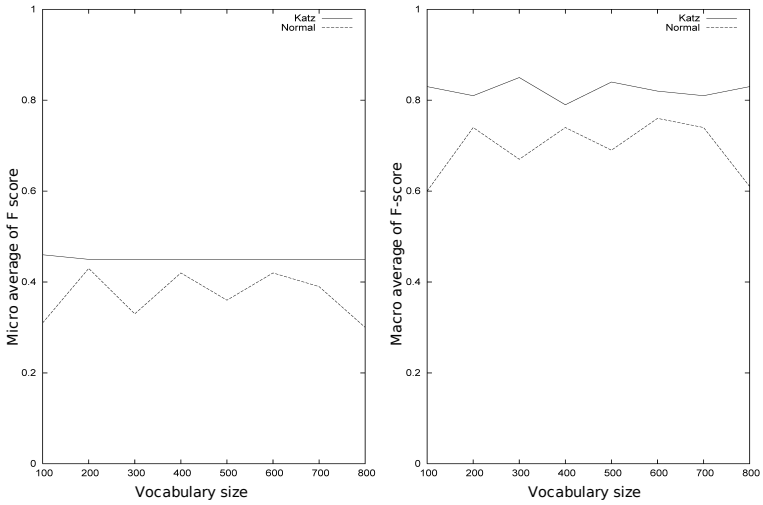


Figure 2.9: Cluster quality comparison on RCV1 data. The left panel shows the micro average of F-score and the right panel shows the macro average of the F-score.

explains the flat nature of the plot of micro averaged F score with Katz based CLASSIT. The larger of the clusters generated by the algorithm do not change much over different vocabulary sizes, so, the micro-averaged F score remains nearly constant. Therefore, we also compute the macro-averaged F score, where each cluster gets equal weight, and find that Katz based CLASSIT performs better than Normal based CLASSIT over a wide range of vocabulary sizes.

Evaluating hierarchy

We evaluate the generated cluster hierarchy using the *topic* hierarchy of classes[5] as our reference. There are 63 different topic codes in the documents we used, where as in the entire topic hierarchy there are 103 topic codes.

We pre-processed the documents using the steps described in the previous section. Evaluated the accuracy of the parent/child cluster configurations as described in Section 2.5.2. The results are given in Table 2.4.

The values in the table cells are the average sibling precision of internal nodes of the cluster hierarchy. As we can see there is no clear winner in this case, although, both the algorithms do reasonably well in assigning sibling classes under the same cluster. However, we must be careful to interpret these values as the correctness of the sibling classes getting grouped together and not as recovering all of the original class hierarchy.

[5]This can be obtained from (Lewis et al. 2004) Appendix 2.

V	Normal Macro avg	Katz Macro avg	Normal Micro avg	Katz Micro avg
100	0.925	0.956	0.814	0.959
200	0.924	0.935	0.797	0.943
300	0.926	0.874	0.825	0.871
400	0.92	0.866	0.814	0.789
500	0.918	0.896	0.812	0.871
600	0.922	0.841	0.814	0.989
700	0.929	0.836	0.846	0.653
800	0.918	0.855	0.832	0.718

Table 2.4: Evaluation of the cluster hierarchy using RCV1 data

number of documents	196555
number of unique words	16133
average document length	167
number of classes	14138

Table 2.5: OHSUMED dataset (88-91)

2.6.2 OHSUMED (88-91)

The OHSUMED test collection is a set of 348,566 abstracts collected from 270 medical journals over a period of 5 years. Each abstract is annotated with MeSH (Medical Subject Heading) labels by human observers. This indicates the topic of the abstract. Unlike the RCV1 dataset, these documents are not in temporal order. Another property of this dataset is, being from a specific subject area, they contain words from a much smaller vocabulary. Due to the presence of human assigned MeSH keywords over such a large collection, this dataset provides us with an opportunity to evaluate our algorithm over a large dataset and against real topic labels.

Evaluating clusters

Experiment Setup We used the Ohsumed 88-91 dataset from the TREC-9 filtering track to evaluate our algorithm for the correctness of assignment of documents to the classes. We selected only those articles for which both the MeSH labels and the abstract text were present. There were 196,555 such articles. As with the RCV1 dataset most informative words in the dataset were selected using $\mathrm{cf} \times \log\left(\frac{N}{\mathrm{df}}\right)$ score of the words. We repeated the clustering exercise using 25 to 200 words at a step size of 25. To determine the number of different topics present in this dataset one can look at the unique MeSH labels present in the dataset. But, as there are tens of thousands of such labels present we used fixed number of clusters to evaluate (see Table 2.6) the algorithms.

Body Regions;A01
Abdomen;A01.047
Abdominal Cavity;A01.047.025
...

Figure 2.10: First three lines of MeSH labels file (filename: mtrees2005.bin)

Results and discussion The F-score results of the experiments are given in Table 2.6.

We can see from the table that Normal-CLASSIT is the most competitive when the vocabulary size is small *and* the number of clusters formed is large. For all other settings, i.e., when the size of the vocabulary used is larger or when the number of clusters formed is smaller, Katz-CLASSIT performs better. This shows that the Katz-CLASSIT algorithm is more robust as it performs well across a much larger range of parameter values.

Performances of both the algorithms suffer when we create more number of clusters, which makes sense because, there are fewer features based on which to distinguish between clusters.

Evaluating hierarchy

MeSH labels present in the OHSUMED collection has a hierarchical structure to it[6]. This provides us with another opportunity to evaluate the correctness of our hierarchy. This class hierarchy is much larger than the *topic* hierarchy of RCV1 dataset. There are 42610 different MeSH labels. Each MeSH label has a code attached to it. The class hierarchy information can be directly read from this code. For instance the first three records of 2005 "ASCII MeSH collection" reads

This says that the topic labeled "Abdominal Cavity" (*A01.047*.025) is a child topic of label with code A01.047, which we can find from the file as the topic "Abdomen" (*A01*.047), which in turn is a child topic of a topic with code A01. We can find from the file that this is the code of the label "Body Regions". This "." separated topic codes let us easily find the parent topics by dropping the suffix of the code. Not all the MeSH labels are seen in our dataset. There were only about 14138 different MeSH labels used in document set we used for our experiments.

Documents were pre-processed as described in the previous section. Entire cluster hierarchy was generated and the correctness of the hierarchy was evaluated as described in Section 2.5.2. The precision values are reported in table

Here again both the algorithms do reasonably well in grouping classes with common parents under the same cluster with Katz-CLASSIT seems to have an advantage over Normal-CLASSIT across all vocabulary sizes. But, we must be careful here not to interpret these precision values as closeness of the entire cluster hierarchy to the ex-

[6]The entire collection of MeSH labels can be downloaded from the web-site of National Institute of Health (http://www.nlm.nih.gov). We have used 2005 MeSH label collection for our purpose.

k→ V↓	5			10			20			40			80			160			320			640		
25	?	K	N	?	K	N	√	K	N	?	K	N	√	K	N	×	K	N	?	K	N	×	K	N
	μ	57	55	μ	57	53	μ	57	53	μ	57	53	μ	55	38	μ	55	61	μ	36	34	μ	27	34
	M	62	62	M	60	61	M	63	62	M	62	62	M	60	54	M	37	54	M	49	52	M	43	52
50	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	×	K	N	×	K	N
	μ	70	57	μ	70	57	μ	69	57	μ	69	57	μ	69	57	μ	69	57	μ	48	51	μ	47	51
	M	74	65	M	75	63	M	75	65	M	76	69	M	76	70	M	76	71	M	60	65	M	59	65
75	?	K	N	?	K	N	?	K	N	×	K	N	×	K	N	×	K	N	√	K	N	√	K	N
	μ	70	70	μ	70	70	μ	70	70	μ	69	70	μ	69	70	μ	69	*70*	μ	69	39	μ	69	35
	M	71	71	M	69	70	M	73	77	M	76	80	M	77	81	M	78	*82*	M	78	56	M	79	53
100	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N
	μ	70	62	μ	69	62	μ	69	62	μ	69	62	μ	68	62	μ	68	45	μ	69	45	μ	69	45
	M	72	69	M	71	70	M	75	73	M	78	75	M	78	76	M	79	62	M	80	62	M	80	62
125	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	×	K	N	√	K	N	√	K	N
	μ	71	61	μ	71	61	μ	69	61	μ	69	61	μ	69	61	μ	53	61	μ	53	47	μ	53	46
	M	74	68	M	76	68	M	77	71	M	78	72	M	80	74	M	68	74	M	69	61	M	69	60
150	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	×	K	N
	μ	72	54	μ	72	51	μ	59	51	μ	59	51	μ	55	51	μ	54	51	μ	54	51	μ	48	51
	M	72	65	M	77	61	M	72	64	M	74	66	M	71	66	M	71	67	M	71	67	M	66	67
175	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N
	μ	71	52	μ	71	51	μ	71	51	μ	**71**	51	μ	59	51	μ	58	43	μ	54	43	μ	54	41
	M	74	64	M	78	62	M	81	64	M	**83**	66	M	75	67	M	74	60	M	71	60	M	71	58
200	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N	√	K	N
	μ	62	52	μ	62	50	μ	62	50	μ	62	50	μ	62	50	μ	62	50	μ	62	50	μ	62	50
	M	72	63	M	75	62	M	77	65	M	78	65	M	79	66	M	79	67	M	79	67	M	79	67

Table 2.6: Cluster quality comparison on OHSUMED data at different number of clusters (k) and vocabulary size (V). The figures in the table are F-score×100. K stands for Katz-CLASSIT, N for the original CLASSIT. μ and M row in the smallest table hold the micro and macro average of the F-score respectively. The cells where Katz-CLASSIT performs better are marked with a $\surd$, the cells where Normal-CLASSIT performs better are marked with a $\times$ and the cells where there is no clear winner are marked with a ?. Best Katz-CLASSIT and the best Normal-CLASSIT have been highlighted by grey cells.

V	Normal Macro avg	Katz Macro avg	Normal Micro avg	Katz Micro avg
25	0.786	0.795	0.626	0.749
50	0.781	0.831	0.667	0.784
75	0.79	0.857	0.654	0.831
100	0.801	0.888	0.742	0.891
125	0.828	0.939	0.788	0.976
150	0.847	0.935	0.812	0.963
175	0.876	0.91	0.859	0.858
200	0.894	0.958	0.819	0.919

Table 2.7: Evaluation of the cluster hierarchy using OHSUMED data

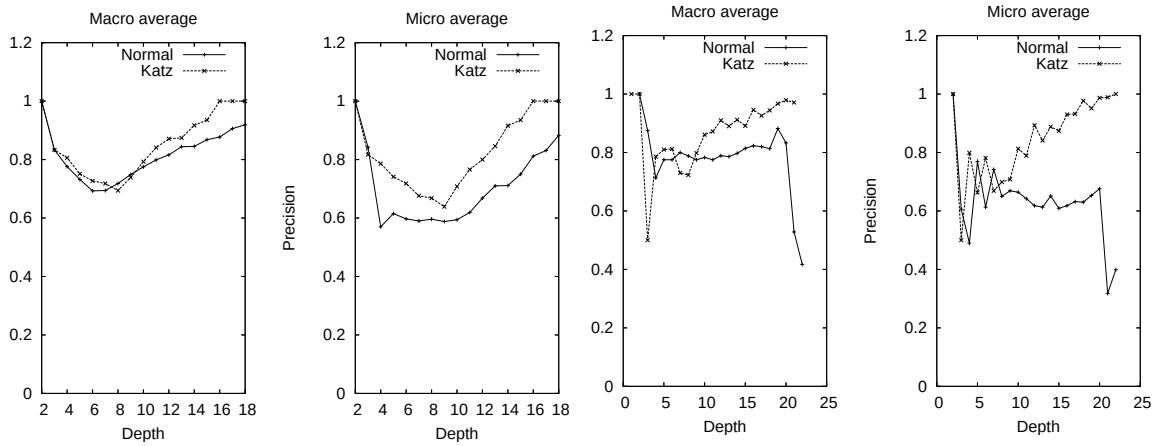


Figure 2.11: Tracing the cluster sibling precision over the height of the tree. Vocabulary sizes 25 and 75.

isting class hierarchy. Instead it is the accuracy of the algorithms in classifying sibling classes under same parent cluster.

We also tracked the sibling precision score at different depths of the generated cluster tree (Figure 2.11 to 2.12).

These plots show the general trend at different vocabulary sizes. As we can see there is considerable variation in the sibling precision over different depths. Amidst these variation we can observe that the sibling precision is higher and more consistent when we look at the nodes occuring at the lower layers of the tree. Also, we find that on these layers the Katz-CLASSIT usually performs better than the Normal-CLASSIT.

It is interesting to observe the general consistency at the lower levels of the tree and lack of it at higher levels of the tree. At the lower levels we have a large number of nodes at each layer. When we average the perfomance of each algorithm over these large number of nodes we get a score that is robust to random mistakes. So, we get a consistent score from layer to layer and it is easier to see which algorithm does better.

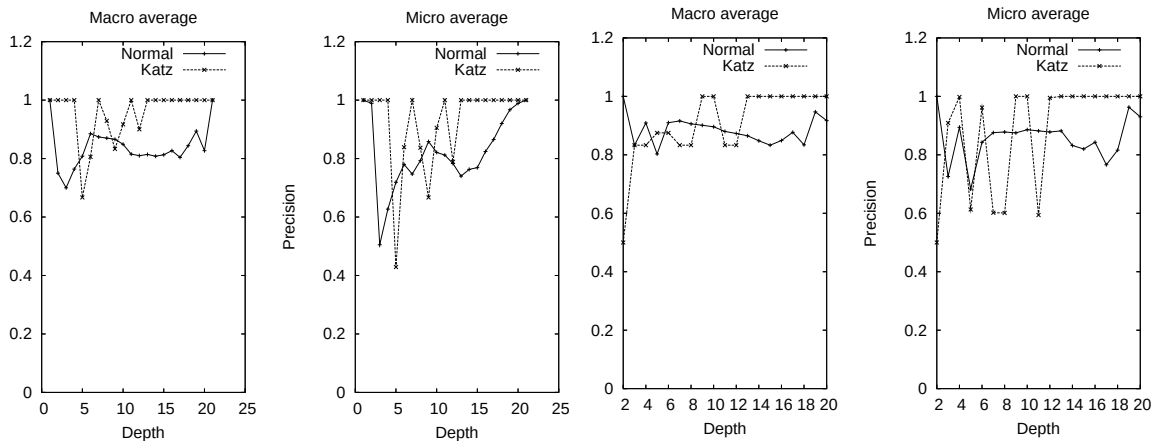


Figure 2.12: Tracing the cluster sibling precision over the height of the tree. Vocabulary sizes 125 and 175.

But, it is not so in the higher levels. In the higher levels we have only a few nodes in each layer over which to average the score. So, the average is more sensitive to random mistakes. Note that both the micro average and macro average are sensitive to these random mistakes. The wrong nodes in the higher levels of the tree either get a weight equal to other nodes (macro average) or they get a weight that is proportional to the number of *documents* in them. Both of these weights are significant at these levels of the tree. This is the reason why we find the plot of average sibling precision fluctuating a lot at these levels and we do not get a clear winner across the layers in the upper part of the tree.

2.7 Conclusion

This is the first attempt of incremental hierarchical clustering of text documents to our knowledge. We have evaluated an incremental hierarchical clustering algorithm, which is often used with non-text datasets, using text document datasets. We have also proposed a variation of the same that has more desirable properties when used for incremental hierarchical text clustering.

The variation of COBWEB/CLASSIT algorithm that we have demonstrated in this work uses Katz's distribution instead of Normal distribution used in the original formulation of the CLASSIT algorithm. Katz's distribution is more appropriate for the word occurrence data as has been shown in prior work(Katz 1996) and empirically observed in our work. We have evaluated both the algorithms over Reuters-RCV1 dataset, which allows us to carry out the experiments in a scenario very similar to the real life. We tested the algorithms by presenting them Newswire articles from Reuters-RCV1 dataset in time order and have shown that our algorithm performs consistently better than the Normal based CLASSIT algorithm as measured by both the micro and

macro average of the F score over a range vocabulary sizes. We have also evaluated both the algorithms using OHSUMED 88-91 dataset and have found that Katz-CLASSIT performs better except for the narrow range of parameter values with small vocabulary sizes *and* large number of clusters, where results are likely to be unreliable. This shows that the performance of Katz-CLASSIT is more robust across broad parameter settings.

We have also proposed a way to evaluate the quality of the hierarchy generated by the hierarchical clustering algorithms, by observing how often children clusters of a cluster get children classes of the class assigned to the cluster. We found that although, both the existing algorithm and our proposed algorithm perform well in this metric, our algorithm performs marginally better on OHSUMED dataset.

The most important contribution of this work is a separation of attribute distribution and its parameter estimation from the control structure of the CLASSIT algorithm. Thus, one can use a new attribute distribution, which may be different from Normal or Katz but is more appropriate for the data at hand, inside the well established control structure of the CLASSIT algorithm to carry out incremental hierarchical clustering of a new kind of data. For instance, if it is considered that Negative Binomial could be better fit for the word distribution than Katz distribution, and one can come up with an efficient way to estimate the parameters of the distribution, it can be used in the framework of the existing CLASSIT algorithm as demonstrated in this work. One can also experiment using a Bayesian approach to estimate the parameters of the distribution and carry out incremental hierarchical clustering in this framework, which might lead to better results due to more reliable parameter estimates for clusters with a small number of documents.

CHAPTER 3

On Multi-component Rating and Collaborative Filtering for Recommender Systems: The Case of Yahoo! Movies

Abstract

Collaborative filtering algorithms learn from the ratings of a group of users on a set of items to find recommendations for each user. Traditionally they have been designed to work with one dimensional ratings. With interest growing in recommending based on multiple aspects of items (Adomavicius and Kwon 2007, Adomavicius and Tuzhilin 2005) we present an algorithm for using multi-component rating data. This mixture model based algorithm uses the dependency structure between the rating components discovered by a structure learning algorithm and validated by the psychometric literature on the halo effect. This algorithm is compared with a set of model based and instance based one component rating collaborative filtering algorithms and their variations for multi-component rating collaborative filtering. We evaluate the algorithms using data from Yahoo! Movies. Use of multiple components leads to significant improvements in recommendations. However, we find that the choice of algorithm to use depends on the sparsity of the training data. It also depends on whether the task of the algorithm is to accurately predict ratings or retrieve relevant items. In our experiments model based multi-component rating algorithm was able to better retrieve items when training data is sparse. However, if the training data is not sparse, or if we are trying to predict the rating values more accurately then the instance based multi-component rating collaborative filtering algorithms perform better. Beyond generating recommendations we show that the proposed model can fill-in missing rating components. Theories in psychometric literature and the empirical evidence suggest that rating specific aspects of a subject is difficult. Hence, filling in the missing component values leads to the possibility of a rater support system to facilitate gathering of multi-component ratings. We also show that this allows recommendations to be generated for more users.

Item type recommended	Commercial	Non Commercial
Music	iTunes, Last.fm, Yahoo! Music	iRATEradio.com, mystrands.com
Movies	Netflix.com, blockbuster.com	movielens.umn.edu, filmaffinity.com
Books	StoryCode.com	gnooks.com
Dating	reciprodate.com	
Webpages		GiveALink.org, StumbleUpon.com
Aggregated	Amazon.com, half.ebay.com	

Table 3.1: Examples of Collaborative Filtering based recommender systems

3.1 Introduction

Recommender systems are increasingly used in online communities, e.g., shopping sites, subscription service sites, and online meeting places (see Table 3.1). The recommendations are generated from the collection of user preferences, yet they are personalized to each user. Recommender systems are especially useful when the user has too many choices to explore; they assist the users in discovering items that may appeal to them.

From the retailer's perspective, recommender systems may be used to target-advertise items to its customers. A merchant at an online marketplace can use a recommender system to induce demand for the less-known items in the system. By using its proprietary recommender system Netflix is able to effectively merchandise its collection of more than 100,000 movies. They are able to create demand for older, and often less-known, movies by advertising them to users who might like those movies. Given the constraint on the number of movies a subscriber can rent at a time, increase in demand for older movies reduces the demand for the newer releases which are more expensive to stock. As reported in the annual SEC filing of the company in 2006, the success of the Netflix business model depends, to certain extent, on effective use of, and user satisfaction in relation to, their recommender system (Netflix 2006). Online storefronts are not the only places where recommender systems can be used. There are communities of users with common interests who use recommender systems to find new items that they might enjoy. Some examples of such communities are Last.fm and Pandora.com (Internet radio stations with music recommender systems), StumbleUpon.com (a web page recommender system) and KindaKarma.com (a system to get recommendation on authors, games, movies and music). These developments suggest that recommender systems are important tools in mining collective user preferences to help users better navigate large choice spaces.

A key input to recommender system is the ratings given by the users on the items in the system. Ratings provide information about the quality of the item as well as about the taste of the user who gave the rating. Most recommender systems have been designed for single-valued ratings, i.e., for each (user, item) pair we have one rating indicating how much the user liked the item. However, sometimes there are multiple components to a rating. For instance, the popular Zagat survey (zagat.com)

rates restaurants on four criteria: food, decor, services and cost. Similarly, a movie could be rated for its plot, acting, visual effects, and direction. When such ratings are available from users, it is plausible that a recommender system could be designed that makes use of these component ratings and produces better recommendations for the users.

Contributions of this chapter

In this chapter we extend a successful collaborative filtering approach for generating recommendations from single component ratings to multi-component ratings. We do this by discovering dependency structure among multi-component rating data using Chow-Liu structure discovery algorithm. The discovered structure is validated by psychometric analysis of the multi-component rating data for halo effect. We embed this dependency structure in a flexible mixture model (FMM) (Si and Jin 2003). FMM has been shown to work better than the traditional mixture models for collaborative filtering. We evaluate a set of model based and instance based one component rating collaborative filtering algorithms and their extensions for multi-component rating dataset. The algorithms were tested using multi-component rating data collected from Yahoo Movies. The test results show a significant improvement from the use of multiple component ratings. We identify which multi-component rating algorithm performs better in which scenario and provide some insight into the behaviors of model based and instance based algorithms for collaborative filtering. We also show that the proposed model can be used to fill-in the missing rating components for incomplete records. This allows us to generate better recommendations for more users when there are incomplete ratings in the data. This also raises the possibility of a rater support system for helping the users in rating specific aspects of an item.

It is important to distinguish the current work from collaborative filtering in the presence of multi-dimensional *context information*, such as finding a recommendation for a movie to watch on a *Sunday* in the *evening* with *children*. Studies exist in the literature to incorporate multi-dimensional context information into the recommendation generation process. Adomavicius et al. (2005) suggests a reduction based approach where context specific recommendation is generated by using only the ratings collected in the context of interest. In the current work we address a different research question: how can we use the information in various components of a rating to make better recommendations for the users?

Background

Given a user's ratings on a subset of items and its peers' ratings on possibly different subsets of items, collaborative filtering algorithms predict which of the items the user would like among the items that he/she has not yet rated. Collaborative filtering algorithms recommend to each user, items that are popular among the group of users who are similar to him. This can be thought of as automating the spread of information through word-of-mouth (Shardanand and Maes 1995). Since, collaborative filtering

algorithms do not use the content information of the items, they are not limited to recommending only the items with content that the user has rated before.

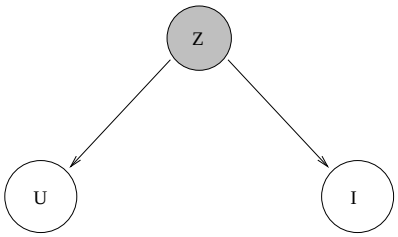
The first group of collaborative filtering algorithms were primarily instance based (Resnick et al. 1994). In the training step they build a database of user ratings that is used to find similar users and/or items while generating recommendations. These algorithms became popular because they are simple, intuitive, and sufficient for many small datasets. However, they do not scale to large datasets without further approximations. Also, because they do not learn any user model from the available preferences, they are of limited use as data mining tools (Hofmann 2004).

A second group of collaborative filtering algorithms, known as model-based algorithms, surfaced later (Breese et al. 1998, Chien and George 1999, Getoor and Sahami 1999). They compile the available user preferences into a compact statistical models from which the recommendations are generated. Notable model based collaborative filtering approaches include singular value decomposition to identify latent structure in ratings (Billsus and Pazzani 1998); probabilistic clustering and Bayesian networks (Breese et al. 1998, Chien and George 1999); repeated clustering (Ungar and Foster 1998); dependency networks (Heckerman et al. 2001); latent class models (Hofmann and Puzicha 1999) and latent semantic models (Hofmann 2004) to cluster the ratings; and flexible mixture models to separately cluster users and items (Si and Jin 2003). Unlike the instance based approach the model based algorithms are slow to train, but, once trained they can generate recommendations quickly.

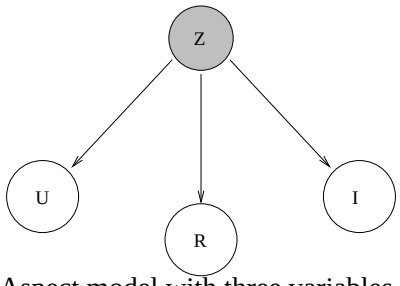
The model based algorithms are often described with the help of probabilistic graphical models. Probabilistic graphical models provide a framework based on probability theory and graph theory to approximate complex distributions (Pearl 2000). They graphically express conditional independencies among variables. The variables are represented as nodes and dependence among them is expressed as edges in a graph. The assumptions they encode about the distribution is: each node is independent of the non-descendent nodes conditional on its parent nodes. This allows one to use the chain rule of probability to factor the joint distribution over all the variables into the product of small conditional distributions. These smaller distributions can be individually estimated. This simplifies the operation on the joint distribution during training and inference (Koller and Friedman 2009). The latent class models presented in (Hofmann and Puzicha 1999) and the Flexible Mixture Model presented in (Si and Jin 2003) are given in Figure 3.1 and 3.2 respectively.

Product recommendation systems have been explored in marketing science as well. Often the goal is to predict the purchase outcome when the consumer is target advertised. Recently, Moon and Russel have developed an Autologistic recommendation model based on tools from the spatial statistics literature (Moon and Russell 2008). Their model uses the consumers purchase history to estimate the probability of a future purchase.

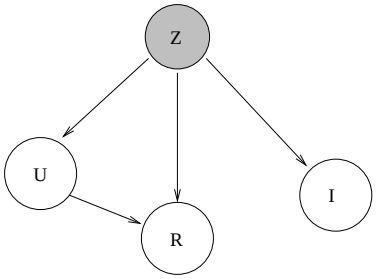
Most of the algorithms in the literature are designed to use unidimensional ratings. In a recent work Adomavicius and Kwon present approaches for multi-criteria rating collaborative filtering(Adomavicius and Kwon 2007). Their work is instance-based in



Aspect model with two variables.



Aspect model with three variables.



Rating depends on User and latent class.

Figure 3.1: Aspect model in Hofmann and Puzicha, 1999. Latent variable nodes are shaded and observed variable nodes are not shaded.

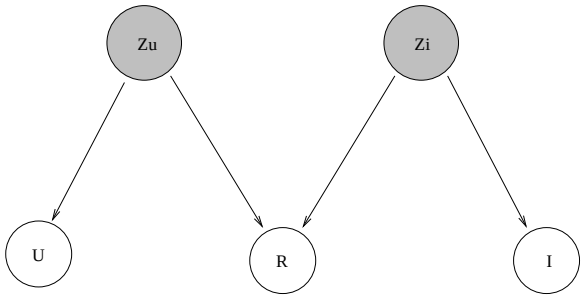


Figure 3.2: Flexible Mixture model of Luo Si, Rong Jin, 2003. Latent variable nodes are shaded and observed variable nodes are not shaded.

identifying similar users, but model based in aggregating component ratings into one overall rating. Lee and Teng use skyline queries to generate multi-criteria based recommendations from individual component rating predictions where each component rating is predicted using traditional collaborative filtering algorithms (Lee and Teng 2007).

The multi-component rating collaborative filtering has some apparent similarity with the conjoint analysis in marketing science (**?**). In conjoint analysis the objective is to estimate a consumer's preference function in terms of weight the consumer assigns to the attributes of a product. However, collaborative filtering is effective for experience goods for which attributes can not be readily determined. For instance directorial style of a movie is hard to express using an attribute-value system. It is worth noting that high rating for directorial style of a particular movie does not mean the user puts more weight on the directorial quality. Rather, it means that the user perceives a better match of the directorial style, not part of the data, with her preference. In this regard content based filtering strategies have more similarity with the conjoint analysis than do the collaborative filtering strategies.

The current work is built upon the Flexible Mixture Model (FMM) for collaborative filtering (Si and Jin 2003). FMM models the user and item distribution separately by using two latent variables. It has been shown to work better than other latent class models for collaborative filtering. We extend it for multi-component ratings taking into account specific properties of these type of data.

Multi-component rating data exhibits high correlation among the rating components. This is known as halo effect (Thorndike 1920, Wells 1907). Halo occurs in part due to the failure of the raters to evaluate each component independent of the others (Cooper 1981, Shweder 1975). The other reason often lies in the design of the questionnaire that collects multi-component ratings. If the definition of the components are ambiguous or not sufficiently different from each other the collected ratings are likely to be correlated (Kevin R. Murphy 1988). Although, there is some debate whether halo is entirely bad (Cooper 1981, Fisicaro 1988), it is generally considered undesirable because correlated components provide less information than independent ones. Halo can be reduced at the rating time by increasing the familiarity between rater and subject (Heneman 1974, Koltuv 1962, Landy and Farr 1980), reducing the time between observation and rating (E. F. Borgatta 1958, Shweder and D'Andrade 1980), clever questionnaire designs that makes the rater aware of the difference between the components (Rizzo and Frank 1977), and sensitizing the raters by training them to observe and avoid halo (Borman 1979, G. P. Latham 1980, Ivancevich 1979).

Some halo is almost always present despite the precautions. Holzbach suggests using a global component in the rating to collect each rater's overall impression of the subject and statistically remove its effect from each component rating (Holzbach 1978). Similar approaches are taken by Landy et al (Steele 1980) and by Myers (Myers 1965) to arrive at more accurate component ratings.

User	Movie	story	acting	visuals	direction	overall
u_1	m_1	4	1	2	1	2
u_2	m_1	2	1	1	4	2

Table 3.2: An example of multi-component rating. Ratings are on a scale of 0 to 4.

3.2 Multi-component rating recommender system

By rating multiple aspects of an item users provide more information about their preferences. The variation in different users' component ratings while they seemingly agree on their overall impression of the item can be informative. For instance, consider two users u_1 and u_2 who have given same overall ratings to the movie m_1 (Table 3.2). But, they differ in how they rate the components of the movie. The user u_1 likes the plot of the movie, while the user u_2 likes the direction in the movie. Without the component ratings we would have concluded that the users would not particularly like any movie similar to m_1. But, the component ratings tell us more. They suggest that the user u_1 might like other movies that have a story similar to m_1, while user u_2 might like a movie that has been directed by the same director or a director with similar style. Therefore, if we can effectively use the information in the component ratings we should be able to find more relevant items for the users. This would lead to higher precision and recall in the top K items we recommend the user.

Our empirical work has been motivated by the availability of extensive multi-component rating data from the Yahoo! Movies web-site. Although, the general approach taken in this work is applicable for any data with component ratings, for clarity we shall describe the methods of this work with the help of the Yahoo! dataset. A description of the dataset follows.

3.2.1 Data description and preliminary analysis

The rating data was collected from Yahoo movies website using a custom written program written in Java programming language.Each record of the rating data has seven variables: item or movie id (I), user id (U), ratings on story (S), acting (A), visuals (V), direction(D) and overall (O) quality of the movie. The ratings are on a thirteen point scale $(A+, A, A-, B+, B, B-, C+, C, C-, D+, D, D-, F)$. We recoded them to a scale of 0 to 4 ($\{A+, A, A-\} \longrightarrow 4, \{B+, B, B-\} \longrightarrow 3, \{C+, C, C-\} \longrightarrow 2, \{D+, D, D-\} \longrightarrow 1, \{F\} \longrightarrow 0$), so that there will be enough data points in each rating bucket. This is especially important for the conditional probability tables estimated in this chapter. The models are estimating the probability of observing certain rating values when a user and a item probabilistically belong to some latent classes. If there is not enough data points for a rating value, the probability estimates will be unreliable. Although, there were 691,496 (user, item) pairs in the original dataset, the user frequency in the data turns out to be skewed (Figure 3.3). Ratings from users who have rated very few movies are not useful for collaborative filtering, since, we

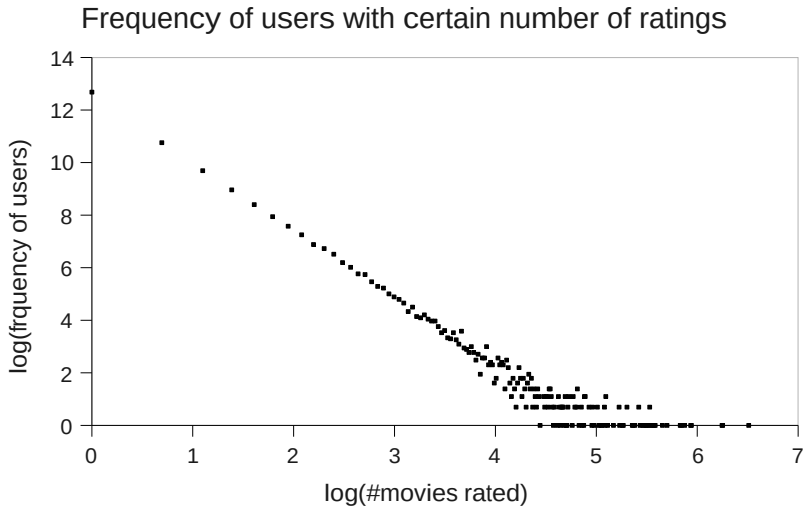


Figure 3.3: *log-log* plot of frequency of users who have rated a certain number of movies. *log*s are calculated with base e.

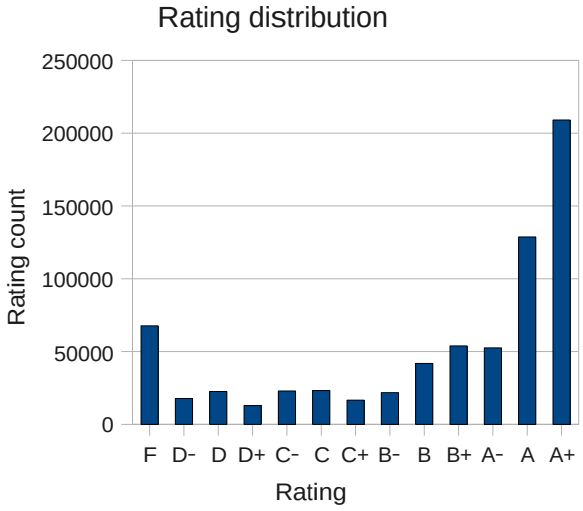


Figure 3.4: Movie rating distribution. Note the dip in the middle suggesting that people with strong opinions rate more often.

	S	A	D	V	O
S	1.00	0.79	0.82	0.74	0.87
A	0.79	1.00	0.81	0.73	0.83
D	0.82	0.81	1.00	0.79	0.88
V	0.74	0.73	0.79	1.00	0.80
O	0.87	0.83	0.88	0.80	1.00

Table 3.3: Correlation among components of rating

Component	One	Two	Three	Four	Five
% variance explained	84.5	5.7	4.4	3.2	2.2

Table 3.4: Principal components

can not reliably know the preferences of a user from only a few of his ratings. Also, we need enough ratings per individual to both train and test the model. Therefore, we have retained only those records that contain users who have at least 20 ratings. After this filtering there were 45,892 records, 1058 unique users and 3430 unique movies.

Examining the dataset for halo effect, we find that the components are highly correlated (Table 3.3). One way to detect halo is by Principal Component Analysis (PCA) and Factor Analysis (Morrison 1967). If most of the variance in the components can be explained by one principal component or one factor then it suggests the presence of halo (D. Kafry 1979). Principal Component Analysis of the ratings show that there is one component that explains 84.5% variance. Factor Analysis of the components produced a factor structure dominated by one factor (Table 3.5). These indicate that there is halo error in the collected ratings.

3.2.2 Modeling component ratings for collaborative filtering

The information contained in the multi-component rating has two parts: the overall component captures the overall impression of the user about the item and the variation among the components after partialling out the effect of the overall component tells us how the user evaluates aspects of the item. Traditionally, only the overall component has been used to carry out collaborative filtering, e.g., in (Si and Jin 2003). In

	Factor 1	Factor 2	Uniquenesses
S	0.91	0.23	0.11
A	0.87	−0.02	0.21
V	0.93	−0.08	0.10
D	0.84	−0.12	0.25
O	0.95	0.03	0.07

Table 3.5: Factor loadings after quartimax rotation

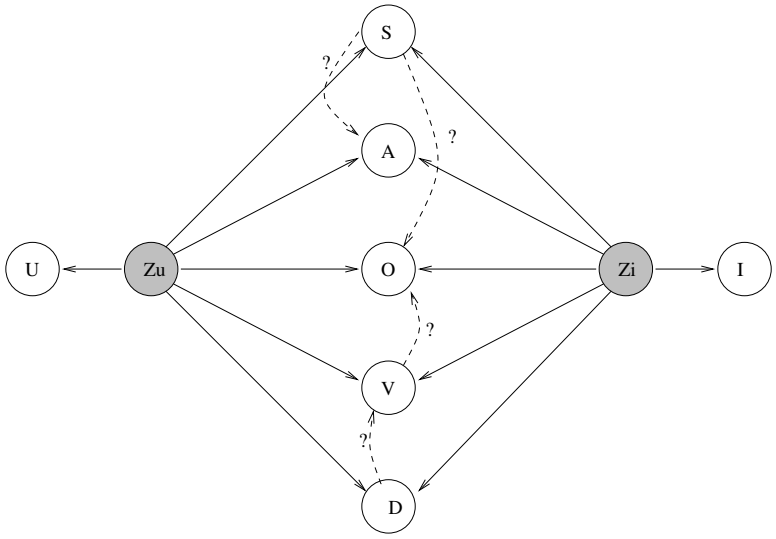


Figure 3.5: Flexible Mixture model for component rating collaborative filtering

this section we show how to use the additional information in components *along with* the overall component.

We use a mixture model similar to the Flexible Mixture Model (Si and Jin 2003) (Figure 3.2). FMM proposes that the rating an item receives from a user is governed by a small number of latent classes for the users and a small number of latent classes for the items. Given the latent classes, the rating is independent of the particular user and the particular item. We can start by embedding all the five rating components in the graphical model in place of the only overall component used in FMM. However, rating *components* are highly correlated as shown in Table 3.3. An incorrect independence among the component ratings in the model would mislead us to believe that each rating component provides completely new additional information about the user preferences.

Therefore, one would do well to find the dependency structure among the five components of the rating (Figure 3.5). This can be posed as a search problem where the goal is to find the dependency graph that maximizes the probability of the data. Just as during maximum likelihood estimation the parameter that maximizes the probability of data is deemed to be closest to the true parameter, the structure that maximizes the probability in this exercise is deemed the best approximation of the true dependency (Koller and Friedman 2009). To estimate the number of different subgraphs among the five components note that there are ten unique pairs, each of which can have an edge between them in either direction or have no edge at all. Discarding structures containing cycles we have 29,281 candidate structures to search through. This is time consuming. Another problem with a complete model search is that the configurations with multiple parents will lead to large conditional probability tables for which we do not have enough data to estimate the probabilities reliably.

Mutual Information

	S	A	V	D	O
S	–	0.61	0.72	0.51	0.88
A	0.61	–	0.68	0.47	0.73
V	0.72	0.68	–	0.64	0.92
D	0.51	0.47	0.64	–	0.66
O	0.88	0.73	0.92	0.66	–

MST $\Longrightarrow$

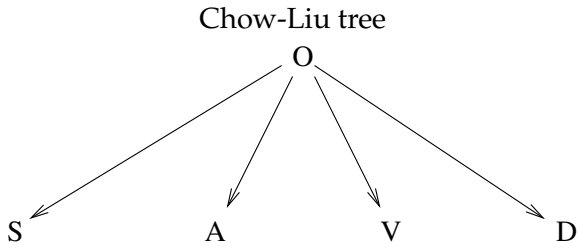


Figure 3.6: Discovered structure in the sub ratings

We strike a balance between having completely independent rating variables with no edge between them and accommodating fully connected rating variables that account for all possible dependence. We restrict ourselves to a category of structures that can capture much of the dependency among the rating variables while being amenable to fast and reliable parameter estimation. Chow and Liu have shown that if we have only discrete variables and we restrict ourselves to only those structures in which there is at most one parent node for each node, i.e., trees, we can efficiently search through them to find the tree that maximizes the probability of data. When the probability distribution is factored according to a tree dependency structure the graph that maximizes the log probability of the data is the one that maximizes the sum of pairwise mutual information[1] over each edge in the graph. Hence, the tree can be found by a maximum weight spanning tree algorithm (Chow and Liu 1968).

Such an exercise over the five component ratings leads to the structure shown in Figure 3.6. The structure states that the strongest of the dependencies among the components of the ratings is between the Overall rating and the components, as can be verified from the pairwise mutual information table in Figure 3.6. This shows the strong influence of a user's Overall impression of a movie on the perception of the other aspects of the movie. In the data we would find evidence of dependence between any pair of variables, but changing the parent for any of the components from O to any other variable (under the tree structure restriction one variable can have at most one parent) would lead to a lower probability of the data. Another way of reading this discovered structure is: given the Overall rating the components are independent. Note that this dependency structure says that if we do not condition on the Overall rating, then the S, A, D, V variables are dependent or correlated, which is consistent with the expectation we started with.

3.2.3 Parallels

It is interesting to compare the approach taken in the psychometric literature (Holzbach 1978, Myers 1965, Steele 1980) with the discovered Chow-Liu tree dependency structure among the movie rating components.

[1]Mutual information is a metric to measure the strength of the dependence between two variables(MacKay 2003)

$r_{R_iR_j.O}$	S	A	D	V
S	1.00	0.25	0.26	0.15
A	0.25	1.00	0.32	0.22
D	0.26	0.32	1.00	0.33
V	0.15	0.22	0.33	1.00

Table 3.6: Partial correlations controlling for Overall rating

Following a procedure like Holzbach's when we statistically remove the halo effect of the Overall component on the other components through partial correlation, the average inter-component correlation among variables S, A, D, V reduces from 0.78 to 0.26. As all correlations are positive some reduction in correlation is expected when computing partial correlations. However, the average partial correlation among the variables is the least when we control for the variable O among the possible five variables. The average partial correlations when we controlled for S, A, D, V are 0.47, 0.53, 0.35 and 0.60 respectively. These observations are in accordance with Holzbach's proposition that by controlling for Overall rating we can peer beyond the halo effect at more accurate component ratings.

The dependency structure given in Figure 3.6 says that if we condition on the Overall rating (O), then the components should be independent of each other. Strictly speaking, this assertion of the Chow-Liu tree is correct only if the assumption that the dependency among the rating components can be described by a tree structure is correct. However, a weaker assertion that states that among all possible variables that we might have conditioned on, conditioning on O leads to least residual dependency among the remaining components is still true. We found that the discovered structure persists over different random subsets of the data, suggesting that it is robust.

This result empirically validates the approach taken by (Holzbach 1978), (Myers 1965) and (Steele 1980) using a much larger dataset. It is interesting to note that the Chow-Liu tree structure discovery method, which is agnostic to the meaning of the rating components, arrives at a conclusion based on the empirical distribution of the data that agrees with the what researchers in psychometric literature arrived at based on the general impression theory of the halo effect. We believe this agreement adds to the validity of both the approaches.

3.2.4 Model estimation using EM algorithm

Using the discovered structure between the components of the ratings we construct the model shown in Figure 3.7 for collaborative filtering. As we have hidden variables the parameters need to be estimated using an indirect method. We propose an algorithm based on the EM framework (Dempster et al. 1977). The algorithms based on EM framework have two alternating steps that monotonically increase probability of the data or the likelihood of the parameters:

E (expectation) step where one computes the distribution of the unobserved variable

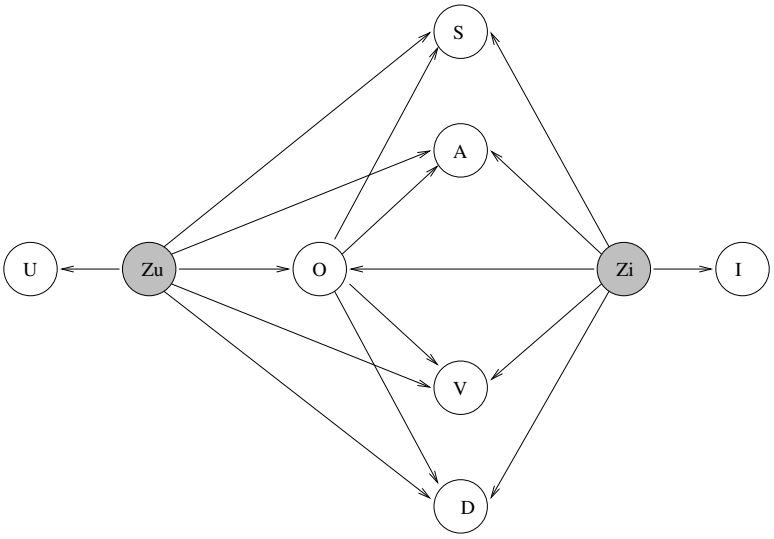


Figure 3.7: Flexible Mixture Model with component rating dependency structure

given all the observed variables. This is same as doing an inference on the graphical model for the hidden variables. It can be shown that among all distributions over the hidden variables the posterior distribution given the observed data maximizes the expected log-probability. Intuitively, the posterior distribution over the hidden variables given the observation is our best guess about the values of the hidden variables.

M (maximization) step where one estimates the parameters of the complete distribution (consists of observed and unobserved variables) using the standard maximum likelihood estimation. This operation maximizes the expected log probability given the posterior distribution of the unobserved variables.

The E and the M steps in our case are:

E-step

$$P(Z_u, Z_i|\vec{X}) = \frac{P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i, \mathrm{Pa}_{R_j}\right)}{\sum_{Z_u}\sum_{Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i, \mathrm{Pa}_{R_j}\right)} \quad (3.1)$$

M-step

$$P(Z_u) = \frac{1}{L}\sum_l \sum_{Z_i} P(Z_u, Z_i|\overrightarrow{X_{(l)}}) \tag{3.2}$$

$$P(Z_i) = \frac{1}{L}\sum_l \sum_{Z_u} P(Z_u, Z_i|\overrightarrow{X_{(l)}}) \tag{3.3}$$

$$P(U|Z_u) = \frac{\sum_{l:U_{(l)}=U} \sum_{Z_i} P(Z_u, Z_i|\overrightarrow{X_{(l)}})}{\sum_l \sum_{Z_i} P(Z_u, Z_i|X_l)} \tag{3.4}$$

$$P(I|Z_i) = \frac{\sum_{l:I_{(l)}=I} \sum_{Z_u} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)}{\sum_l \sum_{Z_u} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)} \tag{3.5}$$

$$P\left(R_j|Z_u, Z_i, \mathrm{Pa}_{R_j}\right) = \frac{\sum_{l:R_{j(l)}=R_j \,\&\, \mathrm{Pa}_{\mathrm{Rj}(l)}=\mathrm{Pa}_{\mathrm{Rj}}} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)}{\sum_{l:\mathrm{Pa}_{\mathrm{Rj}(l)}=\mathrm{Pa}_{\mathrm{Rj}}} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)} \tag{3.6}$$

where,

$Z_u =$ Latent class variable to cluster the users
$Z_i =$ Latent class variable to cluster the items
$R_j = j$th rating node. $R_j \in \{S, A, V, D, O\}$
$\mathrm{Pa}_{R_j} =$ parent rating node of R_j
$L =$ number of records in the dataset
$l =$ record index
$\overrightarrow{X_{(l)}} =$ record numbered l. It consists of observations for U, I, S, A, V, D, O
$U_{(l)} =$ variable U in the record numbered l
$I_{(l)} =$ variable I in the record numbered l
$R_{j(l)} =$ rating variable R_j in the record numbered l
$\mathrm{Pa}_{R_j(l)} =$ rating variable Pa_{R_j} in the record numbered l

The E-step shown above is the conditional distribution computed by dividing joint distribution of all variables, factored using the conditional independencies, by the joint distribution of only the observed variables, obtained by marginalizing out the hidden variables. The M-step in the EM algorithm estimates the MLE of the parameters using both the observed and the unobserved variables. If we could observe all variables, we could find the MLE of parameters of each conditional probability table by dividing the number of records with matching values for all the variables in the conditional probability table by the total number of records with matching values of the conditioning variables. But, we do not observe the hidden variables. Therefore, we have to use our best guess about their number of occurrences or their expected occurrence counts at a record given the observations of other variables in the same record. This is obtained from the posterior distribution of the hidden variable. Since this conditional distribution is multinomial the expected number of times a hidden variable takes a certain value in one record is same as the probability of the hidden variable taking that value given the value of the observed variables. All equations of

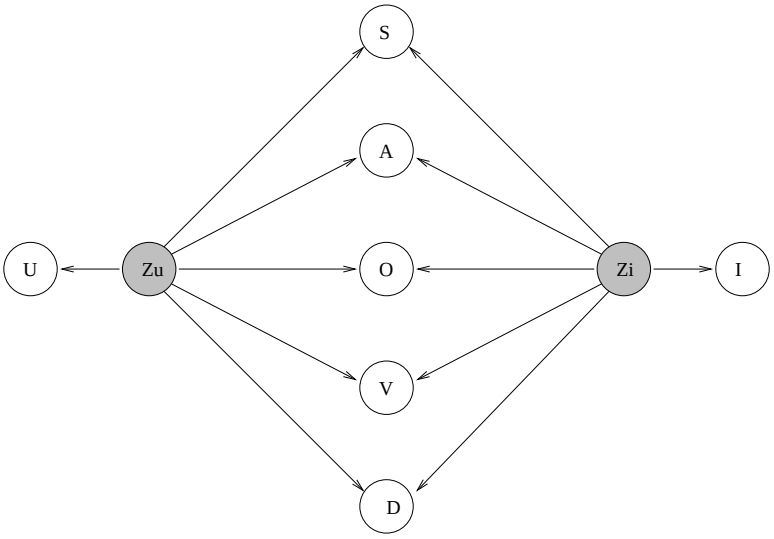


Figure 3.8: Flexible mixture model with independent component ratings

the M-step can be obtained by tabulating the values of the observed variables and, for the hidden variables, using the expected number of times the hidden variables take a certain value.

We compare this model with discovered dependency structure with the model that assumes independence among the component ratings conditional on the latent classes (Figure 3.8). The E and the M steps for the case when all components are assumed independent are derived in a similar manner.

E-step

$$P(Z_u, Z_i|\vec{X}) = \frac{P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i\right)}{\sum_{Z_u}\sum_{Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i\right)} \quad (3.7)$$

M-step

$$P(Z_u) = \frac{1}{L}\sum_l \sum_{Z_i} P(Z_u, Z_i|\overrightarrow{X_{(l)}}) \tag{3.8}$$

$$P(Z_i) = \frac{1}{L}\sum_l \sum_{Z_u} P(Z_u, Z_i|\overrightarrow{X_{(l)}}) \tag{3.9}$$

$$P(U|Z_u) = \frac{\sum_{l:U_{(l)}=U} \sum_{Z_i} P(Z_u, Z_i|\overrightarrow{X_{(l)}})}{\sum_l \sum_{Z_i} P(Z_u, Z_i|X_l)} \tag{3.10}$$

$$P(I|Z_i) = \frac{\sum_{l:I_{(l)}=I} \sum_{Z_u} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)}{\sum_l \sum_{Z_u} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)} \tag{3.11}$$

$$P\left(R_j|Z_u, Z_i\right) = \frac{\sum_{l:R_{j(l)}=R_j} P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)}{\sum_l P\left(Z_u, Z_i|\overrightarrow{X_{(l)}}\right)} \tag{3.12}$$

Note that the key difference between these two sets of expressions is the absence of any parent node in the conditioning part of the conditional probability of the component ratings (Expressions 3.6 and 3.12). The intuition behind these equation are similar to those described for the previous set.

We also compare these approaches with the baseline case where there is only one rating: the Overall rating on the movie. The E and the M steps can be borrowed from (Si and Jin 2003) or derived following the approach used to arrive at Equation 3.1–3.12.

E-step

$$P(Z_u, Z_i|U, I, O) = \frac{P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)P\left(O|Z_u, Z_i\right)}{\sum_{Z_u}\sum_{Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)P\left(O|Z_u, Z_i\right)} \tag{3.13}$$

M-step

$$P(Z_u) = \frac{1}{L}\sum_l \sum_{Z_i} P\left(Z_u, Z_i|U_{(l)}, I_{(l)}, O_{(l)}\right) \tag{3.14}$$

$$P(Z_i) = \frac{1}{L}\sum_l \sum_{Z_u} P\left(Z_u, Z_i|U_{(l)}, I_{(l)}, O_{(l)}\right) \tag{3.15}$$

$$P(U|Z_u) = \frac{\sum_{l:U_{(l)}=U} \sum_{Z_i} P\left(Z_u, Z_i|U_{(l)}, I_{(l)}, O_{(l)}\right)}{L \times P(Z_u)} \tag{3.16}$$

$$P(I|Z_i) = \frac{\sum_{l:I_{(l)}=I} \sum_{Z_u} P\left(Z_u, Z_i|U_{(l)}, I_{(l)}, O_{(l)}\right)}{L \times P(Z_i)} \tag{3.17}$$

$$P\left(O|Z_u, Z_i\right) = \frac{\sum_{l:O_{(l)}=O} \sum_{Z_u} P\left(Z_u, Z_i|U_{(l)}, I_{(l)}, O_{(l)}\right)}{\sum_l P\left(Z_u, Z_i|U_{(l)}, I_{(l)}, O_{(l)}\right)} \tag{3.18}$$

In each of these approaches the number of the classes (levels of Z_u and Z_i) is a user specified parameters. The more the number of classes the better will be the fit of the model to the data resulting in larger probability of data, but, that will increase the risk of overfitting the model to the training data. Another factor to keep in mind is that increasing the number of classes by n times leads to n^2 times more multiplications in the $E-step$ and n times more additions in the $M-step$. In our experiments the time taken to complete with higher levels of classes has been the bottleneck. We have experimented with 4, 8, 16 classes and found that the results improve with the number of classes. The time taken to complete the experiments were approximately 2.5 hours, 17 hours, and 92 hours respectively on a 3.0GHz pentium processor computer. In the reported results we are using 16 classes for each of the model based approaches.

Smoothing of parameter estimates using BDe prior(Koller and Friedman 2009)

To guard against over-fitting to the training data we smooth the parameter estimates in the M-step using a Dirichlet prior, which is a multivariate generalization of a Beta distribution and conjugate prior for multinomial distribution. A parameter $(\theta_1, \ldots, \theta_K)$ following a Dirichlet distribution with the hyper-parameters $(\alpha_1, \ldots, \alpha_K)$ has the probability distribution

$$P(\theta_1, \ldots, \theta_K) \sim \prod_k \theta_k^{\alpha_k - 1}$$

The expected value of $\theta_k = \frac{\alpha_k}{\sum_{k=1}^{k=K} \alpha_k}$. If we update this prior using multinomial data with counts $M_1, \ldots, M_K$, then we obtain the posterior that is another Dirichlet distribution with hyper parameters $(\alpha_1 + M_1, \ldots, \alpha_K + M_K)$. Thus, the expected values of the components can be obtained by adding the counts to the numerator and denominator of the original formula.

$$E(\theta_k) = \frac{M_k + \alpha_k}{(\sum_k M_k) + (\sum_k \alpha_k)} \tag{3.19}$$

Therefore the α's can be thought of as a pseudocounts and the sum of α's is a measure of the weight of the prior. Note that for each combination of values of the parent nodes there is a different set of parameters and priors. If the same Dirichlet prior is used for each conditional distribution the nodes with more parent configurations would have larger weights of the priors. BDe prior is a Dirichlet prior constructed so that the weight of the prior would be the same for each node: irrespective of how many parent configurations and, thus conditional probability tables, there are for each of them (Nir Friedman 1998). We use a BDe prior to smooth the parameter estimates during the model training phase.

3.2.5 Predicting the Overall rating

The goal is to predict the rating a user would give to an item he has not yet rated. Hence, the partial observation consists of the *user* and the *item*, and we are trying to predict the rating.

The joint distribution over all variables (observed and latent) is product of the conditional probability table estimated in Section 3.2.4 from which we need to marginalize away those variables that we are not interested in. In this section we focus on our ability to predict the overall rating. So, we need to marginalize all variables except U, I and O. For the three models discussed in the previous section the distribution over the variables U, I and O is:

$$P(U,I,O) = \sum_{Z_u} P(Z_u)P(U|Z_u) \sum_{Z_i} P\left(O|Z_u,Z_i\right) P(Z_i)P(I|Z_i) \tag{3.20}$$

Although the expression for all three models is the same the parameters estimated are different due to different conditional independence assumptions.

For user u_a and item i we are interested in the conditional distribution of the overall rating, namely,

$$P(O|u_a,i) = \frac{P(u_a,i,O)}{\sum_O P(u_a,i,O)} \tag{3.21}$$

The mode of this distribution of O is predicted as the output[2].

3.2.6 Instance based approaches

We compare the performance of these proposed algorithms with several of the existing one component and multi-component instance based methods. As a baseline we use the framework proposed by (Breese et al. 1998):

$$\widehat{R_{tj}} = \bar{R}_t + \frac{1}{\sum_i^{N_u} abs(sim(t,i))} \sum_i^{N_u} sim(t,i)(R_{ij} - \bar{R}_i) \tag{3.22}$$

Where, the expected rating a user t would give to item j is computed by ratings of other users weighted by similarity of those users to the target user t. One of several metrics can be used for computing the similarity between two users who are represented by the vectors of ratings they have given. Some choices are cosine, correlation, inverse eucledian distance etc. We use correlation coefficient for obtaining baseline results since it has been often used in the literature.

(Adomavicius and Kwon 2007) has generalized the similarity measure in Equation 3.22 to the case of ratings with multiple components. Their predicted rating can be summarized as

$$sim(t,i) = \frac{1}{1 + dist_u(t,i)} \tag{3.23}$$

$$dist_u(t,i) = \frac{1}{|U_t \cap U_i|} \sum_{l \in U_t \cap U_i} dist_r(R_{tl}, R_{il}) \tag{3.24}$$

Several distance metrics are explored for $dist_r$ between two multi-component ratings, such as :

[2]Use of expectation of O to predict did not change the conclusions of the experiments.

1. Manhattan distance
$$\sum_k |R_i(k) - R_j(k)|$$

2. Euclidean distance
$$\sqrt{\sum_k (R_i(k) - R_j(k))^2}$$

3. Chebyshev distance
$$max_k|R_i(k) - R_j(k)|$$

They found that Chebyshev distance based approach performs best among the distance based approaches considered. Therefore, we compare our results with the results obtained by using Chebyshev distance to compute user similarities.

Because of the correlation among the component ratings, it is worth considering if it would suffice to isolate the principal rating component through a principal component analysis and use that in a single component rating collaborative filtering algorithm. We obtain the principal component of the multi-component rating through a PCA rotation and compute the correlation between pairs of users based on their thus identified principal component ratings on movies. After computing the similarity between the users we use original ratings in Equation 3.22 to compute the predicted value of a user's rating on a movie.

The distance metrics proposed in (Adomavicius and Kwon 2007) for multi-component rating collaborative filtering do not take into account correlation between rating components. One multi-dimensional distance measure that takes into account correlation between dimensions is Mahalanobis distance (**?**). Mahalanobis distance between two random vectors $\vec{x}$ and $\vec{y}$, that are assumed to be drawn from one common distribution, is:

$$dist_{maha}(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})} \tag{3.25}$$

Where S is the covariance of the distribution. In one set of experiments we use Mahalanobis distance to compute recommendations. This is done by using Equation 3.25 in Equation 3.23 and 3.24 to compute recommendations. We compute the covariance matrix from the ratings used in the training data.

The set of collaborative filtering methods compared are summarized in Table 3.7.

3.3 Results and discussion

3.3.1 Experiments with Random Training Sample

To compare the effectiveness of the three models we use a fraction of the user ratings to train our models (training set) and use the remaining to test the prediction (test set). A certain fraction of each user's records were randomly selected to include in the training data to make sure that there is some training data for each user in the test

	Model based	Instance based
Multi-component	• 5 dependent subratings • 5 independent subratings	• Chebyshev • Mahalanobis • PCA
One component	Flexible Mixture Model	User-User correlation

Table 3.7: Set of metrics compared

set. For each user-item pair in the test data we predict their overall rating (O) using each model. We calculate the Mean Absolute Error (MAE) of the predictions with the help of the known ratings. We also evaluate the algorithms' ability to select the highest rated items. These two results need not be correlated (Herlocker et al. 2004). Appropriateness of each depends on the application environment.

In an application where the user is recommended the items along with the ratings they might assign to the items it is important to be able to predict the numerical values of these ratings accurately. One example of such application environment can be found in the movie recommender system of Netflix. The rating prediction accuracy can be measured using Mean Absolute Error of the predictions.

$$\text{MAE} = \frac{1}{L_{\text{test}}} \sum_{l=1}^{L_{\text{test}}} |o_l - \widehat{o_l}|$$

where, L_{test} = the number of records in the test data

o_l = the true rating

$\widehat{o_l}$ = the predicted rating

However, in many other applications the user is only presented the top few items he is likely to rate highly. The numerical values of the items are deemed of no interest. One example of such an application environment can be found at Amazon.com. Here, if the recommender system can identify the top few items for the user with little noise then it can be considered to have fulfilled the requirement. It does not matter if all the predicted ratings are biased up or down, as long as the predicted ratings order the items in the same way as the user would, i.e., assign a relatively higher rating for items that the user would rate A followed by lower ratings to items that the user would rate B and so on. This correctness of such ordering of items for users can be evaluated using Precision-Recall plots. To describe this briefly, let's assume for a moment that the user would be only interested in the items rated A. Consider the top-N item predictions. *Precision* is the fraction of the N items that the user would have given rating A. *Recall* is the fraction of items that the user would have rated A that are in the top-N. With increasing N more A-rated items are fetched, improving

recall. But, at the same time, more of those items that are rated less than A are also retrieved, damaging the *precision* score. A good recommender system should be able to retrieve much of the A-rated items while maintaining high precision. Hence, a plot of precision at 11 standard recall levels ($0\%, 10\%, 20\%, \ldots, 100\%$) is commonly used to compare the performance of different systems (Baeza-Yates and Ribeiro-Neto 1999).

Often times in retrieval and recommendation tasks the quality of only the top few recommendations is of interest, becasue, users typically don't look beyond the first few items recommended. We measure the quality of top items recommended using two metrics:

1. Mean precision of the top-5 items recommended
2. Mean reciprocal rank of the first relevant item recommended (**?**)

$$MRR = \frac{1}{\#users} \sum_{i}^{\#users} \frac{1}{rank_{firstrelevant}}$$

Mean reciprocal rank measures how soon a person gets a relevant recommendation from the algorithm.

Accuracy in rating prediction

The average MAE score using 30 different random train/test partitions are shown in Figure 3.9. As expected, the overall error for each model decreases with increasing amount of training data. However, we see that in this set of results using Chebyshev and Mahalanobis distance metrics in an instance based multi-component framework does best in predicting the rating values. Using the principal component after a PCA rotation of the 5 component ratings, does not do any better or worse than simply using the overall rating in a single component correlation based instance based approach.

However, one of the advantage of model based approaches is that after training the models, the rating prediction step is quick. Therefore, they are more suitable for online recommendation generations. If one is interested in using a model based recommender system we have the following insights. Naively extending the existing Flexible Mixture Model for collaborative filtering with component ratings without considering the existing correlation among the component ratings does not lead to any improvement in the prediction of overall ratings over using only one component. As discussed in Section 3.2.2, components of the ratings are correlated and assuming independence among them given latent class leads to over counting of evidence. When we capture the dependence among the rating components through the Overall rating and explicitly model for it, the prediction accuracy improves. Error-plots of the two model based algorithms, one using only the overall rating and the other using 5 components with dependency, show that there is an advantage of using multi-component ratings when the training set is small—up to about 40% of the available dataset for training in this case. But, when there is enough training data, using only the overall

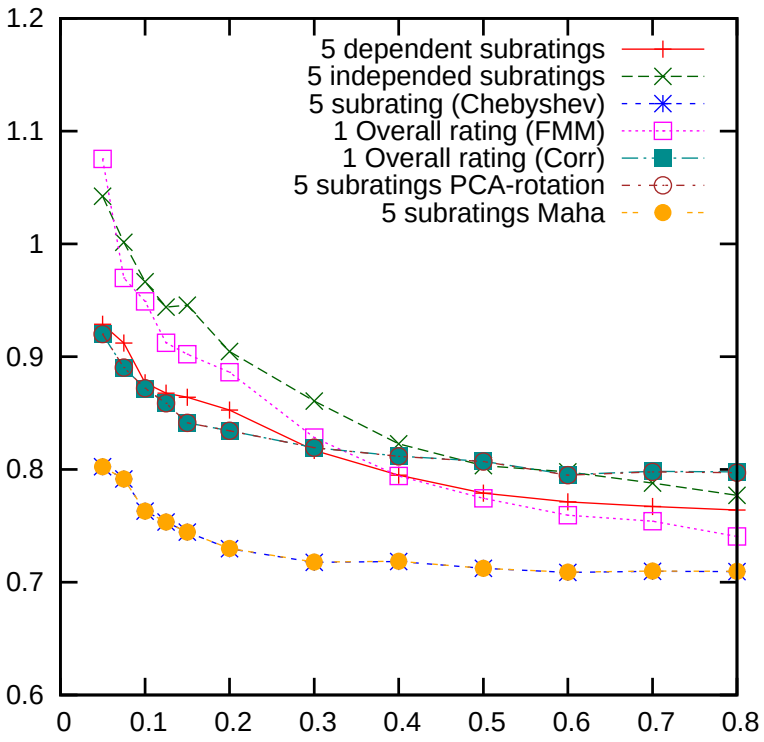


Figure 3.9: Plot of errors by amount of training data used, for different models.

rating leads to more accurate prediction of Overall rating. This suggests that when we have a user's Overall rating over a large number of items adding component ratings does not lead to any further improvement in the ability to predict the Overall rating the user might place on a new item.

To verify that the difference in the average MAE seen in Figure 3.9 are significant and not a result of chance, we performed a pairwise $t-\text{test}$ using MAE obtained at the 30 different train/test splits. We found that the differences are indeed significant except where the error lines in Figure 3.9 cross.

Accuracy in retrieving top-N items

The seven algorithms were trained as described in Section 3.2.4 at different training set sizes. Then the ratings for each user-item pair in the test set were computed using the prediction function of each method. This creates an ordering over the test item set for each user. A recommender system would recommend items from this list in decreasing order of predicted rating. The goal of the precision-recall curve is to find out how the precision of the recommendation is affected as more items are included from this list in the pursuit of retrieving all the relevant items. In this set of experiments we treat movies with rating 4 in the test set to be relevant. The precision vs recall curve is given in the Figure 3.10. A separate experiment that treats movies with rating 3 or higher in the test set to be relevant returns similar results albeit with a higher precision value at each recall level due to the presence of more relevant items.

When the training fraction is low the model with discovered structure gives the highest precision at each recall level. However, when we have more training data the instance based multi-component rating algorithms using Chebyshev or Mahalanobis distance do better. Instance based approaches using only the overall component or the principal component have the worst precision. The model with independence assumption among the component rating returns lowest precision among the model based approaches. However, as we use more training data the difference between the model based approaches diminishes. The interesting point to note here is that although when using only Overall as we use more training data we get a lower Mean Absolute Error than using all components in the model, it does not perform better in selecting top-N items. As pointed out in (Herlocker et al. 2004) these metrics measure two different aspects of the performance of the models and are often not correlated. One must use the appropriate evaluation metric to measure the suitability of an models for the task at hand.

We also compute the mean precision after top-5 retrievals and mean reciprocal ranks of each algorithm at these training fractions. The results shown in Figure 3.11 agree with the general observation made in the precision-recall curves in Figure 3.10. Among the model based approach the model with dependency structure performs best. However, as we use more training data instance based multi-component rating approaches that use Chebyshev or Mahalanobis distances out perform the model based approaches.

One possible reason for better performance of model based approaches over the

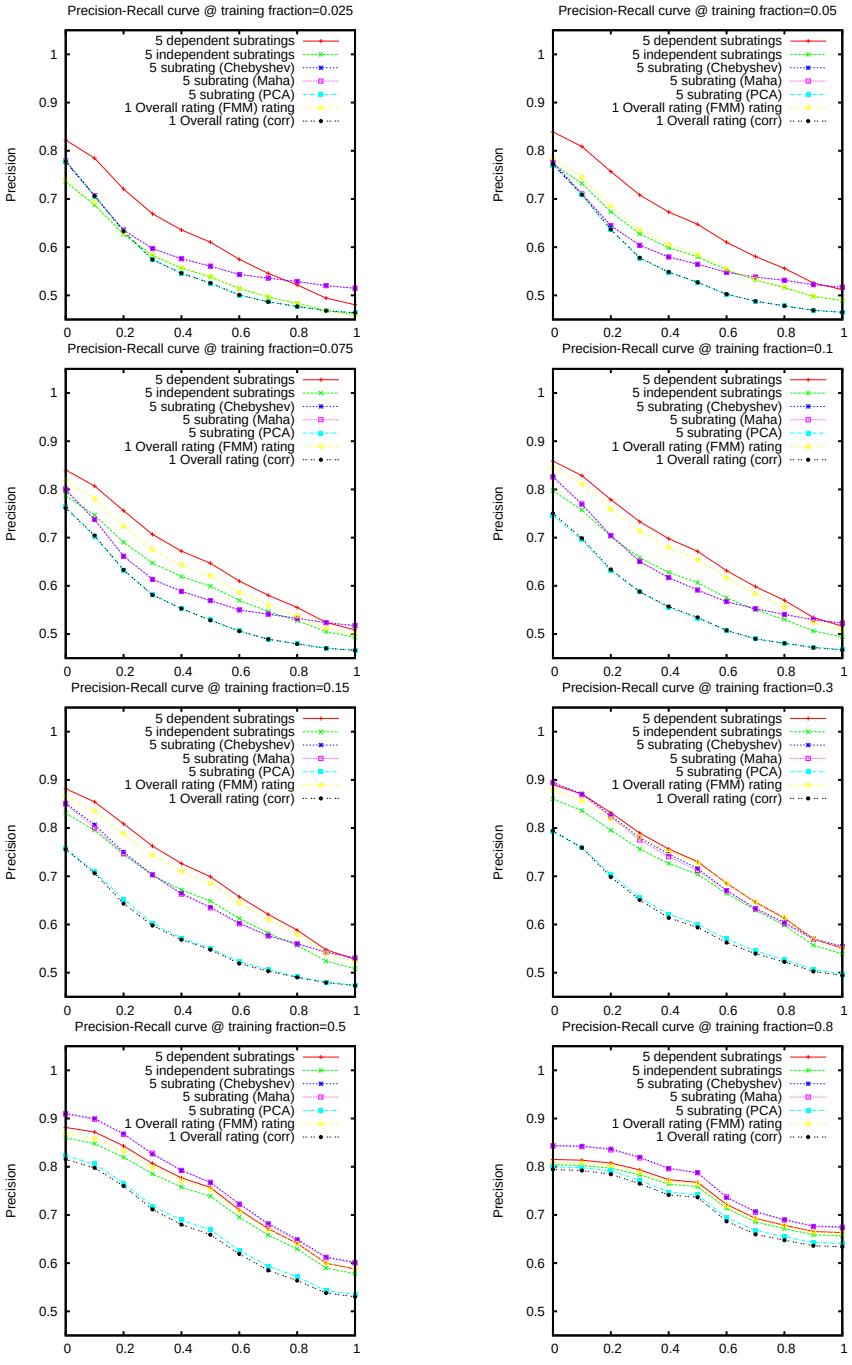


Figure 3.10: Precision-Recall curve for Three methods

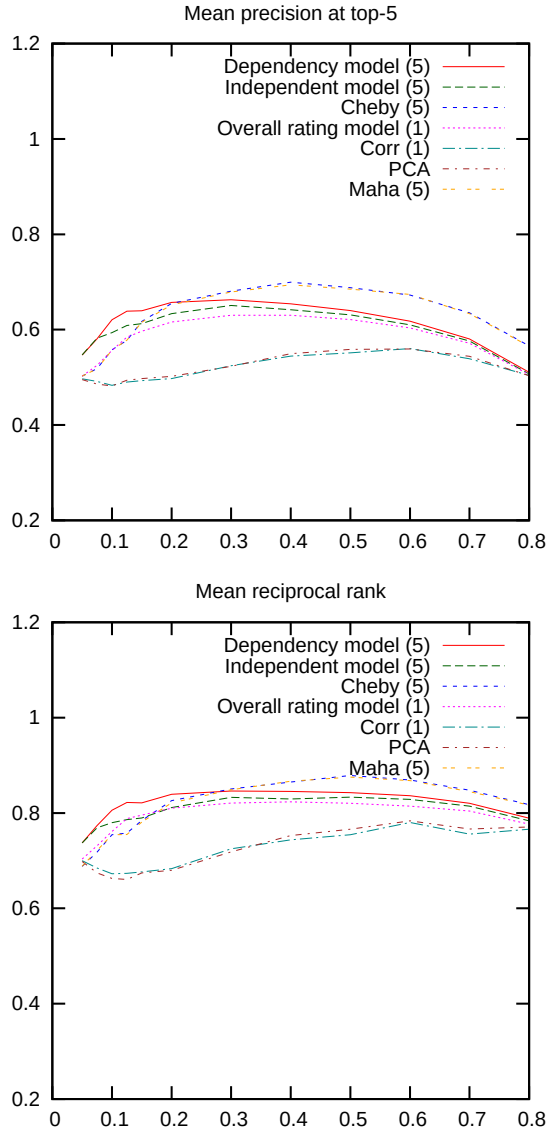


Figure 3.11: Mean precision at top-5 and Mean reciprocal rank

instance based approaches when the training fraction are low is that instance based approaches are based on pairwise similarity computations. When the training data is sparse the pairwise comparisons of users are unreliable. However, the model based approaches suffer less from these problems because each user is effectively compared with a model of a group of users that is less sparse. Thus the users are classified into the correct class more accurately. However, when there is more training data the pairwise user-to-user comparisons can be done more reliable. This leads to improved performance of the instance based multi-component approaches.

Hence, the takeaway from these tests is that we can improve the rating prediction accuracy by using multi-component rating, although, the right algorithm to use depends on the sparsity of the training data. It also depends on whether the recommender system is being used to predict the ratings accurately or retrieve the most relevant items quickly.

3.3.2 Experiments with time ordered data

In the previous section we used randomly selected subset of ratings for training and the remaining for testing. The advantage of using random partitions for training and testing is that by repeating this exercise multiple times we can evaluate the algorithms on the entire dataset. Moreover, the average result over multiple random train/test partitions is resistant to the peculiarity of any one partition. But, such data is not often available in real life. Ratings become available in time order and the only option is to use the past ratings to predict future ratings. To complicate matters, raters *might* be less consistent in their ratings when they start rating than they are after a while. So, using *past* ratings to predict *future* ratings may be harder than using random sample of ratings to predict remaining ratings. In the current set of experiments we address this question. We train our algorithms on the first few ratings collected from each person and test the predictions on her later ratings. Although, this does not measure the performance of the algorithm on the entire dataset, it simulates the real life scenario more closely than the random train/test split does.

Also, in the experiments described in Section 3.3.1 each user contributes a different number of ratings to the training set. So, it was hard to answer the question: until how many training points is there a benefit from using five components and after what point it is good to use only Overall? To answer this we use only the first 20 ratings of each user in this experiment. There were 22920 records, 2331 movies and 1146 users. After sorting the ratings by time order the algorithms were trained using first $1, 2, \ldots,$ 18 ratings for each user and tested using the remaining ratings. Comparing results from training data randomly selected with the results with training data selected by time order, we see that the random selection leads to lower MAE—as hypothesized (Figure 3.12).

The conclusion of the earlier comparison of the models is still valid in this more realistic test (Figure 3.13 and 3.14). When there are up to 5 training instances modeling for the dependency between all 5 components leads to lower MAE than using only overall rating. In the presence of 5-11 training instances they perform about equally

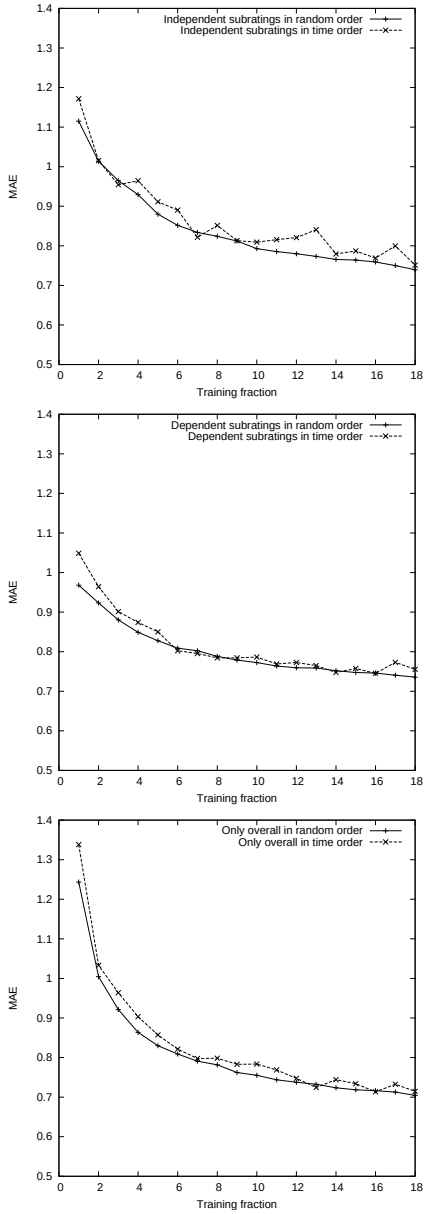


Figure 3.12: MAE comparison using randomly selected training data, vs., using training data as they are received.

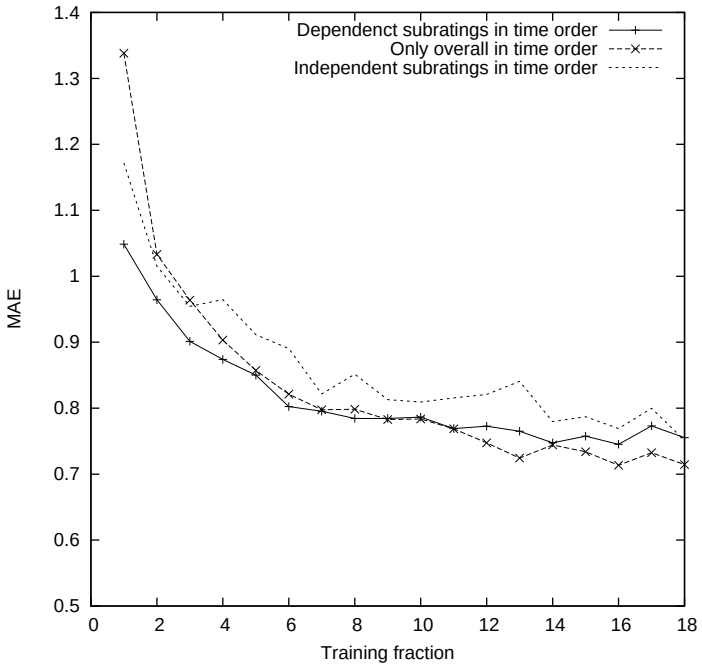


Figure 3.13: Three algorithms compared by using data in time order for training

well. When there are more than 11 training instances using only overall ratings leads to lower MAE. However, we find that the results are sensitive to the particular partition used. The MAE trend in this case is not as smooth as they are when the results with multiple random training and testing partitions are averaged. This variability is also observed in the precision and recall curves while top-N relevant items are retrieved (Figure 3.14). However, there is an advantage of using five dependent components as can be observed from the figure.

3.3.3 Filling-in missing component ratings

Raters find it easier to form an overall impression about their subject than to objectively evaluate specific aspects of it (Feeley 2002). This leads to two kinds of problems while collecting multi-component rating data:

1. **Halo Effect** If they choose to rate the components without deliberating over it enough to evaluate it objectively rating values get biased by their overall impression of the subject. This, known as the halo error, is treated in Section 3.1, 3.2.1, and 3.2.3.

2. **Missing Values** If the raters choose to skip rating the components, we have a missing data problem for rating components. In our dataset 34% of the records (235,659 out of 691,495) had incomplete rating information and thus needed to

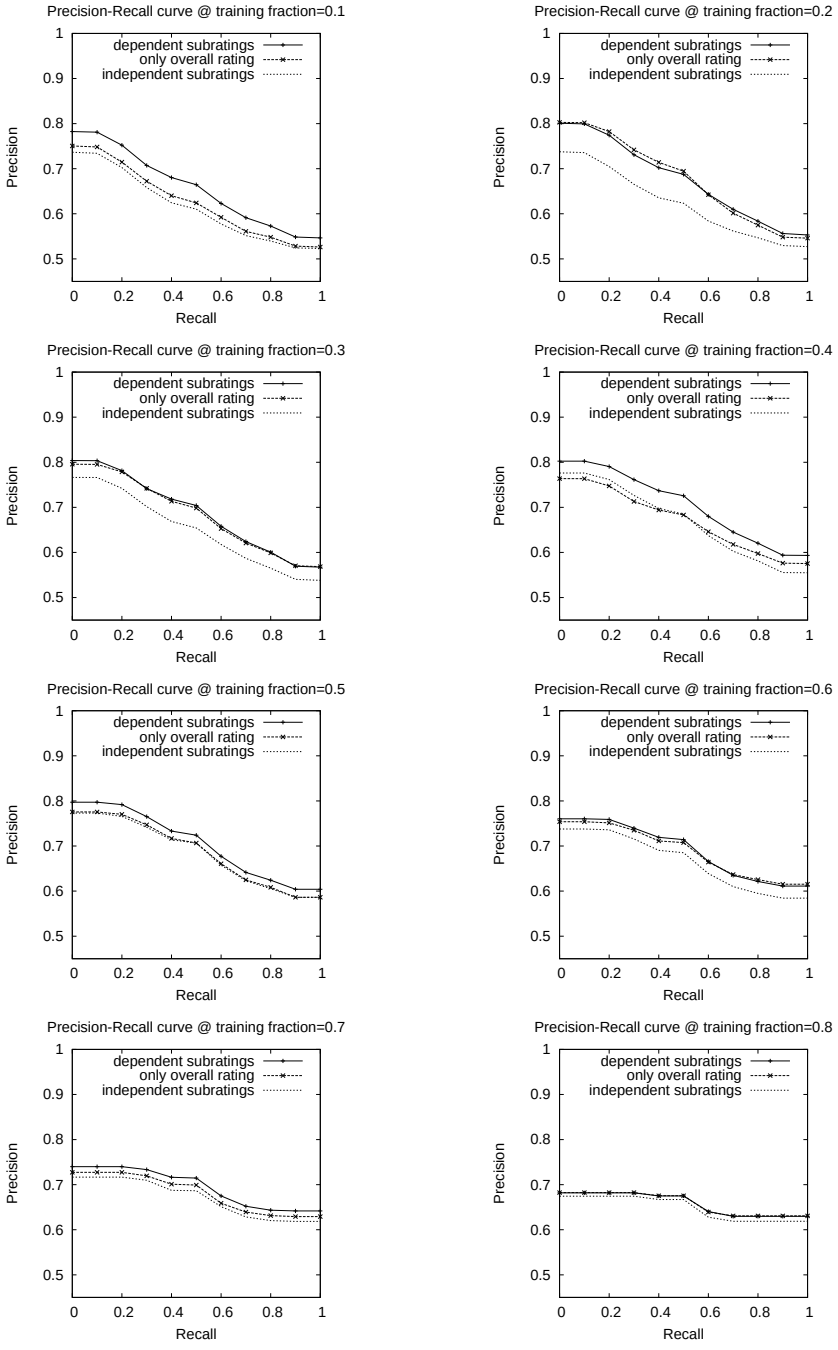


Figure 3.14: Precision-Recall curves using prior data for training and later data for testing

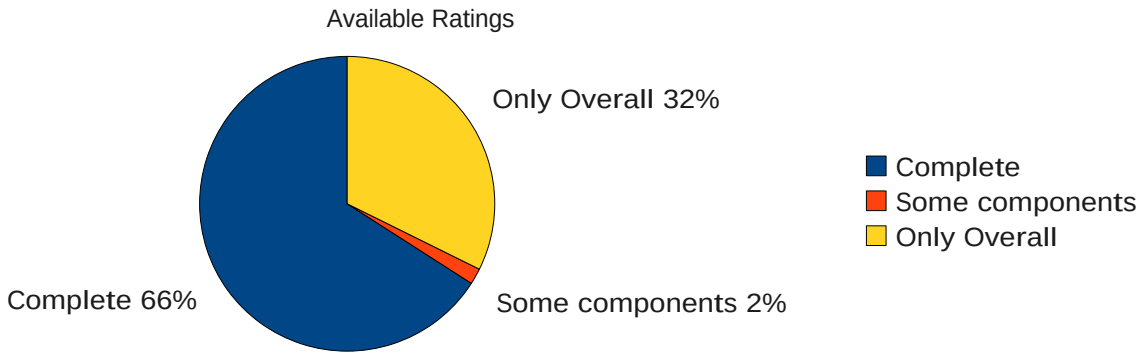


Figure 3.15: Records with partial information

	With unfilled components	Filled in components	% Increase
Number of User	1058	1680	59%
Number of Item	3430	3926	14%
Number of Records	45892	74110	61%

Table 3.8: Increase in dataset size after filling in missing components

be discarded for the experiments described in the previous sections. Of those 235,695 incomplete records 225,515 (95%) have only the Overall rating. This indicates the difficulty in obtaining component ratings from the users.

There are two opportunities for contribution here:

1. If we can predict the harder aspect ratings for a user for an items taking the user's Overall rating into account then we can design a rating support system. One use case is: the user gives his Overall rating on the item and the system pre-fills the component ratings. Then the user confirms them or modifies them if he feels they are different from how he would rate.

2. If we can fill in the missing values in the dataset we can generate recommendations for more users. Since we need a minimum number of ratings per user in the dataset, discarding incomplete records eliminates many users, and consequently many of their records even with complete ratings. Table 3.8 shows the difference between sizes of the dataset when we discard the incomplete records and when we fill-in the missing values using the method described in this section.

We showed in Section 3.2 that the probability distribution over all the variables can be factored as:

$$P(U, \vec{R}, I) = \sum_{Z_u, Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u) \prod_{j=1}^{5} P\left(R_j|Z_u, Z_i, \mathrm{Pa}_{R_j}\right) \quad (3.26)$$

Method	MAE
Multi-component FMM	0.353
SPSS EM	0.368
SPSS Regression	0.569
CF predicting Components	0.701

Table 3.9: Comparison of different methods filling in missing rating components.

Since, we always have Overall rating in our data we focus on predicting missing component ratings. To make an inference about one of the component ratings such as S using the values of U, I and O variables we need to carry out two operations on distribution given in Eq 3.26:

1. Marginalize away the variables we do not need, i.e., $R_j \in A, D, V$
2. Plug-in the values of the variable we have. Let's denote them as u, i and o.

The operations result in the following

$$\begin{aligned} P(U,I,S,O) &= \sum_{Z_u} P(Z_u) \sum_{Z_i} P(Z_i) P\left(O|Z_u,Z_i\right) P(I|Z_i) P(U|Z_u) P\left(S|Z_u,Z_i,O\right) \\ \Rightarrow P(u,i,S,o) &= \sum_{Z_u} P(Z_u) \sum_{Z_i} P(Z_i) P\left(o|Z_u,Z_i\right) P(i|Z_i) P(u|Z_u) P\left(S|Z_u,Z_i,o\right) \\ &\propto P(S|u,i,o) \end{aligned}$$

The result is a function of S that is proportional to its posterior distribution given the variable values u, i and o. The mode of this distribution is output as the predicted value of S.

Experiments and Results

First we use only complete records in this experiment to be able to predict the missing components and verify the predictions. The component ratings are hidden in the test data. Only U, I and O variable values were used from the test data to predict the hidden component ratings. We predicted each of the component ratings (S, A, V, D) for every record in the test set and computed Mean Absolute Error. 10-fold [3] cross validation was used to generate the training and testing samples (Mitchell 1997). We compared our results with the performance of the Missing Value Analysis (MVA) routines of SPSS. The Error values are given in Table 3.9.

MAEs in predicted missing values are close to 0.36 on a scale of length 5. When we predict the component ratings using only the U and I values as done with traditional collaborative filtering the MAE values are between $0.6 - 0.7$. This suggests that our method is able to extract considerable benefit from the additional available information in the Overall rating. The regression approach to predict missing values, part of

[3] Experiments with 2, 3 and 5 fold cross validations yield similar results

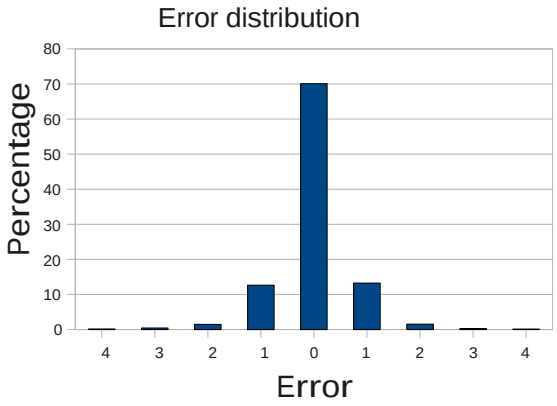


Figure 3.16: Error distributions

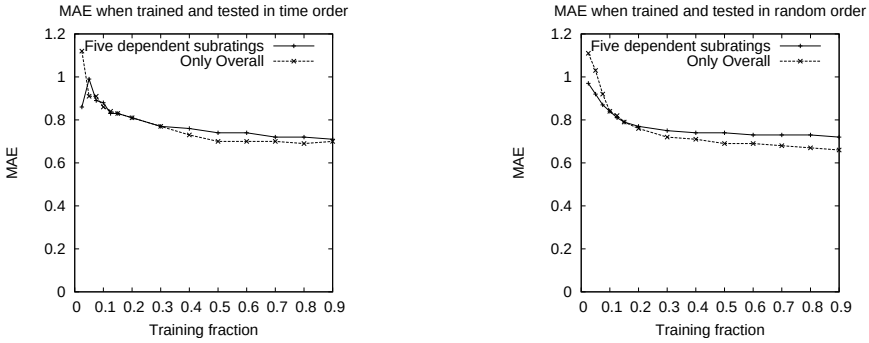


Figure 3.17: Accuracy of collaborative filtering after filling in missing components

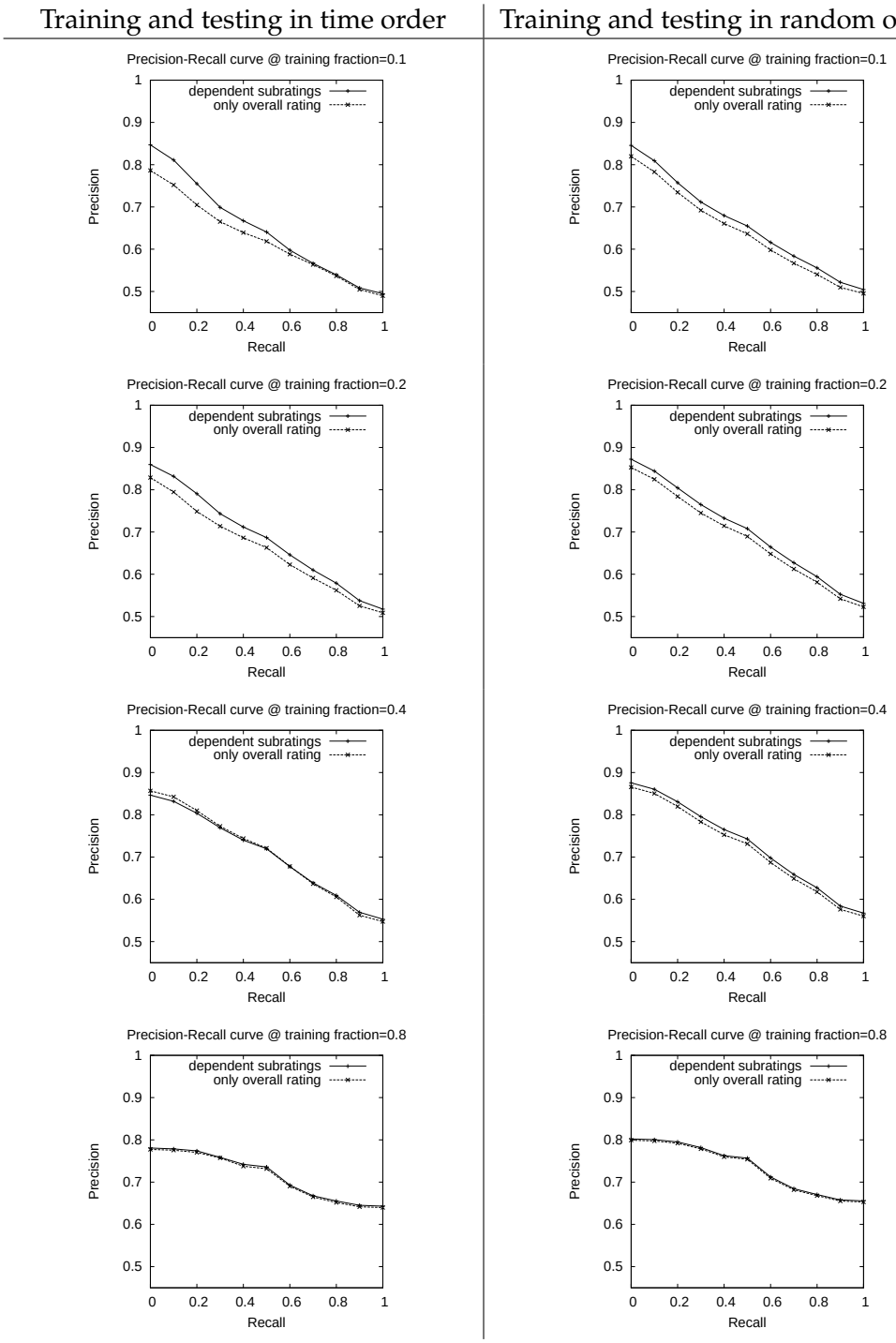


Figure 3.18: Precision recall curves for filled in data

the SPSS MVA module, was not very successful at an error of about 0.6. However, the EM algorithm used in the SPSS MVA module produced results almost as good as ours. The algorithm takes an iterative approach that alternates between the following two steps until convergence. Starting with random initialization of the missing values,

1. Regress each variable in turn against all other remaining variables and estimate the co-efficients,

2. Predict missing variable values of the incomplete records using linear regression model with the help of the co-efficients estimated so far and the other variables in the record.

Examining the error distribution of our method, we found that the errors are well behaved, with very few predictions off by a large margin (Figure 3.16). In about 70% of the cases we were accurate in our prediction of missing value and in about 96% of the cases the prediction was within one rating from the true value.

In the second part of the experiment, we used this method to fill in the missing component ratings in the dataset. We then used the completed dataset to generate recommendations. We found that our prediction accuracy is similar to what we get when we use only complete records (Figure 3.17 vs Figure 3.9). The advantage of using multiple components for collaborative filtering when small amount of training data is available is preserved as well (Figure 3.17, 3.18). Admittedly, it is less clear from the MAE scores when the data is used in time order, where, using only Overall ratings has an advantage when there is large amount of training instances.

To summarize: the proposed imputation method can extend the use of multi-component rating collaborative filtering algorithms to datasets consisting of records with missing rating components while preserving the algorithms' advantages over the single component rating collaborative filtering algorithm. It can also be used as a rater support system that uses a rater's Overall rating, that is easier to get, to make a knowledgeable prediction of the component ratings, that are harder to get.

3.4 Conclusions

We started this study with the following question:

> Can the recommendations by collaborative filtering algorithms be improved by using multiple component ratings?

To answer this question we collected multi-component movie rating data from Yahoo! Movies. Since component ratings are correlated due to halo effect a structure discovery exercise was carried out to find the dependency tree that captures most of the dependencies among the components. The discovered structure is interesting in itself. It says that the component ratings provided by the users are more correlated to the Overall ratings than they are to other component ratings. This suggests the possible relation between the Overall impression of the user and the ratings given

to the components. In this context, we draw a connection to the work on the halo effect in the psychometric literature. The body of work on halo effect indicates that component ratings are influenced by the presence of other strong factors and by the overall impression.

We develop a mixture model based collaborative filtering algorithm incorporating the discovered dependency structure. In addition several one component algorithms and their variations for multi-component ratings were evaluated on the collected dataset. The multi-component rating algorithms lead to better performance than the one component rating algorithms in both predicting the rating values accurately and in retrieving the most relevant movies quickly. The model based algorithm using dependency structure leads to better *retrieval* performance when the training data is sparse. However, when more training data is available, using instance based multi-component rating approaches, that use Chebyshev or Mahalanobis distance to measure distance between two ratings, perform better. In addition these two instance based multi-component rating algorithms are able to predict the ratings more accurately than other algorithms that we tested.

One of the advantages of the model based approaches is that after the model is calibrated, it can be used to quickly generate recommendations. This makes them suitable for scenarios like shopping web sites where real time recommendation generation is important. When the training data is sparse, a common problem faced by real world scenarios, there is an advantage of using multi-component ratings in a model that accounts for the Halo effect among the components. However, if we have more training data one component rating flexible mixture model is able to better predict the ratings than other model based approaches.

Another constraint of real life implementation of recommender systems is that we can only train our algorithms on past ratings to predict future ratings. Such prediction can be harder than using random partitions of the entire dataset for training and testing, because, raters could be inconsistent in their ratings early on. Although, each algorithm performs worse when ratings are used in time order, the advantage of multi-component rating–especially when using small amount of training data–is preserved.

The proposed multi-component model can be used to predict values of the missing component ratings. This is useful because in the current dataset approximately one third of the records have one or more of the component ratings missing. We show that the missing values can be filled in reliably. This allows us to generate recommendations for 59% more users and recommend 14% more items.

Multi-component rating collaborative filtering algorithms can suffer because of poor data. One can argue that objectively rating aspects of an item requires deliberation that only be expected from professional critiques. This can cause halo and reduce the information in the components. In this work we provide an approach to use the limited but important information in the components to make better recommendation. However, with increased emphasis on user generated content and on valuable services using them we expect the quality of such data to improve. The proposed

algorithm will be even more valuable in such a scenario.

Our work suggests several future research directions. One of foremost importance is the evaluation of the proposed method in other multi-component rating datasets. Also, in the presence of adequate number of ratings a more complete dependency graph among the ratings might be discovered and used as it will better capture the dependency among the rating components. We have shown that the proposed model can be used to fill in the missing rating components. Such an approach can be used to design a rater support system that predicts a user's component ratings using his overall rating. But, such a system might bias the rater. The implementation and evaluation of such a rater support system is an interesting topic to explore.

CHAPTER 4

Socio-temporal analysis of conversations in intra-organizational blogs

Abstract

Blogs have been popular on the Internet for a number of years and are becoming increasingly popular within the organizations as well. The analysis of blog posts is a useful way to understand the nature of expertise within the firm and identify opinion formation and the opinion leaders in organization. In this chapter, we are interested in understanding the *topics of conversations* that evolve through the blog posts and the replies to blog posts. While keywords within blog posts can be used to characterize the topic being discussed, the timestamps permits one to distinguish among the objects of the discussion, and the authors of posts provide a mean of separating different perspectives on the matter. Based on this observation we define *topics of conversation* using *keywords*, *people*, and *timestamps* of the blog posts and replies. We use tensors to capture these multiple modes of the blog data. We generalize the idea of *importance by association* that has been extensively used in social networks and other two-dimensional data analysis to multi-modal data. We show that such importances can be calculated by an operation of tensor factorization. This approach is illustrated by applying it to a dataset extracted from the blog network within a large globally distributed IT services provider over 30 months. We discuss implications of this work for monitoring opinion developments and detecting opinion leaders within the organization. Finally tensor factorization is applied to discover communities from the online social conversation and the effectiveness is measured with the help of author provided community labels on the conversations.

4.1 Introduction

Increasingly organizations are creating private blogs to promote peer-to-peer communication and collaboration among employees. With the advent of Enterprise 2.0 employees, who used to be end users of information, are playing a larger part in generating and disseminating knowledge with the help of blogs and user forums. The

activities in the blog network permit monitoring employee opinion, identify leaders or experts in different topics and enable an organization to develop a map of the expertise that is available within the organization. The automated analysis of large scale blog data to stay on top of happenings within the firm and gather organizational intelligence is a topic of considerable interest to managers and decision makers.

The Knowledge Management community has done a lot of work in developing Expert Finder systems to identify experts within the information system of an organization (Yimam 2000, Becerra-Fernandez 2006). The dominant theme of the expert finder systems is indexing expertise in a database and providing an interface for the manager to query it. However, it has grown to include user referrals (Kautz and Selman 1998) and personalized agents to identify experts (Vivacqua 1999). The data used in building such system include user response to expertise surveys, their posts in discussion groups(Krulwich et al. 1996), and the technical documents they produce (Streeter and Lochbaum 1988a,b).

On the other hand sociologists have been interested in identifying people with higher status, prestige, and influence in intra-organizational social networks (Brass 1992, Bonacich 1987, Bonacich and Lloyd 2001). They have proposed a number of centrality measures such as eigenvector centrality, betweenness, closeness etc. Usually, such measures are computed on networks with ties of one type that varies in intensity for different pairs of nodes, e.g., number of times a person seeks advice from another in an advice network, strength of friendship in a friendship network, etc. Only a few have looked at differing content of the ties in the network (Burt and Schøtt 1985).

Due to the adoption of online social media at the enterprise level employees are participating in creating and disseminating a wide variety of knowledge. Some of which could be of interest to organization, such as organizational practice, technical developments; where as others fall in domains outside of organizational interest, such as sports, politics etc. Therefore, to identify people who have high status in topics of particular interest to the organization as a whole or to groups of people within the organization, one needs to look into the content of the social exchange taking place in the online social network within the organization. This is the domain of text data mining literature.

The text data mining community has made considerable progress over last decade in analyzing and tracking topics in public text posts. The Topic Detection and Tracking initiative (Allan et al. 1998, Yang et al. 1998, Franz et al. 2001, Doddington et al. 2000), extension of the Latent Dirichlet Allocation (LDA) for temporal topic modelling (Wang and McCallum 2006), construction of patterns of statistically significant word occurrences (Swan and Jensen 2000) in news streams are important examples of work in this area. Another group of works have ventured beyond using just the word occurrence in text documents to include the author of the documents and the social environment in which they post. The author-recipient-topic model (ART) extends LDA to incorporate sender and recipient information for modeling email topics (McCallum et al. 2004). The content-community-time model is a two step probabilistic clustering approach for identifying time bound news topics in blogosphere (Qamra et al.

2006). The modeling of co-voting records by various senators is yet another example of socio-textual analysis for detecting groups of actors associated with certain topics (Wang et al. 2005).

Another stream of research has taken a matrix/tensor factorization approach to uncovering topics and trends in online social media. Eigen-trend is a method of tracking the importance of a keyword in the blogosphere taking into account weights of different blogs in which they are mentioned (Chi et al. 2006). They also propose a higher order singular value decomposition approach to compute hubs and authority scores of a set of blogs specific to a particular keyword. (Zhu et al. 2007) has proposed a technique to simultaneously factorize a web linkage matrix and web-page content matrix to generate a parsimonious representation of the original matrices. They have shown on WebKB and Cora dataset that using this parsimonious representation of one can perform page classification with accuracy as good or better than other state-of-the-art classification algorithms. TOPHITS is an approach to compute topical hub and authority in a set of web pages by representing the hyperlinks labeled by anchor text in a $from \times to \times keyword$ tensor and factorizing the tensor(Kolda and Bader 2006).

Dynamic modeling of relations in a social network has seen recent interest as well. (Chi et al. 2007) present a non-negative matrix factorization approach to model interaction between blogs over time as sum of interaction within communities of blogs whose intensities vary over time. A three way nonnegative tensor factorization approach has been applied in (Bader et al. 2007) for tracking relationship between employee from the Enron email dataset using the number of emails exchanged between them. They have also applied tensor factorization on international trade data for tracking trade relation between countries. A latent space model to track relation between authors in NIPS publication network has been presented in (Sarkar and Moore 2005).

Despite such progress in tracking social interaction and topic developments in news streams, topical analysis of social interaction over time has not been well explored. This could be of considerable interest for detecting and tracking significant topics of *conversation* between actors in a social network. Here we define conversation as set of messages exchanged between two or more people. Since, the significance of a topic of conversation depends not only on the content of the conversation, but, on the significance of the people participating in the conversation, we need an analysis framework that can handle actors at both side of the conversation, text content, and time stamp on the conversation. Author-recipient-topic model is based on one such framework, but, at the current stage it does not track topics or actors over time. The current chapter aims to fill this gap. However, we will follow a tensor based framework as opposed to probabilistic graphical modeling framework that ART is based on.

Objective

The objective of this chapter is to detect significant topics in conversations occurring in an online social network along with the important actors in those topics and important time periods when those topics developed.

The contributions of this work are:

1. A tensor based framework for representing the multi-modal data that is part of social conversation

2. An interpretation of tensor factorization that defines significance of entities in a multi-modal online conversation dataset

3. Proposal and evaluation of a tensor factorization approach for community discovery and tracking in blogosphere

4.2 Importance of entities in intra-organizational blogs

Importance by association is behind the calculation of many importance computations.

Eigen centrality has been used to defined status in a social network (Wasserman and Faust 1994). The centrality of an actor is determined by the centrality of other actors in the network that it is connected to. This follows the intuition that if a person is affiliated with other people who has high status in the network the person also has high status because of it. The adjacency matrix representation of such social networks are symmetric and its elements are positive. On such a matrix the centralities of the actors are given by the leading eigenvector. The leading eigenvector can be computed by singular value decomposition.

Singular value decomposition of an adjacency matrix ($from \times to$) of a network of directed hyperlinks between a set of web pages produces the hub and authority scores of the pages(Pagerank (Brin and Page 1998), HITS (Kleinberg 1999)). The leading left singular vector gives the hub scores whereas the leading right singular vector gives the authority scores. The node with high hub scores are the ones that link to nodes with high authority scores and the nodes with high authority scores are the ones linked to by nodes with high hub scores. Usually the leading singular vector pair is used since they explain most of the variations in the data, however subsequent singular vector pairs can also be used if their singular values indicate that they explain substantial portion of the data as well. Subsequent pairs have the same relation between the hubs and the authorities. Each pair corresponds to a different community sub-structure over the nodes. The first k pairs of singular vectors provide a decomposition of the two dimensional data matrix into k rank-1 matrices. This method is unsupervised: topics are determined solely from the co-occurrence patterns in the data. Although, HITS and Pagerank use SVD to identify authoritative pages in a set of hyperlinked webpages, they differ in their interpretation and in the adjacency matrices they operate on. HITS takes a more traditional network factorization approach to identify hubs *and* authorities. It focuses on a smaller sub-network identified by local crawling around a root web-page. However, Pagerank works on the entire WWW network. It views the authority of a webpage as the probability of a random surfer being at the webpage after a long time. The random surfers transition from one page to another can be modeled by a Markov process. The authorities of the web page is

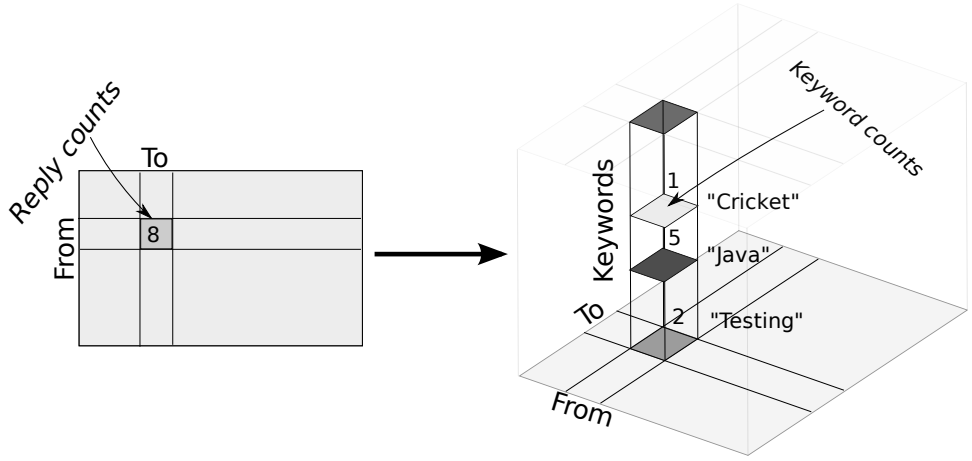


Figure 4.1: Tensor representation of the conversation

the stationary probabilities at the webpages. This is obtained by SVD on a transition probability matrix over the entire network.

Outside the network analysis literature SVD has been applied to the co-incidence matrix of documents and terms. This is known as Latent Semantic Indexing (LSI)(Deerwester et al. 1990). LSI produces a parsimonious representation of the original document-term matrix where each document has certain weight on a small number of semantic topics and each word has some weight on those semantic topic. In each topic, the weight of the document is determined by the weight of the keywords in the document and the weight of each keyword is determined by the weight of the documents in which it occurs.

Not all datasets can be satisfactorily represented by a two dimensional matrix. In a blog network where relations are indicated by citations and replies, encoding the relation by a single number would lose the content and the context of the relation. Or, in the case of an evolving network, where a timestamp is associated with each edge, a two dimensional representation of the relational data would have to be at the expense of temporal information. Such data is better represented and analyzed in a tensor. For example text messages exchanged between people in the social networks can be represented in a $from \times to \times keyword$ tensor (Figure 4.1).

Each cell of the tensor contains co-occurrence count, or a value derived thereof, of the three corresponding $author$, $recipient$, and $keyword$. The cell value indicates the strength of association between the three. Other similar examples can be found in TOPHITS (Kolda and Bader 2006) and three way DEDICOM (Bader et al. 2007). The current work builds on this literature using tensors for encoding semantic graphs and focuses on identification of significant themes along with important actors and important dates in each as part of the larger investigation into mapping expertise within an enterprise blog network.

4.2.1 Summary of notations

Here is a list of notations used in this chapter.

$a, b, c \ldots$ are used to represent scalars.

$\mathbf{a}, \mathbf{b}, \mathbf{c} \ldots$ are used to represent a vector.

$\mathbf{A}, \mathbf{B}, \mathbf{C} \ldots$ are used to represent two dimensional matrices.

$\mathbb{A}, \mathbb{B}, \mathbb{C} \ldots$ are used to represent tensors that have more than two dimensions or modes.

$\mathbf{a} \circ \mathbf{b}$ is the outer product between vector $\mathbf{a}$ and vector $\mathbf{b}$. The result is a matrix whose ith row and j th column contains $a_i b_j$. This can be extended to outer product between more than two vectors. Outer product of three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ would result in a tensor whose i, j, kth element contains $a_i b_j c_k$.

$\|\cdot\|_F$ represents the Frobenious norm of a matrix or a tensor. This is equal to square root of the sum of square of the elements in the matrix or the tensor.

$\times_k$ is the k mode multiplication of a tensor with a vector (Kolda and Bader 2008). It is defined as

$$\begin{aligned} \mathbb{Y} &= \mathbb{X} \times_k \mathbf{v} \\ \Leftrightarrow \mathbb{Y}_{i_1, i_2, \ldots, i_{k-1}, i_{k+1}, \ldots, i_M} &= \sum_{i_k} x_{i_1, i_2, \ldots, i_{k-1}, i_k, i_{k+1}, \ldots, i_M} \times v_{i_k} \end{aligned}$$

Notice that (1) the length of $\mathbf{v}$ must equal to the size of the kth mode of $\mathbb{X}$ and (2) $\mathbb{Y}$ has $M - 1$ modes. This is similar to multiplication of a matrix with a vector: multiplying dimensions of the matrix and the length of the vector must match; and the result is a vector of length equal to the non-multiplying side of the matrix.

The last one is the Kruskal operator $[\![\cdot]\!]$ defined by (Kolda 2006) as

$$[\![\boldsymbol{\lambda}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(M)}]\!] = \sum_{r=1}^{R} \lambda_r \times \mathbf{a}_\mathbf{r}^{(1)} \circ \cdots \circ \mathbf{a}_\mathbf{r}^{(M)}$$

Each $\mathbf{A}^{(k)}$ matrix has R columns. The matrices are of equal width, but, they need not have equal height. The Kruskal operator adds together R outer products of M vectors to produce is a M mode tensor. The length of the kth side of the tensor is equal to the height of $\mathbf{A}^{(k)}$.

4.2.2 Importance definition for multi-modal data

Definition 1. Generalizing *importance by association* to tensors we propose that the *importance* of an entity in a multi-modal dataset captured in a co-incidence tensor depends on the importance of the entities in the other modes it is co-incident with.

If we assume that each mode makes a multiplicative contribution of importance for each co-incidence, as is done in many matrix based importance calculations, we can formalize the definition in the following way.

For a co-incidence tensor $\mathbb{X}$ of M modes let the importance of entities along kth mode be $\mathbf{a^{(k)}}$. $\mathbf{a^{(k)}}$ satisfies the following condition

$$a_j^{(k)} = \sum_{i_1}\sum_{i_2}\cdots\sum_{i_{k-1}}\sum_{i_{k+1}}\cdots\sum_{i_M} x_{i_1,i_2,\ldots,i_{k-1},j,i_{k+1},\ldots,i_M} a_{i_1}^{(1)} a_{i_2}^{(2)} \ldots a_{i_{k-1}}^{(k-1)} a_{i_{k+1}}^{(k+1)} \ldots a_{i_M}^{(M)}; \forall k \tag{4.1}$$

Using tensor multiplication notation the Equation 4.1 can be compressed to

$$\mathbf{a^{(k)}} = \mathbb{X}\prod_{i\neq k} \times_i \mathbf{a^{(i)}} \tag{4.2}$$

Notice that after the sequence of multiplications $\mathbb{X}$ reduces to a vector of length equal to the kth side of the tensor $\mathbb{X}$. This gives the weight of the entities along kth mode of the tensor $\mathbb{X}$.

Applied iteratively for all k, $\mathbf{a^{(k)}}$s converge to minimize $\|\mathbb{X} - \mathbf{a^{(1)}} \circ \cdots \circ \mathbf{a^{(M)}}\|_F$ (De Lathauwer et al. 2000). In other words

$$\mathbb{X} \approx \mathbf{a^{(1)}} \circ \cdots \circ \mathbf{a^{(M)}} \tag{4.3}$$

or, $\mathbf{a^{(1)}} \circ \cdots \circ \mathbf{a^{(M)}}$ is the rank-1 approximation of $\mathbb{X}$. $\{\mathbf{a^{(1)}}, \ldots, \mathbf{a^{(M)}}\}$ is the most dominant factor in $\mathbb{X}$.

One can compute the best rank-R approximation of $\mathbb{X}$ by using parallel factorization of the tensor (Harshman 1970)—often abbreviated to PARAFAC. Denoting the kth mode vector of rth factor by $\mathbf{a_r^{(k)}}$, the rank-R approximation can be expressed as:

$$\mathbb{X} \approx \sum_{r=1}^{R} \lambda_r \times \mathbf{a_r^{(1)}} \circ \cdots \circ \mathbf{a_r^{(M)}} = [\![\boldsymbol{\lambda}; \mathbf{A^{(1)}}, \ldots, \mathbf{A^{(M)}}]\!] \tag{4.4}$$

λ_r is a normalizer to keep norm of each weight vector $\mathbf{a_r^{(k)}}$ equal to 1. Each of R sets of $\{\mathbf{a^{(1)}}, \ldots, \mathbf{a^{(M)}}\}$ importance weights satisfy 4.2.

The popular approach to compute PARAFAC is based on Alternating-Least-Square error minimization (ALS). The error $\|\mathbb{X} - [\![\boldsymbol{\lambda}; \mathbf{A^{(1)}}, \ldots, \mathbf{A^{(M)}}]\!]\|_F$ is minimized by successively optimizing one of the M matrices while keeping the remaining $M-1$ matrices constant. The detailed ALS algorithm can be found in (Kolda and Bader 2008). An implementation is available in their TensorToolbox matlab package (Bader and Kolda 2007).

We illustrate two applications of this method for blog data analysis next.

4.2.3 Blog post developments

One view of the blogs is that they are self-publication media where bloggers write on topics of their interest. If the goal is to identify important developments of a topic in the posts, we posit that we need to look beyond the word occurrences in the blog posts. We also need to consider the importance of the author of the post. The post made by an authority in a subject is a stronger signal of a development in the topic,

than a post that is made by someone who is not an authority in the subject. Therefore, to identify different topic developments in blog posts, the relevant variables are the authors, timestamps and keywords of the blog posts. This data can be represented as a $author \times keyword \times timestamp$ co-incidence tensor $\mathbb{X}$, where, each cell of the tensor contains $tf - idf$ weighted and length normalized counts of the word occurrences. This value indicates the strength of association of the three variables. Spelling out Definition 1 for $author \times keyword \times timestamp$ tensor we obtain the following reinforcing definition of *authority* of bloggers, *importance* of keywords and *intensity* of a topic at a given time period for a particular topic:

1. The *authority* of a blogger in a topic can be judged from her participation during the period when the intensity of the topic is high and from her use of important keywords.

2. The *importance* of a keyword in a topic can be judged from its use by the authorities in the topic and from its use during the period when the intensity of the topics is high.

3. The *intensity* of a topic during a time period can be measured from the number of posts made by authorities and the presence of important keywords in the topic.

This is a higher order extension of hub and authority computation. When we want to identify only the most dominant topic, the importance of pth blogger in this topic can be calculated according to the definition as:

$$a_p = \sum_q \sum_r x_{pqr} k_q t_r \iff \mathbf{a} = \mathbb{X} \times_2 \mathbf{k} \times_3 \mathbf{t} \tag{4.5}$$

Similarly

$$\mathbf{k} = \mathbb{X} \times_1 \mathbf{a} \times_3 \mathbf{t} \tag{4.6}$$

$$\mathbf{t} = \mathbb{X} \times_1 \mathbf{a} \times_2 \mathbf{k} \tag{4.7}$$

where, $\mathbb{X} \in \Re^{|\mathbf{a}| \times |\mathbf{k}| \times |\mathbf{t}|}$; $\mathbf{a}$, $\mathbf{k}$, and $\mathbf{t}$ are the vectors of importance of the authors, keywords and time periods; $\times_j$ is the j-mode product of a vector with a tensor. Applied iteratively the vectors $\mathbf{a}$, $\mathbf{k}$, and $\mathbf{t}$ converge to minimize the error $\|\mathbb{X} - \mathbf{a} \circ \mathbf{k} \circ \mathbf{t}\|_F$. Thus $\mathbf{a} \circ \mathbf{k} \circ \mathbf{t}$ is the rank-1 approximation of the tensor $\mathbb{X}$. Extending from one dominant topic to R topics and using a set of normalizers $\boldsymbol{\lambda}$ to make each vector of unit length, the approximation can be expressed as sum of R rank-1 tensors:

$$\mathbb{X} \approx \sum_r^R \lambda_r \times \mathbf{a_r} \circ \mathbf{k_r} \circ \mathbf{t_r} = [\boldsymbol{\lambda}; \mathbf{A}, \mathbf{K}, \mathbf{T}] \tag{4.8}$$

where, $\mathbf{A}$,$\mathbf{K}$,and $\mathbf{T}$ are the three modal matrices each with R $\mathbf{a_r}$, $\mathbf{k_r}$, and $\mathbf{t_r}$ as column vectors respectively.

Blog post and reply data	
Bloggers	4.8K
Commentors	16K
Blogs	4.7K
Blog posts	71.5K average length 300 words
Comments	286K average length 33 words
Date range	Apr '06–Oct '08

Table 4.1: Blog data description

4.2.4 Blog conversation development

In this extension we take into account the conversational nature of the blog posts. A comment to a blog post or a post with a citation has an author and a recipient. Subject of the post not only depends on who is making the post but also who it is targeted to. To capture this we represent the blog data in a fourth order tensor ($author \times recipient \times kewords \times timestamp$). The idea behind evaluating the importance of a variable is similar to that in blog topic development analysis. The extension is that the importance of the recipient of the conversation influences the importance of the variables in other modes.

4.2.5 Comparison with the existing methods

The HITS algorithm (Kleinberg 1999) separates a blog network into multiple layers of network. But, it does so based on the pattern of links—not taking into account the content. One could envision an approach where first the blog posts are clustered into topics and then HITS is performed in each to find important people in the group. Although, this approach separates conversations into topics based on the content of the documents, it does not take into account the importance of the words said in identifying the important bloggers. This Blog conversation development work has more similarities with the TOPHITS (Kolda and Bader 2006) where a $from \times to \times term$ tensor was constructed for hyperlinked web pages. TOPHITS uses the anchor text as the description of the link. We use text in the blog posts and replies that are much longer; and require more cleanup and normalization of the term vectors. Our work is also different in its extension with a time dimension to track topic intensities over time.

4.3 Dataset

The data for this study is extracted from an employee-only blog network in a large IT services firm. It contains the blog posts and replies along with timestamps and demographic information about the bloggers. A subset of the blog articles are posted in one of the 25 communities by their authors. These 25 communities have been further classified as work related and non-work related by experts at the firm(Table 4.3).

Total # of posts 71.5K			
In community			Outside
44K			27.7K
	Work	Non-Work	
# posts	15K	29K	
# communities	11	14	

Table 4.2: Posts in different communities

Non work	Work
Photography	Senior Management
CorporateSocialResponsibility	Testing
Fun	KM
Puzzles	Feedback
Sports	FLOSS
Poetry-Stories	BusinessDevelopment
Education-Motivation	CorporateFunctions
Geographies	Project Management
Movie-TV-Music	Technology
Religion-Spiritual-Culture	Domains
Miscelleneous	Practices Programs Accounts
History-Culture	
Arts	
Books	

Table 4.3: Different communities

All			Unique (reader, post)
	4.5M	$\longrightarrow$	2.5M
	$\downarrow$		$\downarrow$
In communities	2.4M	$\longrightarrow$	1.4M
Non work	2M		1.2M
Work	0.4M		0.24M

Table 4.4: Blog reading counts obtained from the web access logs. The raw data often contains one person accessing a post multiple times. For some purposes it might be more meaningful to consider only the unique instances of a person reading a blog post.

Total # of posts made	Average # of unique readers per post
639	3
534	4
532	3
512	1
480	3

Table 4.5: Average number of people who read posts of the most frequent bloggers. Average over the entire blog dataset is 22

We also have collected web access logs from the blog hosting server. The log contains the reading activity of the employees in the firm. This dataset is prepared to contain reader id, time stamp of reading, and blog post URL. The reading counts are summarized in Table 4.4.

From the examination of the blog posts we found that a majority of the posts are chatters: collection of jokes, quotes, and interesting news items etc. that form no consistent pattern and generate interest for only a short duration. Often they are of limited value to the organization. In a random sample of 50 blog posts we found 31 to be collected joke, quote, or news item and 16 to be original and work related (2 of the remaining were non-English, 1 was empty post). In fact the bloggers who post most frequently exhibit this behavior. However, their posts are not read by many (Table 4.5). Therefore, the bloggers who are the most active in the blogosphere are not necessarily the experts or the most popular bloggers[1]. In order to determine the expertise of a person one needs to consider the content of her posts along with other signals from the blogosphere such as the people from whom the post derives a response.

Buried in the chatters there are long running topics driven by one or a small group of bloggers who could be considered authorities in the subject (See Table 4.6). In this chapter, one of our objectives is to identify such topics along with the bloggers who are driving such topics.

4.4 Application of tensor factorization

4.4.1 Data preparation

We used two subsets of the data for the two methods described in Section 4.2 for blog data analysis. For blog post development analysis we used the text of the blog posts, author-ids, and the timestamp on the post. Texts of the posts are converted to vectors of term weights by removing the stopwords from them and tokenizing them. Only the terms/words that occurred in a certain minimum number of posts (10 in the reported results) were kept. This helps in keeping the vocabulary size small while removing very rare words. Then occurrence counts of the terms in a post are $tf - idf$

[1]Spam was not an issue in this dataset, since, no one outside the firm had access to the blog network

Id: xxx081 Date: 2007-09-05	**Id: xxx991 Date: 2007-11-09**	**Id: xxx368 Date: 2007-10-10**
Diodes can be classified by the functions of the circuit in which it is being used, or more commonly, by the shape that is demanded by the size of the products in which it will be mounted. The complicated point is that there is no direct relation between the two and you must keep them both in your mind at all times. . . .	*Benefits of Automated Testing. If you have ever tested applications or Web sites manually, you are aware of the drawbacks. Manual testing is time-consuming and tedious, requiring a heavy investment in human resources. Worst of all, time constraints often make it impossible to manually test every feature thoroughly . . .*	*20 Minute Home Work Out. If you are busy, not able to get up early morning or have no time for gym just follow this 20 minute home work out to stay healthy and fit. 1. Jog in one place for 3 minutes. Simple light jogging on the spot. 2. Jumping jacks: 25 repeats. When landing, bend your knees slightly to reduce the impact on knee . . .*
(125 more posts by xxx081 in next ten days on "voltage", "diodes" and "semiconductors")	(150 more posts by xxx991 in next eight weeks on "software", "test", "automation")	(190 more posts by xxx368 in next hundred days on "exercise", "muscle", "weight")

Table 4.6: Some of the topics in a blog network along with posting pattern of people behind them.

Number of replies	260K
to blog posts	176K
to other replies	84K
to multiple post/reply	12K

Table 4.7: Descriptive statistics of the reply network

weighted and normalized for document length before using them as term weights. The timestamps were coalesced to a granularity of a day. This data is stored in a $author \times keyword \times timestamp$ tensor. Each cell of the tensor contains total weight of a keyword used by an author on a particular day. This resulted in a $4.6K \times 900 \times 22.5K$ tensor with 4.4M nonzero entries ($sparsity = 6.6 \times 10^{-5}$).

For the blog conversation development analysis we used only the words in the reply text, the author-id, the id of the target blogger and the timestamp. The reason for excluding the blog posts is that it is not clear who the initial blog post is targeted to. To determined the target of a reply we searched for mentions of names of authors who have written the blog post or replied to the blog post prior to the current reply. If one or more names are mentioned in the current reply, then the reply is taken to be targeted to latest posts or replies in the enclosing reply chain by the named authors. If a reply does not contain any name then it is taken to be directed at no one other than the person who wrote the original blog article. The resulting reply statistics is shown in Table 4.7.

We carried out the same transformation of text and the timestamp as done in the case of blog-post-development, but, this time we stored the data in a $author \times recipient \times keyword \times timestamp$ tensor. Each cell of this tensor contains the sum of the weight of a word said by the author to the recipient on a particular day. This resulted in a $16.8K \times 3.8K \times 900 \times 11.8K$ tensor with 1.6M nonzero entries ($sparsity = 3.4 \times 10^{-9}$).

4.4.2 Illustrative results

Each tensor was decomposed into 25 rank-1 tensors using PARAFAC, because evidence from community labels suggest that there are 25 communities in the data. Some of the resulting factors are displayed in Figures 4.2 and 4.3. For each factor the top five keywords are shown along with the importance of the top-authors in descending order and the daily intensity of the topic.

As we can see from Figure 4.2 the decomposition is separating activities in different topics. The intensities of the topics over time gives us insight into how the topic has evolved. Posts about "software testing" have generated interest for much shorter period compared to the conversations about "new technology companies". The importance scores of the top authors also give insight into the nature of the conversation. Posts about physical exercise have seen activity over about 100 days, but, they are primarily made by one person. On the other hand more people have contributed to

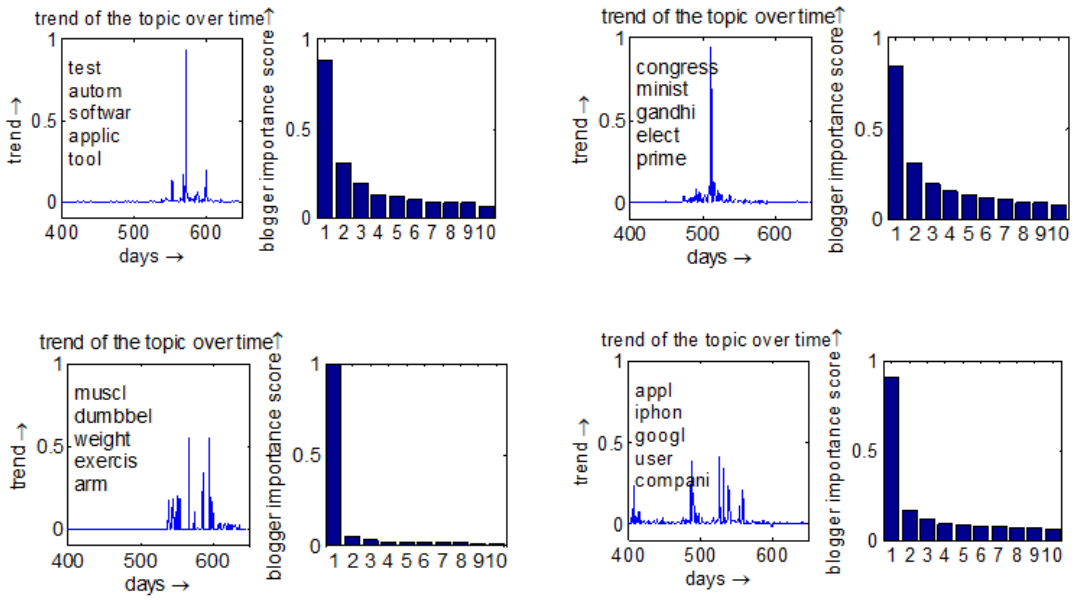


Figure 4.2: Trends of topics and importance of top bloggers in each topic. Four identified topics are illustrated. The top-left one is about software testing. The top-right one is about Indian politics. The bottom-left one is about physical exercise. The bottom-right one is about new technology companies. The histograms of the actor importances show different degree of participation of actors in the topic. For example, the topic of physical exercise can be seen to be driven primarily by one person.

"software testing" and "Indian politics" topics, though they were active for a shorter period. The "Indian politics" topic has one well defined spike. Given our start point of the time axis is 21st April 2006, it turns out that this activity starts around middle of July 2007 which coincides with the Indian presidential election in 2007. The predominance of non-work related posts in the plots shown can be traced to the predominance of non-work related posts in the dataset. Among the posts that were made into any community two-third were non-work related posts.

Analysis of comments on the blog posts reveals a different set of factors. These are usually the factors that generate more than average amount of reactions from the bloggers, e.g., open source softwares (Figure 4.3a), religion (Figure 4.3b), mythology (Figure 4.3c), and Hinduism (Figure 4.3d). In these sets of plots we have a set of most significant reply recipients in addition to the most significant keywords, authors, and days that were present in the previous set of plots. The effectiveness of the decomposition is illustrated by the latent semantic separation of "mythology" and "religion" that can be thought of as a part of broader topic of religion. There were several factors in the result that are about the same overall topic.

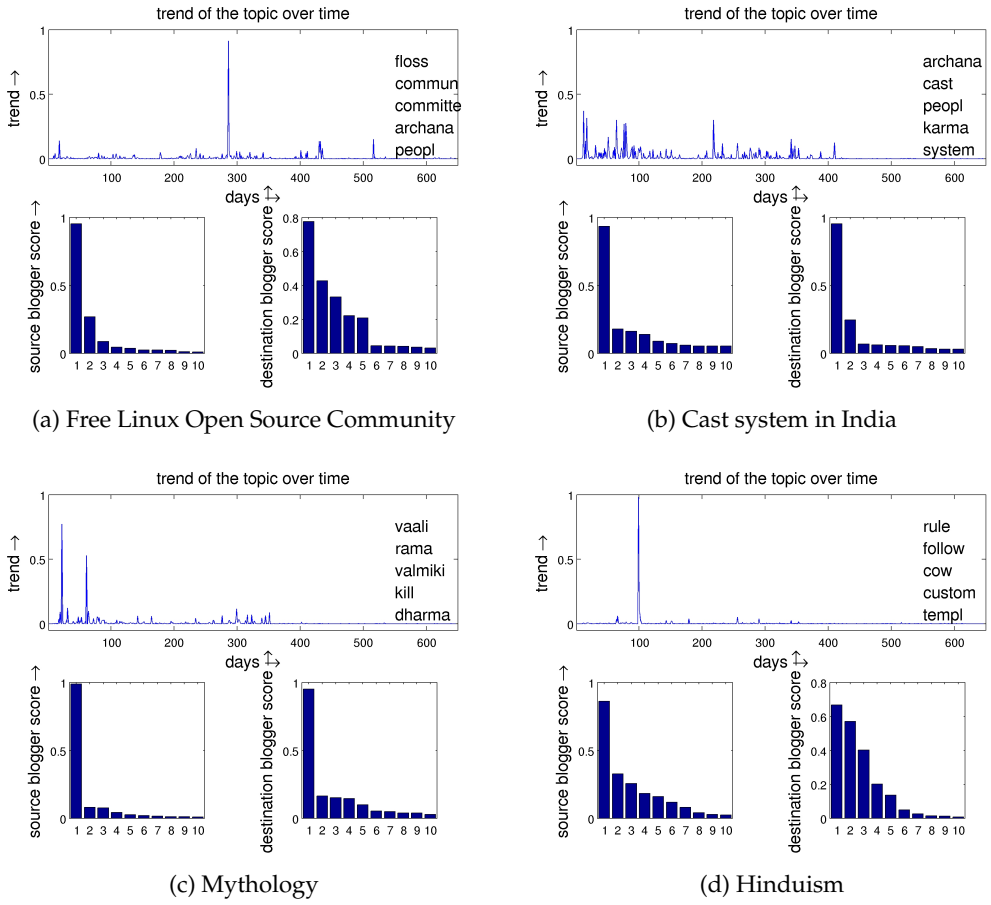
We further analyzed the posts made into individual communities separately. For illustration we present some of the sample topics detected in the community "Sports" (Figure 4.4) and "FLOSS" (Free Linux Open Source Software, Figure 4.5). The topics in the Sports community are largely event driven as can be observed from the intensity of the cricket and soccer topics. The topics in FLOSS community reveal another characteristic. By comparing the distribution of participation of source bloggers to the word description of the topic we can see that topic related to Linux file systems draws a very skewed audience than the topic of multimedia in Linux. This could be because the topic of multimedia applications is popular, but the topic involving file system issues has a niche audience in the FLOSS community.

4.4.3 Comparison with content independent hub and authority

We compare the hubs and authorities identified by tensor factorization with those identified by the HITS algorithm. Since, hubs are measure of the quality of the sender of the responses, we also compare these to the most frequent response makers. For illustration we show the size of the overlap of the top-10 author list identified by these three methods. These are given in Table 4.8. As we can see HITS algorithm, that does not consider the content of the reply or the timestamp of the replies, tends to sample more from the set of most *vocal* bloggers than the PARAFAC tensor factorization algorithm does. Similar behavior is also observed when we compare the top-10 authorities identified by the two factorization algorithms with the top-10 bloggers who received most responses (Table 4.9).

To shed more light on the hubs and authorities identified by PARAFAC and HITS we compare them to the *writers* and *readers* that are central in the blog *reading* network. To compute centrality of the readers and writers over the reading network, the reading data is represented as a $reader \times writer$ matrix (Table 4.10).

Reading can be considered a vote towards the goodness of a writer's writing qual-



(a) Free Linux Open Source Community

(b) Cast system in India

(c) Mythology

(d) Hinduism

Figure 4.3: Trends of topics and importance of topic specific hubs and authorities in each. The plots from the top-left to bottom-right can be roughly described to be on topics "Free Linux open source community", "Cast system in India", "Mythology", and "Hinduism". The histogram of source bloggers show the extent to which the top bloggers have posted in the topic. The histogram of target bloggers show the extent to which they are the recipient of messages in the topic—presumably because of their prior participation in the topic.

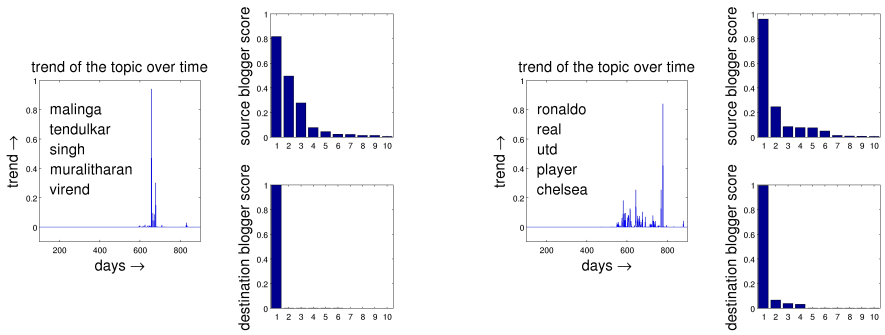


Figure 4.4: Analysis of messages exchanged in the Sports community reveals Cricket and Soccer topics

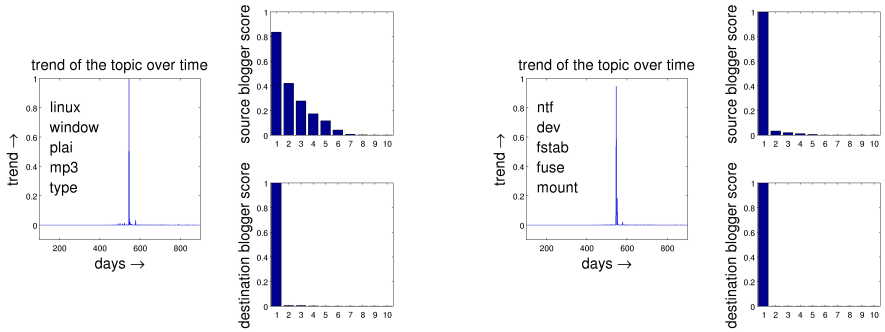


Figure 4.5: Discussion of file systems and Linux medial players in FLOSS community

Factor	$\lvert H_H \cap H_v \rvert$	$\lvert H_P \cap H_v \rvert$	$\lvert H_H \cap H_P \rvert$
1	8	4	4
2	5	4	4
3	5	2	3
4	6	2	2
5	4	3	4
6	4	2	2
7	7	2	4
8	4	4	0
9	3	2	1
10	0	1	0
11	6	6	3
12	0	3	0
13	4	3	1
14	2	0	0
15	3	3	0
16	2	3	0
17	2	2	2
18	2	7	0
19	2	6	0
20	3	1	2
21	4	6	2
22	0	1	0
23	2	4	1
24	0	3	0
25	0	4	0

Where, H_H is the set of top 10 hubs identified by HITS,
H_P is the set of top 10 hubs identified by PARAFAC,
and H_v is the top 10 bloggers who *wrote* most replies.
The table cells show the number of common bloggers in a pair of top-10 bloggers.

Table 4.8: Comparison of hubs.

Factor	$\lvert A_H \cap A_p \rvert$	$\lvert A_P \cap A_p \rvert$	$\lvert A_H \cap A_P \rvert$
1	6	1	2
2	4	3	2
3	4	1	1
4	4	3	0
5	4	3	2
6	8	2	1
7	6	1	0
8	4	3	0
9	2	2	0
10	8	1	1
11	6	2	1
12	7	2	1
13	0	3	0
14	1	2	0
15	2	2	0
16	1	1	0
17	1	3	0
18	0	6	0
19	3	3	2
20	0	4	1
21	1	2	0
22	1	1	0
23	0	3	1
24	1	4	0
25	0	3	0

Where, A_H is the set of top 10 authorities identified by HITS,
A_P is the set of top 10 authorities identified by PARAFAC,
A_p is the set of top 10 bloggers who *received* most replies.
The table cells show the number of common bloggers in a pair of top-10 bloggers.

Table 4.9: Comparison of authorities.

	$writer_1$	$writer_2$	...	...	$writer_N$
$reader_1$	10	15	...		
$reader_2$	1	0	1		
...	...				
$reader_M$					

Cells have the number of times a reader has read a post by a writer.

Table 4.10: Reader writer adjacency matrix

Factor	$cor(A_H, A_R)$	pval	$cor(A_P, A_R)$	pval
1	-0.0024	0.8931	-0.0018	0.9173
2	-0.0016	0.9283	-0.0002	0.9932
3	**0.7279**	0	**0.1019**	0.0000
4	**0.0484**	0.0058	0.0022	0.8984
5	**0.2574**	0	**0.1258**	0.0000
6	**0.4358**	0	**0.1585**	0
7	**0.4876**	0	**0.2309**	0
8	**0.5407**	0	**0.1390**	0.0000
9	**0.4604**	0	**0.2060**	0
10	**0.4685**	0	**0.1480**	0
11	**0.3399**	0	**0.0424**	0.0156
12	**0.1221**	0.0000	0.0332	0.0588
13	**0.1819**	0	**0.3336**	0
14	-0.0014	0.9355	0.0019	0.9137
15	**0.2960**	0	**0.1401**	0.0000
16	**0.1251**	0.0000	**0.0614**	0.0005
17	**0.4833**	0	**0.0977**	0.0000
18	**0.3510**	0	**0.2220**	0
19	**0.5183**	0	**0.1699**	0
20	**0.4733**	0	**0.1486**	0
21	**0.1700**	0	0.0191	0.2770
22	**0.1217**	0.0000	**0.0604**	0.0006
23	**0.2910**	0	**0.1896**	0
24	**0.3356**	0	**0.1608**	0
25	**0.4894**	0	**0.0656**	0.0002

Table 4.11: Correlation of authorities identified by HITS and PARAFAC with centralities of the writers (A_R) on the reading network.

ity. However, there are some readers who are indiscriminate in their reading and read a lot, where as there are others who are selective and follow only a few bloggers who they consider to be good. To measure the goodness of the writers in the reading network taking into account the goodness of the readers who follow them we compute hubs and authorities through an SVD of the reading adjacency matrix.

It is worth noting that the reading activity at the blogs is often invisible on the World Wide Web. Therefore, such a quality metric is often not available. However, in this particular dataset since we have access to the server access log we can identify who is reading whose posts and compute this quality metric.

The correlation of the authorities from HITS and PARAFAC with the authorities in the reading network is given in Table 4.11. The correlation values that are significant at p-value < 0.05 are given in **bold**.

We see that authorities on the reply network obtained by HITS, that does not take into account the words used, have a higher correlation with the authoritative writers on the reading network. It instructive to note that both HITS on $from \times to$ reply network and SVD on the $reader \times writer$ reading network are content free methdods, whereas PARAFAC takes into consider not only who replied to who but also the content of the reply in calculating the authority of the blog authors.

4.4.4 "On topic" quality of the top hubs' response

Tensor factorization simultaneously determines the significant words, authors and recipients in a blog reply network. We illustrate the benefits of such an approach over the existing methods such as HITS that do not take into account the words exchanged between the authors. For this we need a quality metric for the hubs identified by different methods. We decided to measure the quality of the hub by the closeness of the reply to the target post. The idea behind this measure is that a good responder would contribute positively to the topic of discussion by keeping his responses within the topic of discussion as opposed to someone who makes offtopic comments. There are several approaches for computing the distance of a text post from another, e.g., Cosine correlation, Jaccard index, Dice coefficient, KL-divergence etc. We decided to use KL-divergence in this evaluation.

The KL-divergence of probability distributions Q from probability distribution P is (Kullback and Leibler 1951)

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{4.9}$$

To measure the closeness of an authors replies to the targets of the replies we create two probability distribution for each author.

Q : Distribution of the words in all of the replies written by the author

P : Distribution of the words in all the posts and replies that are targets of the replies

A reply could be targeted to the blog article that started reply thread. It could also be targeted to one of the replies within the thread that occurred at a time before the current reply was written. If the name of one of the authors who wrote the blog article or any subsequent reply occurs in the current reply then the current reply is deemed to be targeted to that particular author.

Then the KL-divergence of Q from P are computed for each author. We compare this score for the top hubs identified by the Tensor factorization with the top hubs identified by the HITS algorithm over the entire reply network.

In addition we compare these to a simpler topic specific HITS algorithm. In this approach we identify keywords in the reply set by Latent Semantic Indexing (LSI). For each factor identified by LSI the top 10 words were chosen as the keywords. Hub and Authority scores were computed over the replies that contain these keywords. This approach produces a overlapping but different sets of Hubs and Authorities than the previous two approaches.

It is worth noting that these three approaches form a continuum of methodologies where keyword information in the replies are used to different extents. In the HITS over the entire reply network only the reply count between two authors is used. No keyword information is used in this approach. In the simpler topic specific HITS approach a two step approach is followed. First a set of keywords are selected and then Hub and Authority is computed on the reply network on which those keywords occur. In the tensor factorization approach the keywords, hubs, and authorities are determined simultaneously.

The average KL-divergence of the hubs identified by all three methods are plotted in Figure 4.6. The confidence interval is drawn at two standard deviation from the mean using dashed lines.

From this figure we note that by using keywords in the responses while computing hubs and authorities we can identify hubs that are closer to the target of their response. For the hubs identified by HITS the KL divergences of their replies from the targets is the largest. Thus they can be considered to be most off topic in their replies. The KL divergences of the responses made by the hubs identified by the keyword specific HITS algorithm is lower. So, they can be thought of being closer to the target posts in their response. The hubs detected by the Tensor factorization have the lowest KL-divergence of response from the target posts. Therefore, they can be considered to be most on topic in their response to another post or reply.

4.4.5 Community discovery

We evaluate the effectiveness of the tensor factorization by applying it to discover communities in the blog conversation data. Communities in an online social network are based on user interests that manifest in the text exchanged between the users over the network. We hypothesize that because of its use of multi-modal data a tensor based approach is better suited for discovering communities than methods that do not use the multi-modal data that constitutes conversation within online communities.

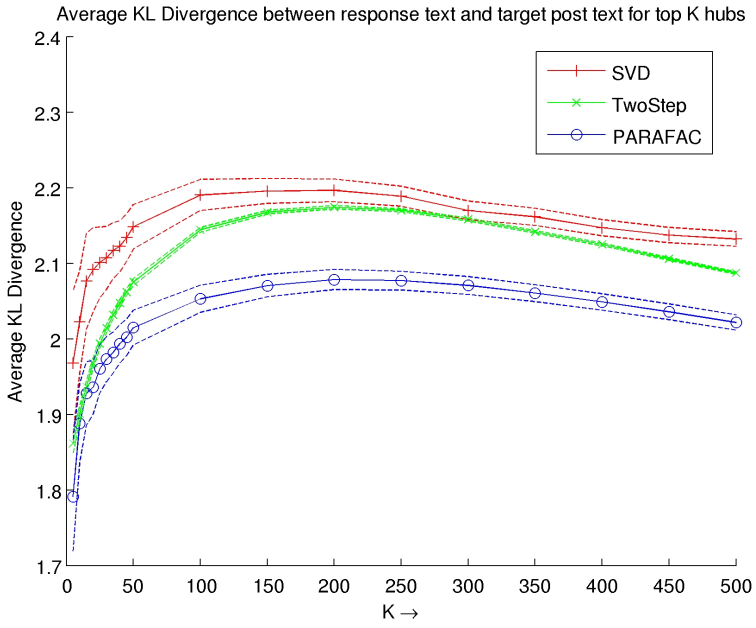


Figure 4.6: Average KL divergences of the top hubs identified by HITS, keyword specific HITS, and Tensor factorization

Definition 2. We define a *conversation* in the blog network to be a set of replies to a blog post from one or more bloggers in the network. Conversation data consists of text, author, target, and timestamp of the replies.

Conversations are different from text documents in that each of them is a set of text messages with author, target, and timestamp.

Task We define the *community discovery* task as given a set of conversations to discover the clusters of similar conversations in them.

To perform this task using tensor factorization we follow two steps:

1. Identify the predominant factors in the entire blog conversation dataset by tensor factorization (Section 4.2.4). Each factor consists of a set of weights over the authors, recipients, keywords, and days.

2. Map each conversation to the factor it most closely aligns with by an operation of *tensor query*

Tensor Query

We pose each conversation as a query to retrieve the factor most similar to it. This is done by approximating each conversation, represented as a tensor $\mathbb{C}$, by a linear combination of the R factors in the original tensor. $\mathbb{C}$ is assigned to the factor with the highest coefficient.

If $\mathbf{A^{(1)}}, \ldots, \mathbf{A^{(M)}}$ are the M modal matrices obtained from the factorization (Equation 4.4) then the approximation is

$$\mathbb{C} \approx \sum_r \beta_r \times \mathbf{a_r^{(1)}} \circ \cdots \circ \mathbf{a_r^{(M)}} \tag{4.10}$$

Let $V(\cdot)$ be the vectorization operator that reshapes a tensor to a column vector in a given order. Let $\mathbf{v_r} = V(\mathbf{a_r^{(1)}} \circ \cdots \circ \mathbf{a_r^{(M)}})$. Then, Equation 4.10 can be expressed as:

$$V(\mathbb{C}) \approx \sum_r \beta_r \times \mathbf{v_r} \tag{4.11}$$

$$V(\mathbb{C}) \approx [\mathbf{v_1}, \ldots, \mathbf{v_r}] * \boldsymbol{\beta} \tag{4.12}$$

where, $*$ is the left product of a matrix with a vector. Equation 4.12 is a least square problem of which $\boldsymbol{\beta}$ is the solution. Magnitude of the elements of $\boldsymbol{\beta}$ indicate how closely $\mathbb{C}$ is aligned to the corresponding factors.

The factor with the largest β_r is the factor that $\mathbb{C}$ aligns most closely with. Therefore, we assign the conversation to that factor. Repeating this for each conversation in the dataset gives us a set of clusters of conversations. These clusters of conversations can be evaluated using one of many cluster evaluation techniques if we have some kind of true label on each conversation. We use the author provided "community labels" that is part of the blog dataset (Table 4.3) for this purpose.

	Macro average		Micro average	
	Tensor factorization	Document clustering	Tensor factorization	Document clustering
P	0.4	0.46	0.25	0.37
R	0.73	0.35	0.71	0.27
F	**0.51**	**0.40**	**0.37**	**0.31**

Table 4.12: Precision recall of community discovery

Experiment setup and results

For community discovery we used only the part of the dataset that has community labels on it, so that we can evaluate the resulting communities. Only the reply data is used to form conversations among bloggers. This dataset has 180K responses forming 21.5K conversations. We cluster these conversations into 25 communities using tensor factorization. To obtain a baseline to compare with we clustered the response text using repeated bisection document clustering algorithm. Using the author provided community labels we computed the precision (P), recall(R), and the F-score for the clusters formed (Section 2.5.1). The scores are shown in Table 4.12.

As we can see the tensor factorization has a lower P score meaning the communities it is uncovering have more of the other community conversations than the communities uncovered by document clustering algorithm. However, tensor factorization is keeping conversations in each community together in one cluster more so than document clustering algorithm (reflected in larger R value). This leads to a higher F score for the tensor factorization approach for community discovery. The results show that the conversation clusters generated by tensor factorization agrees with the author provided labels on the blog articles more than the clusters generated by document clustering do.

4.5 Conclusion

In this chapter an importance definition is proposed for multi-modal data is part of online social conversations. The definition is a higher order extension of importance computation approaches already seen in the literature, namely in eigenvector centrality computation, latent semantic indexing, etc. This higher order computation requires a multi-modal representation of the data that is not possible with matrices. We use tensors to represent the multi-modal data and perform operations on it.

We show that the proposed importance of the entities in the multi-modal data can be computed by a process of tensor factorization. To illustrate our results we collect blog data from the corporate Intranet of a large IT services firm. We capture the blog data in two different tensors representing two views of the blogosphere: 1. Blogs are self publication media, 2. Blogs are also a place to discuss topics and exchange ideas for people with common interests. We apply tensor factorization to illustrate

significant topics of conversation, important bloggers in each, and the development of the topic over time.

We also assess the quality of the hubs identified by tensor factorization vis-a-vis HITS on the entire network and keyword specific HITS. For each author we measure the closeness of the replies from the target posts of their replies using KL-divergence. The lower the KL-divergence the more on topic the authors' replies are. The hubs identified by Tensor factorization had the smaller KL-divergence from their targets than the hubs identified by keyword specific HITS. HITS on the entire reply network, without taking into account the content of the replies, produces hubs with highest KL-divergence from the target.

To evaluate the tensor factorization approach we apply it to discover communities in the conversations occurring in the blog network. Conversations are defined as a set of messages exchanged in response to a blog post. The idea behind this task is that if we can identify important actors in different communities and important keywords used by the people in those community we should be able to identify different communities using these information. To achieve this objective we propose a *tensor query* operation to retrieve the factor most similar to a conversation posed as a query. This operation takes into account the weight of entities in each mode of the data to compute the similarity of the conversation to a factor. Using author provided community labels on the conversations we show that tensor factorization performs better than document clustering on conversation text in identifying communities in the conversation.

The primary limitation of this study is that the claimed importance definitions have not been *directly* validated. Obvious importance measures, such as number of times a blogger's posts are read, importance of a blogger based on importance of people who read the blogger's post, are content independent. In our experiment we found them to be better predicted by simpler eigenvectors of the adjacency matrix containing number of replies between the bloggers. The importance of a blogger in a specific topic, although makes intuitive sense, is difficult to obtain as part of a dataset. One approach to obtain such information is by asking experts about the important topics in a given blog network and who are the important bloggers in them. Since, the data used in this study is collected from a private blog network of an IT services firm we have to leave this measurement for a future study.

Tensors are a natural extensions of matrices for capturing and analyzing multimodal datasets. We have shown their application for community discovery in online conversations. However, there are several future research directions. One of the assumptions in this work is the multiplicative endowment of importance due to association. Although, it allows us to use existing tensor factorization techniques for importance computation the assumption may not be valid in all situations. For example (Bonacich 1987) has shown that being surrounded by powerful actors in a network might *reduce* the leverage and power of an actor. Another direction of research is the application of non-negative tensor factorization to obtain entity weights that are non-negative. Non-negative weights are are easier to interpret in many real word scenarios.

CHAPTER 5

Conclusion

This thesis develops data mining applications focusing on three areas of enterprise information system.

The first chapter develops an algorithm to organize and discover topics from streams of unstructured text documents. Most organizations are faced with these kinds of data from newswires, press releases, external blogs and online social networks: sources they must track to be cognizant of the competitive landscape they are conducting business in. We extend an existing incremental hierarchical clustering algorithm in the literature, namely COBWEB/CLASSIT for clustering text document. This was achieved by a conceptual separation of the control flow of the CLASSIT algorithm from the parameter estimation for the data attributes and modifying the parameter estimation for the specific case of word occurrence distribution in documents.

The second chapter develops a collaborative filter to recommend products to consumers based on the consumers' prior ratings on a subset of products. Collaborative filters are used by many online merchants to identify promising target customers to promote a product to and to help customer shift through large number of product offerings to identify the ones they like. Traditionally collaborative filters use one dimensional ratings from customers evaluating the overall quality of the product as perceived by them. Often there are multiple aspects to a product experience, such as, acting, directorial style etc. of a movie; food quality, ambience, service at a restaurant etc. However, we find that user ratings along these different aspects of a product are highly correlated. In this chapter we discover the dependency structure among the rating components in a movie dataset and use the structure in a mixture model that is then used to recommend movies to customers. We compare it to several one component and multi-component algorithms to show the effectiveness of the proposed algorithm.

The third chapter proposes a framework to represent and analyze conversations in an intra-organizational online social network. With the advent of Enterprise 2.0 organizations are creating internal blogs and online social networks to effectively engage with their employees. In an online social network the relation between people develops through the exchange of text messages that form online conversations. The ability to observe and analyze such messages between employees allows us to monitor development of topics and identify significant actors. Most of the literature on topic discovery and the literature on discovering significant actors in a network focuses

Chapter	Method	Data	Application
1	Incremental document clustering	Unstructured text documents	Topic discovery and organization from a stream
2	Dependency structure discovery and mixture model	User-Item multi component ratings	Recommender system using multiple aspects of rating
3	Tensor factorization	Text messages with author, recipient and timestamp	Detect topics and significant people from online conversation analysis

Table 5.1: Thesis summary

on two dimensional co-occurrence data. We extend the significance definitions introduced in the two dimensional document analysis and network analysis methods to the higher order tensors capturing author, recipient, keywords, and timestamps. This chapter shows that such a definition can be applied via a process of tensor factorization to discover communities in the blog network and identify important responders in the network who happen to be more focused in their discussion of a given topic.

These three chapters are outlined in Table 5.1.

Future directions

Information systems in enterprises are rife with interesting data mining opportunities. There are several logical extensions of the presented work. The probabilistic model for collaborative filtering can be extended with user attributes to further guide the recommendations. In an enterprise setting recommending the right sets of work related documents and posts to employee is an important application. In such an application the user attributes of interest might include the department, roles, and skillsets. Another promising extension of multi-component rating collaborative filter is in the context of online reviews and ratings. In many online review sites consumers provide text reviews along with the ratings. Text reviews are inherently multi-dimensional as they contain potential information about more than one aspect of the product. Once such dimensions present in the reviews are identified through text analysis, the proposed model for multi-component rating collaborative filtering can be used to incorporate such multi-dimensional information.

Tensors are natural extensions of matrices for representing and analyzing multi-modal data. However, several interesting open research problems still remain. Factorization of tensors with time dimensions need special handling because as opposed to two different authors, data collected from two different days are likely to have a continuity among them. Alternating Least Square for tensor factorization can be extended to impose this continuity over the data values along time dimension. In the work completed as part of this thesis validation of the importances computed is provided through a topic discovery exercise and by measuring how on-topic the hubs are

in their responses. However, direct validation of the importance of the identified hubs is still an open research problem. Such a validation can be carried out in an enterprise setting by collecting the official status of the employees in the enterprise.

APPENDIX A

Document Clustering

A.1 MLE of Katz's distribution parameters

The Katz's distribution is defined as:

$$\begin{aligned} P(0) &= p_0 \\ P(k) &= (1-p_0)(1-p)p^{k-1}; \text{when } k > 0 \end{aligned} \tag{A.1}$$

where, p_0 and p are the parameters of the distribution.

Let us discuss about the distribution of only one word or term. The data is the count of occurrences of the word in each document in the text collection. So, if we have N documents in the dataset we have N observations, each of which is a count.

Let us also define n_k to be the number of observations equal to k, i.e., number of documents in which the term occur k times. Let's assume the maximum value of k is K.

Hence,

- document frequency $\mathrm{df} = N - n_0 = \sum_{k=1}^{K} n_k$ and
- collection term frequency $\mathrm{cf} = \sum_{k=1}^{K} k n_k$

The likelihood $L(p, p_0)$ of the parameters given data is

$$\begin{aligned} &= \prod_{i=1}^{N} \Pr(\text{the word occurs } x\text{times in document } i) \\ &= \prod_{i=1}^{N} \left[\delta(x)p_0 + (1-\delta_k)(1-p_0)(1-p)p^{x-1}\right]; x \in 1 \ldots K \\ &= p_0^{n_0} \prod_{k=1}^{K} (1-p_0)^{n_k}(1-p)^{n_k}(p^{k-1})^{n_k} \end{aligned}$$

where, $\delta(\cdot)$ is the indicator function that is 1 if argument is zero and 0 otherwise.

Log of likelihood is

$$\begin{aligned} & \mathrm{LL}(p, p_0) \\ = \; & n_0 \log(p_0) \\ + \; & \sum_{k=1}^{K} \left[n_k \log(1-p_0) + n_k \log(1-p) + n_k (k-1) \log(p) \right] \end{aligned}$$

Taking the partial derivative of the log likelihood with respect to p_0 and equating it to 0:

$$\begin{aligned} \frac{\partial \mathrm{LL}}{\partial p_0} &= \frac{n_0}{\widehat{p_0}} + \sum_{k=1}^{K} n_k \frac{-1}{1-\widehat{p_0}} = 0 \\ \Rightarrow \frac{n_0}{\widehat{p_0}} &= \frac{1}{1-\widehat{p_0}} \sum_{k=1}^{K} n_k = \frac{1}{1-\widehat{p_0}} (N - n_0) \\ \Rightarrow \frac{1-\widehat{p_0}}{\widehat{p_0}} &= \frac{N-n_0}{n_0} \\ \Rightarrow \frac{1}{\widehat{p_0}} - 1 &= \frac{N}{n_0} - 1 \\ \Rightarrow \widehat{p_0} &= \frac{n_0}{N} = \frac{N - \mathrm{df}}{N} = 1 - \frac{\mathrm{df}}{N} \end{aligned} \tag{A.2}$$

We can find the MLE of p in a similar manner.

$$\begin{aligned} \frac{\partial \mathrm{LL}}{\partial p} &= \sum_{k=1}^{K} n_k \frac{-1}{1-\hat{p}} + \frac{n_k (k-1)}{\hat{p}} = 0 \\ \Rightarrow 0 &= \frac{1}{\hat{p}} \sum_{k=1}^{K} n_k (k-1) - \frac{1}{1-\hat{p}} \sum_{k=1}^{K} n_k \\ \Rightarrow 0 &= \frac{1}{\hat{p}} \left(\sum_{k=1}^{K} k n_k - \sum_{k=1}^{K} n_k \right) - \frac{1}{1-\hat{p}} \sum_{k=1}^{K} n_k \\ \Rightarrow 0 &= \frac{1}{\hat{p}} (\mathrm{cf} - \mathrm{df}) - \frac{1}{1-\hat{p}} \mathrm{df} \\ \Rightarrow \frac{1-\hat{p}}{\hat{p}} &= \frac{\mathrm{df}}{\mathrm{cf} - \mathrm{df}} \\ \Rightarrow \frac{1}{\hat{p}} &= \frac{\mathrm{cf}}{\mathrm{cf} - \mathrm{df}} \\ \Rightarrow \hat{p} &= \frac{\mathrm{cf} - \mathrm{df}}{\mathrm{cf}} \end{aligned} \tag{A.3}$$

Expressions (A.2) and (A.3) are the MLE of the parameters of Katz's distribution described in Expression

APPENDIX B

Multi-component Rating Collaborative Filtering

B.1 Derivation of marginal distributions

This involves calculating the sum over the large joint distribution exploiting the conditional independencies. Marginalizations for each of the three models are shown below.

B.1.1 $P(U, I, O)$ for the model with dependency among the ratings

The characteristic of this model is Overall rating is a parent node of the component rating variables in addition to the Z_u and Z_i latent class variables.

$$
\begin{aligned}
P(U,I,O) &= \sum_{Z_u}\sum_{Z_i}\sum_{S}\sum_{A}\sum_{V}\sum_{D} P(Z_u, Z_i, U, I, S, A, V, D, O) \\
&= \sum_{Z_u}\sum_{Z_i}\sum_{S}\sum_{A}\sum_{V}\sum_{D} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)P\left(S|Z_u, Z_i, O\right) \\
&\quad P\left(A|Z_u, Z_i, O\right) P\left(V|Z_u, Z_i, O\right) P\left(D|Z_u, Z_i, O\right) P\left(O|Z_u, Z_i\right) \\
&= \sum_{Z_u}\sum_{Z_i} P\left(O|Z_u, Z_i\right) P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u) \sum_{S} P\left(S|Z_u, Z_i, O\right) \\
&\quad \sum_{A} P\left(A|Z_u, Z_i, O\right) \sum_{V} P\left(V|Z_u, Z_i, O\right) \sum_{D} P\left(D|Z_u, Z_i, O\right) \\
&= \sum_{Z_u}\sum_{Z_i} P\left(O|Z_u, Z_i\right) P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u) \\
&= \sum_{Z_u} P(Z_u)P(U|Z_u) \sum_{Z_i} P\left(O|Z_u, Z_i\right) P(Z_i)P(I|Z_i) \qquad \text{(B.1)}
\end{aligned}
$$

The conditional probability terms for $S, A, V \text{ and } D$ could be marginalized and eliminated, since those probabilities sum to 1.

B.1.2 $P(U, I, O)$ for the model with independent component ratings

The characteristic of this model is that all component ratings and Overall rating are independent of each other conditional on the Z_u and Z_i latent class variables.

$$\begin{aligned}
P(U,I,O) &= \sum_{Z_u}\sum_{Z_i}\sum_{S}\sum_{A}\sum_{V}\sum_{D} P(Z_u, Z_i, U, I, S, A, V, D, O) \\
&= \sum_{Z_u}\sum_{Z_i}\sum_{S}\sum_{A}\sum_{V}\sum_{D} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)P\left(S|Z_u, Z_i\right) \\
&\quad P\left(A|Z_u, Z_i\right) P\left(V|Z_u, Z_i\right) P\left(D|Z_u, Z_i\right) P\left(O|Z_u, Z_i\right) \\
&= \sum_{Z_u}\sum_{Z_i} P\left(O|Z_u, Z_i\right) P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u) \sum_{S} P\left(S|Z_u, Z_i\right) \\
&\quad \sum_{A} P\left(A|Z_u, Z_i\right) \sum_{V} P\left(V|Z_u, Z_i\right) \sum_{D} P\left(D|Z_u, Z_i\right) \\
&= \sum_{Z_u}\sum_{Z_i} P\left(O|Z_u, Z_i\right) P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u) \\
&= \sum_{Z_u} P(Z_u)P(U|Z_u) \sum_{Z_i} P\left(O|Z_u, Z_i\right) P(Z_i)P(I|Z_i) \qquad \text{(B.2)}
\end{aligned}$$

B.1.3 $P(U, I, O)$ for the model with only the overall ratings

$$\begin{aligned}
P(U,I,O) &= \sum_{Z_u}\sum_{Z_i} P(Z_u, Z_i, U, I, O) \\
&= \sum_{Z_u}\sum_{Z_i} P\left(O|Z_u, Z_i\right) P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u) \\
&= \sum_{Z_u} P(Z_u)P(U|Z_u) \sum_{Z_i} P\left(O|Z_u, Z_i\right) P(Z_i)P(I|Z_i) \qquad \text{(B.3)}
\end{aligned}$$

B.1.4 $P(U, I, S, O)$ from the complete joint distribution

$$
\begin{aligned}
P(U,I,S,O) &= \sum_{Z_u}\sum_{Z_i}\sum_{A}\sum_{V}\sum_{D} P(Z_u, Z_i, U, I, S, A, V, D, O) \\
&= \sum_{Z_u}\sum_{Z_i}\sum_{A}\sum_{V}\sum_{D} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)P\left(S|Z_u, Z_i, O\right) \\
&\quad P\left(A|Z_u, Z_i, O\right) P\left(V|Z_u, Z_i, O\right) P\left(D|Z_u, Z_i, O\right) P\left(O|Z_u, Z_i\right) \\
&= \sum_{Z_u}\sum_{Z_i} P\left(O|Z_u, Z_i\right) P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)P\left(S|Z_u, Z_i, O\right) \\
&\quad \sum_{A} P\left(A|Z_u, Z_i, O\right) \sum_{V} P\left(V|Z_u, Z_i, O\right) \sum_{D} P\left(D|Z_u, Z_i, O\right) \\
&= \sum_{Z_u} P(Z_u) \sum_{Z_i} P(Z_i)P\left(O|Z_u, Z_i\right) P(I|Z_i)P(U|Z_u)P\left(S|Z_u, Z_i, O\right) \\
\Rightarrow P(u,i,S,o) &= \sum_{Z_u} P(Z_u) \sum_{Z_i} P(Z_i)P\left(o|Z_u, Z_i\right) P(i|Z_i)P(u|Z_u)P\left(S|Z_u, Z_i, o\right) \\
&\propto P(S|u,i,o)
\end{aligned}
$$

B.2 Halo in multi-criteria movie rating

B.2.1 Halo Effect

The phenomenon of observing a higher than expected correlation between ratings collected from human subjects is known as the Halo effect. It was first identified by Wells as a constant error in rating because raters seem to rate subjects for the general merits at the time of rating them for their individual qualities (Wells 1907). Wells has indicated that this constant error is probably not a serious concern and it is difficult to see how it could have been avoided ((Wells 1907) page 21). After about a hundred years of research we still do not have an agreed upon way to prevent, measure or correct halo effect. And there is disagreement in the research community whether Halo is a completely harmful phenomenon (Cooper 1981, Fisicaro 1988).

Thorndike was the first to term this error as Halo error (Thorndike 1920). The paper makes many interesting correlation observations. Correlation between rating of general ability and technical ability of aviation officers was found to be 0.67 where as the author states that the true correlation could not be more than 0.25 after attenuation correction. Students' ratings of the voice of their teachers was found to be correlated at 0.50 with the teachers' interest in community service and at 0.67 with the intelligence. Thorndike asserts that since, these correlations are much higher than the correlation that can be expected between true scores, this is something added by the raters. Although, this universal observation of very high correlation makes a case for something systematic affecting the correlation between the rating components, the manner in which the problem is highlighted indicates a problem that will resurface again and

again: we collect ratings—in most of the cases—when the true scores are impossible to collect, e.g., leadership quality of an officer, lucidity of a teacher's discourse, direction quality of a movie. Therefore, how do we compare the observed rating correlations with a true score correlation and possibly measure the halo effect? Thorndike himself lamented that although this halo effect seems large, we lack an objective criteria by which to determine its exact size.

Sources of Halo

Since, halo is the higher than expected correlation between two rating variables it can be due to two reasons. One reason lies with the raters' behavior, who due to their cognitive distortion add to the correlation of the rating attributes. This is known as *illusory halo*. Cooper (1981) has outlined five related factors that might lead to this behavior(Cooper 1981):

1. Rater's lack of familiarity with the target might prompt him to rely on his global impression and give component ratings based on how he thinks the categories co-vary,

2. Rater might assume that the rating components co-vary with his global impression or with salient components,

3. Components might not be well defined, which would lead the raters to group together somewhat related evidences to generate ratings,

4. Rater might be unwilling to put in enough effort to distinguish between the components or be sensitive to the fact that he might be committing Halo error,

5. If there is a time gap between the observation and rating collection, then the rater might forget the details and add his bias about how rating components co-vary (Shweder 1975).

The second reason could be in the design of the rating components. They may be truly correlated—even before the rater added their error. This is known as the *true halo*. Traditionally (Thorndike 1920, Wells 1907) when one refers to the halo error, it is understood that they are referring to the *illusory halo*. But, it is important to be aware of the difference between the two since, often they co-occur (with true halo possibly affecting the illusory halo(Kevin R. Murphy 1988)) and the metrices that claim to measure the halo are unable to distinguish between the two and measure a combined effect.

Yet another perspective on Halo is provided by (Murphy 1992). They have found from several laboratory experiments that the Halo effect is not a property of the rater or ratee, but, a property of the unique rating situation. If this is true, then an appropriate method of measuring halo should measure the halo on for each rating instead of measuring Halo present in a set of ratings.

Measuring Halo

The main methods discussed in literature to detect and/or measure halo are:

1. Noticing the difference between observed correlation and estimated true correlation. This method was used by Thorndike. Although, this is probably the most direct way to access the halo effect, the problem with this method is that true correlations are often not available.

   > Even when it is possible to compute the correlations from the true scores, the random measurement error attenuates the computed correlation to a value lower than the true correlation. This would inflate the perceived Halo. Hence, we must make correction for this attenuation (Fisicaro 1990).

2. Computing standard deviation across the rating components. The lower the standard deviation, the higher the halo effect. This method does not work very well when the rating components have naturally different mean, in which case it will show a inter-component standard deviation even when the components are perfectly correlated (J. T. Lamiell 1980, Elaine D. Pulakos and Ostroff 1986).

3. Identifying inter-component factor structure. If there is one dominant factor then it suggests that the ratings are generated from this one factor and the rater does not distinguish between various rating components. This suggests presence of halo (D. Kafry 1979).

4. Carrying out a rater×ratee×category anova. If there is a rater×ratee effect then halo is said to be present (M. J. Kavanagh 1971).

Method 1, 2 and 3 has been discussed without a clearly laid out criteria for detecting the halo effect. None of the above methods distinguish illusory halo from true halo. There has been limited attempt at examining the true halo and illusory halo by using certain manipulated rating components for which true halo can be computed (Cooper 1981).

Reducing Halo at rating time

It has been observed that increasing the rater-target familiarity reduces the effect of the halo because it gives the rater a larger sample of target attribute or behavior to base the component ratings on and not fallback on his general impression ((Koltuv 1962), (Heneman 1974), (Landy and Farr 1980)).

Sometimes halo effect is observed because of the raters making judgment based on factors that are not relevant to the components they are rating. It has been found that by explicitly asking the raters to rate *key irrelevant* factors, i.e., the factors that might influence the component ratings but should not, such effect can be reduced (Rizzo and Frank 1977).

Shewder and D'Andrade has shown that halo is consistently higher when ratings are collected for older observations(Shweder and D'Andrade 1980). Borgatta et al. has shown that the rating collected during the observation of target is consistently less haloed than the ratings collected after observations (E. F. Borgatta 1958). Therefore, another strategy to reduce halo in rating could be to collect the ratings at the time of observation or as soon as possible after the observation.

Training the rater through lectures and discussion groups to reduce halo in their ratings have given mixed results. The only method that has given any consistent gain is the use of workshops to sensitize the raters to the halo error they might commit by giving them feedback on their ratings (G. P. Latham 1980, Borman 1979, Ivancevich 1979).

Correcting Halo after collecting the ratings

It has been found that average over the component ratings obtained from multiple raters has lower halo than the individual rater's ratings (J. S. Berman 1976). However, this solution is not always feasible due to lack of adequate raters (Cooper 1981). Moreover, this might lead to more accurate ratings, but, averaging does not help when we are interested in correcting halo occurring for a rater-target pair (e.g. for the purpose of collaborative filtering).

Another method to remove excessive correlation among the components due to the presence of a dominant component is by statistically removing the effect of the component, usually a global rating component (Holzbach 1978). Holzbach observes that it is almost impossible to collect ratings that are free from Halo. However, if we can collect the rating on a global impression component then we might be able to remove the effect of this global component from other components. In a study containing ratings along six job related behaviors and a global rating, he found that if we remove the effect of the global component from the six behavior ratings we can reduce the inter behavior component correlation. To remove the effect of the global component he regresses the six behavior ratings against global component and computes the correlation among the residuals. This is equivalent to computing the partial correlation between the behavior components while holding the global component constant. The component residuals remaining after the regression were less correlated than the components themselves. This by itself may not be a significant result, because, controlling for any variable that is not perfectly uncorrelated with the component ratings would reduce the component correlations when the correlations are all positive: as was the case in Holzbach's work. What is interesting is that this correction leads to a more understandable factor structure among the components instead of a general component dominated one. Holzbach also reanalyzed three of the previously published studies using his proposed statistical correction method and found that he was able to reduce the effect of the halo. Landy et al., used Holzbach's method to remove halo from a rating set of 15 job related behavior and a global rating component and found that median of inter-component correlation reduced from 0.36 to 0.07. Also, the factor analysis results of the ratings changed from a general factor dominated

	s	a	v	d	o
s	1.00	0.79	0.82	0.74	0.87
a	0.79	1.00	0.81	0.73	0.83
v	0.82	0.81	1.00	0.79	0.88
d	0.74	0.73	0.79	1.00	0.80
o	0.87	0.83	0.88	0.80	1.00

Table B.1: Correlation among components of rating—suggesting the presence of a Halo effect

$r_{R_iR_j.O}$	S	A	V	D
S	1.00	0.25	0.26	0.15
A	0.25	1.00	0.32	0.22
V	0.26	0.32	1.00	0.33
D	0.15	0.22	0.33	1.00

Table B.2: Partial correlation given Overall. $R_i, R_j \in \{S, A, V, D\}$

three factor structure to a more interpretable six factor structure(Steele 1980). Before Holzbach, Myers had taken a similar approach to reduce halo where he used job levels as control variable to reduce correlation among job dimensions (Myers 1965).

B.2.2 Halo in movie rating data

Correlation structure

Looking at the correlation matrix we find that the components are highly correlated.

This indicates that there probably is a halo effect in the collected ratings. However, following a procedure like Holzbach's where we statistically remove the effect of the Overall component by taking partial correlation we find that the effect of halo is much less.

The average inter-component correlation among variables S, A, V, D has reduced from 0.78 to 0.26. As all correlations are positive we should expect some reduction in correlation when computing partial correlations. However, the average partial correlation among the variables is the least when we control for the variable O among the possible five variables. The average partial correlations when we controlled for *S,A,V,D* were $0.47, 0.53, 0.35$and0.60 respectively. These results confirms, using a much larger dataset, Holzbach's findings that controlling for Overall rating reduces the Halo. It also shows that this reduction is consistently more than the reductions obtained by controlling for variables other than Overall rating.

There are factors other than the Overall impression that might be responsible for dependency among ratings. For instance, perhaps some pairs of components are harder to distinguish between, because of ambiguity in those component definitions. That would lead to correlation among that pair of components (3rd point in Cooper's

list and to some extent 4th point too). From the partial correlation matrix it seems that there is some ambiguity between Visuals and Direction quality (0.33 partial correlation), Story and Direction (0.32 partial correlation). Or may be there is some true correlation among these pairs. Cooper's point 1, 2, and 5 supports a "general impression leading to higher correlation between all pairs" theory and his 3rd and 4th reason makes it possible to have higher inter-component correlation between specific pairs of components.

PCA

Another method to detect Halo is to carry out a Principal Component Analysis of the correlation matrix and look for the presence of a dominant component. If we take a linear combination of the five variables using weights given by eigen vectors of the correlation matrix or covariance matrix to create new variables, then the five new variables will have zero correlation between themselves and will have variance equal to the corresponding eigen values. Another important property is that if we order the new variables in the decreasing order of their eigen values, then the first new variable will have the highest possible variance among all variables that one may construct by linear combination of the original variables. The second new variables will have the highest variance among all possible variables that we may construct by linearly combining the original variables while keeping it uncorrelated to the first constructed variable. And similarly for the remaining new variables. These variances are same as the eigen values and the sum of these variances is exactly equal to the sum of the variances of the original variables. So, we can find out how much of the entire variance is explained by these newly constructed variance (Morrison 1967).

The eigenvalues of the correlation matrix (Table B.1) are:

Factor	One	Two	Three	Four	Five
Eigen values	4.22	0.28	0.22	0.16	0.11
% variance explained	84.5	5.7	4.4	3.2	2.2

Table B.3: Eigen values of the correlation matrix

This suggests that if we construct uncorrelated variables by linear combination of these five variables so that they have maximum variance then we can find one variable that will have 84.5% of the total variance. A second variable can be constructed by linear combination of the five original variables that has 5.7% of the total variance, while being under the constraint that this second variable has zero correlation with the first. Similarly the remaining variables can be constructed. In other words, if we perform a rigid rotation of the axes—they stay perpendicular to each other—of the five dimensional rating space, 84.5% of the variance would lie along one of the new axis, 5.7% of the variance along another and so on (Morrison 1967). The dominant presence of one component that explains a large amount of variance indicates the presence of a Halo effect among the rating components(Holzbach 1978).

Factor	One	Two	Three	Four
Eigen values	1.77	0.86	0.73	0.63
% variance explained	44.3	21.6	18.3	15.8

Table B.4: Eigen values of the partial correlation matrix

	Factor 1	Factor 2	Uniquenesses
S	0.91	0.23	0.11
A	0.87	−0.02	0.21
V	0.93	−0.08	0.10
D	0.84	−0.12	0.25
O	0.95	0.03	0.07

Table B.5: Factor loadings after quartimax rotation

However, after partialing out the Overall component, i.e., using the Table B.2 we find that the largest component becomes much less dominant—suggesting a reduction in halo.

Factors

Yet another way of detecting the presence of a Halo effect to look for the presence of a factor structure that is dominated by one factor (Holzbach 1978). In factor analysis we try to express each observed rating component as a linear combination of some hidden variables and an error term unique to the component. In this analysis several rotations of the factor loading matrices were tried. Quartimax rotation, which tries to reduce the number of factors for each variable, gives the following structure.

This is dominated by one factor, which points to the presence of a halo. It is interesting to note that most of the variation in the Overall component can be explained by these underlying factors (low uniqueness), but, not as much of the variation in other component variables can be explained by these underlying factor. This suggests that these underlying factors are the closest to the Overall rating[1].

Effect of experience in rating

Studies have shown that training the raters to sensitize them to the halo error can reduce the halo error in their ratings(William T. Hoyt 1999). But, it is not clear whether more experience in rating leads to a lower Halo error. To examine this the halo effect in the ratings of people who have rated different amount of movies were measured using the proportion of variance explained by principal components and by the average inter-component correlation.

[1]The limitation of existing Chi-square test prevents us from using more than two hidden variables to in our Factor analysis of five variables (Morrison 1967)

Users with # of ratings	# of records	fraction of variance explained by components					avg . corr
$\leqslant 5$	346973	0.877	0.042	0.035	0.026	0.02	0.84
$> 5\& \leqslant 10$	37473	0.862	0.048	0.039	0.029	0.021	0.82
$> 10\& \leqslant 20$	27378	0.848	0.053	0.042	0.033	0.024	0.80
$> 20\& \leqslant 40$	18519	0.837	0.057	0.045	0.036	0.026	0.79
> 40	25493	0.855	0.050	0.041	0.031	0.022	0.82

Table B.6: Fraction of variance explained by the principal components and average correlation among the components.

The variance explained by the principal components and the average correlation among components for different groups of users are not very different. Therefore, it does not seem like users with more experience make less halo error. One possible explanation could be that traditionally when people rate a lot of subjects they learn more about rating by receiving some kind of feedback. But, in movie rating it is hard to see how the rating behavior would change since the raters don't get *any feedback*. Cooper has indicated that among rater training programs that consists of lectures, group discussion and workshops, only workshops have produced any consistent reduction in halo. He has indicated that the reason might be that the workshops monitor raters and give them corrective feedback when they commit error (Cooper 1981).

References

Adomavicius, G., Y.O. Kwon. 2007. New Recommendation Techniques for Multicriteria Rating Systems. *IEEE Intelligent Systems* **22**(3) 48–55. 45, 48, 62, 63

Adomavicius, Gediminas, Ramesh Sankaranarayanan, Shahana Sen, Alexander Tuzhilin. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.* **23**(1) 103–145. doi:http://doi.acm.org/10.1145/1055709.1055714. 47

Adomavicius, Gediminas, Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng* **17**(6) 734–749. URL http://doi.ieeecomputersociety.org/10.1109/TKDE.2005.99. 45

Allan, James, Ron Papka, Victor Lavrenko. 1998. On-line new event detection and tracking. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM Press, 37–45. doi:http://doi.acm.org/10.1145/290941.290954. 13, 16, 17, 82

Bader, Brett W., Tamara G. Kolda. 2007. Matlab tensor toolbox version 2.2. URL http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/. 87

Bader, B.W., R.A. Harshman, T.G. Kolda. 2007. Temporal analysis of semantic graphs using ASALSAN. *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining-Volume 00*. IEEE Computer Society Washington, DC, USA, 33–42. 83, 85

Baeza-Yates, Ricardo A., Berthier A. Ribeiro-Neto. 1999. *Modern Information Retrieval*. ACM Press / Addison-Wesley. URL citeseer.ist.psu.edu/baeza-yates99modern.html. 65

Banerjee, J., A.; Ghosh. 2003. Competitive learning mechanisms for scalable, incremental and balanced clustering of streaming texts. *Proceedings of the International Joint Conference on, Neural Networks*, vol. 4. 2697– 2702. 17

Becerra-Fernandez, I. 2006. Searching for experts on the Web: A review of contemporary expertise locator systems. *ACM Transactions on Internet Technology (TOIT)* **6**(4) 333–355. 82

Billsus, D., M.J. Pazzani. 1998. Learning collaborative information filters. *Proceedings of the Fifteenth International Conference on Machine Learning*, vol. 54. 48

Bonacich, P. 1987. Power and centrality: A family of measures. *American Journal of Sociology* 1170–1182. 82, 106

Bonacich, P., P. Lloyd. 2001. Eigenvector-like measures of centrality for asymmetric relations. *Social Networks* **23**(3) 191–201. 82

Bookstein, Abraham, Don R. Swanson. 1975. A decision theoretic foundation for indexing. *Journal of the American Society for Information Science* 45–50. 24

Borman, W. C. 1979. Format and training effects on rating accuracy. *Journal of Applied Psychology* **64** 410–421. 50, 117

Brass, D.J. 1992. Power in organizations: A social network perspective. *The Political Consequences of Social Networks: The Political Consequences of Social Networks: 1992* **4** 295–323. 82

Breese, J.S., D. Heckerman, C. Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. Tech. rep., Microsoft Research. 48, 62

Brin, S., L. Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Seventh International World-Wide Web Conference (WWW 1998)*. URL http://ilpubs.stanford.edu:8090/361/. 84

Burt, R.S., T. Schøtt. 1985. Relation contents in multiple networks. *Social Science Research* **14**(4) 1985. 82

Chakrabarti, Soumen. 2002. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman. URL http://www.cse.iitb.ac.in/~soumen/mining-the-web/. 15

Cheeseman, P., J. Stutz. 1996. Bayesian classification (AUTOCLASS): Theory and results. *Advances in Knowledge Discovery and Data Mining* . 16

Chi, Yun, Belle L. Tseng, Junichi Tatemura. 2006. Eigen-trend: trend analysis in the blogosphere based on singular value decompositions. *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*. ACM, New York, NY, USA, 68–77. doi:http://doi.acm.org/10.1145/1183614.1183628. 83

Chi, Yun, Shenghuo Zhu, Xiaodan Song, Junichi Tatemura, Belle L. Tseng. 2007. Structural and temporal analysis of the blogosphere through community factorization. *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, 163–172. doi:http://doi.acm.org/10.1145/1281192.1281213. 83

Chien, Y.H., E.I. George. 1999. A bayesian model for collaborative filtering. *Proceedings of the 7 thInternational Workshop on Artificial Intelligence and Statistics* . 48

Chow, C. K., C. N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* **14**(3) 462–467. 55

Cooper, William H. 1981. Ubiquitous halo. *Psychological Bulletin* **90** 218–244. 50, 114, 115, 116, 117, 121

Cutting, Douglass R., David R. Karger, Pedersen Pedersen, John W. Tukey. 1992. Scatter/gather: A cluster-based approach to browsing large document collections. *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Interface Design and Display, 318–329. 12, 14

D. Kafry, S. Zedeck, R. Jacobs. 1979. Discriminability in multidimensional performance evaluations. *Applied psychological measurement* **3** 187–192. 53, 116

De Lathauwer, L., B. De Moor, J. Vandewalle. 2000. On the Best Rank-1 and Rank-(R, R,..., R) Approximation of Higher-Order Tensors. *SIAM Journal on Matrix Analysis and Applications* **21** 1324. 87

Deerwester, S., S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* **41**(6) 391–407. 85

Dempster, A. P., N. M. Laird, D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* **39** 1–38. 56

Doddington, George, Jaime Carbonell, James Allan, Jonathan Yamron, Umass Amherst, Yiming Yang. 2000. Topic detection and tracking pilot study final report. 13, 16, 82

E. F. Borgatta, J. H. Mann, L. S. Cottrell. 1958. The spectrum of individual interaction characteristics: An interdimensional analysis. *Psychological Reports* **4** 279–319. 50, 117

Elaine D. Pulakos, Neal Schmitt, Cheri Ostroff. 1986. A warning about the use of a standard deviation across dimensions within ratees to measure halo. *Journal of Applied Psychology* **1** 29–32. 116

Feeley, Thomas Hugh. 2002. Comment on halo effects in rating and evaluation research. *Human Communication Research* **28**(4) 578–586. 72

Figueiredo, M. A. T., A. K. Jain. 2002. Unsupervised learning of finite mixture models. *IEEE Trans. on Patt. Analysis and Machine Intell.* **24**(3) 381–396. 16

Fisher, Douglas H. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* **2** 139–172. 18, 22

Fisicaro, Charles E., Sebastiano A.; Lance. 1990. Implications of three causal models for the measurement of halo error. *Applied Psychological Measurement* **14**(4). 116

Fisicaro, Sebastiano A. 1988. A reexamination of the relation between halo error and accuracy. *Journal of Applied Psychology* **73** 239–244. 50, 114

Franz, Martin, Todd Ward, J. Scott McCarley, Wei-Jing Zhu. 2001. Unsupervised and supervised clustering for topic tracking. *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM Press, 310–317. doi:http://doi.acm.org/10.1145/383952.384013. 13, 16, 82

Frederick Mosteller, David L. Wallace. 1983. *Applied Bayesian and Classical Inference The case of The Federalist Papers*. Springer series in Statistics, Springer-Verlag. 24

G. P. Latham, E. D. Pursell, K. N. Wexley. 1980. Training managers to minimize rating errors in observation of behavior. *Journal of Applied Psychology* **60** 550–555. 50, 117

Gennari, J. H., P. Langley, D. Fisher. 1989. Models of incremental concept formation. *Journal of Artificial Intelligence* **40** 11–61. 18, 22

Getoor, L., M. Sahami. 1999. Using probabilistic relational models for collaborative filtering. *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)* . 48

Harshman, R.A. 1970. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics* **16**(1) 84. 87

Heckerman, D., D.M. Chickering, C. Meek, R. Rounthwaite, C. Kadie. 2001. Dependency networks for inference, collaborative filtering, and data visualization. *The Journal of Machine Learning Research* **1** 49–75. 48

Heneman, H. G. 1974. Comparision of self and superior ratings of managerial performance. *Journal of Applied Psychology* **59** 638–642. 50, 116

Herlocker, Jonathan L., Joseph A. Konstan, Loren G. Terveen, John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* **22**(1) 5–53. doi:http://doi.acm.org/10.1145/963770.963772. 64, 67

Hofmann, T. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)* **22**(1) 89–115. 48

Hofmann, Thomas, Jan Puzicha. 1999. Latent class models for collaborative filtering. Dean Thomas, ed., *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99-Vol2)*. Morgan Kaufmann Publishers, S.F., 688–693. 48

Holzbach, R. L. 1978. Rater bias in performance ratings: Superior, self, and peer ratings. *Journal of Applied Psychology* **63** 579–588. 50, 55, 56, 117, 119, 120

Ivancevich, J. M. 1979. Longtudinal study of the effects of rater training on psychometric error in ratings. *Journal of Applied Psychology* **64** 502–508. 50, 117

J. S. Berman, D. A. Kenny. 1976. Correlational bias in observer ratings. *Journal of Personality and Social Psychology* **34** 263–273. 117

J. T. Lamiell, P. Cavenee, M. A. Foss. 1980. On the relationship between conceptual schemes of and behavior reports: A closer report. *Journal of Personality* **48** 54–73. 116

Jain, A. K., M. N. Murty, P. J. Flynn. 1999. Data clustering: a review. *ACM Computing Surveys* **31**(3) 264–323. URL citeseer.ist.psu.edu/jain99data.html. 12, 14, 16

Katz, Slava M. 1996. Distribution of content words and phrases in text and language modelling. *Nat. Lang. Eng.* **2**(1) 15–59. doi:http://dx.doi.org/10.1017/S1351324996001246. 23, 24, 25, 43

Kautz, H., B. Selman. 1998. Creating models of real-world communities with ReferralWeb. *Working notes of the Workshop on Recommender Systems, held in conjunction with AAAI-98, Madison, WI*. 82

Kevin R. Murphy, Douglas H. Reynolds. 1988. Does true halo affect observed halo? *Journal of Applied Psychology* **73** 235–238. 50, 115

Kleinberg, Jon M. 1999. Authoritative sources in a hyperlinked environment. *J. ACM* **46**(5) 604–632. doi:http://doi.acm.org/10.1145/324133.324140. 84, 89

Kolda, Tamara G. 2006. Multilinear operators for higher-order decompositions. Tech. Rep. SAND2006-2081, Sandia National Laboratories, Albuquerque, NM and Livermore, CA. URL http://www.prod.sandia.gov/cgi-bin/techlib/access-control.pl/2006/062081.pdf. 86

Kolda, Tamara G., Brett W. Bader. 2008. Tensor decompositions and applications. *SIAM Review* To appear (accepted June 2008). 86, 87

Kolda, TG, BW Bader. 2006. The TOPHITS model for higher-order web link analysis. *Workshop on Link Analysis, Counterterrorism and Security*. 83, 85, 89

Koller, Daphne, Nir Friedman. 2009. *Structured Probabilistic Models: Principles and Techniques*. MIT Press. To appear. 48, 54, 61

Koltuv, B. B. 1962. Some characteristics of intrajudge trait intercorrelations. *Psychological Monograph* **76**. 50, 116

Krulwich, B., C. Burkey, A. Consulting. 1996. The ContactFinder agent: Answering bulletin board questions with referrals. *Proceedings of the National Conference on Artificial Intelligence*. 10–15. 82

Kullback, S., R. A. Leibler. 1951. On information and sufficiency. *Ann. Math. Statistics* **22** 79–86. 101

Landy, F. J., J. L. Farr. 1980. Performance rating. *Psychological Bulletin* **87** 72–107. 50, 116

Lee, H.H., W.G. Teng. 2007. Incorporating Multi-Criteria Ratings in Recommendation Systems. *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on* 273–278. 50

Lewis, David D., Yiming Yang, Tony G. Rose, Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research* **5** 361–397. 35, 38

Liu, Xiaoyong, W. Bruce Croft. 2004. Cluster-based retrieval using language models. *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Language models, 186–193. URL http://doi.acm.org/10.1145/1008992.1009026. 12, 16

M. J. Kavanagh, L. Wolins, A. C. MacKinney. 1971. Issues oin managerial performance: Multitrait-multimethod analyses of ratings. *Pshychological Bulletin* **75** 34–49. 116

MacKay, D.J.C. 2003. *Information theory, inference, and learning algorithms*. Cambridge University Press New York. 55

Manning, Christopher D., Hinrich Schütze. 2000. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, England. 14, 15, 16, 23, 24, 25

McCallum, Andrew, Andres Corrada-Emmanuel, Xuerui Wang. 2004. The author-recipient-topic model for topic and role discovery in social networks: Experiments with enron and academic email. *NIPS'04 Workshop on'Structured Data and Representations in Probabilistic Models for Categorization*. 82

Mitchell, Tom M. 1997. *Machine Learning*, chap. 3. WCB/McGraw-Hill, 67. 75

Moon, Sangkil, Gary J. Russell. 2008. Predicting Product Purchase from Inferred Customer Similarity: An Autologistic Model Approach. *MANAGEMENT SCIENCE* **54**(1) 71–82. doi:10.1287/mnsc.1070.0760. URL http://mansci.journal.informs.org/cgi/content/abstract/54/1/71. 48

Morrison, Donald F. 1967. *Multivariate Statistical Methods*. McGraw-Hill Book Company. 53, 119, 120

Murphy, Rebecca L, Kevin R.; Anhalt. 1992. Is halo error a property of the rater, ratees, or the specific behaviors observed? *Journal of Applied Psychology* **77**(4) 494–500. 115

Myers, James H. 1965. Removing halo from job evaluation factor structure. *Journal of Applied Psychology* **49** 217–221. 50, 55, 56, 118

Netflix, Inc. 2006. Form 10-k annual report pursuant to section 13 or 15(d) of the securities exchange act of 1934. UNITED STATES SECURITIES AND EXCHANGE COMMISSION, Washington, D.C. 20549. 46

Nir Friedman, Moises Goldszmidt. 1998. *Learning in Graphical Models*, chap. 15. Kluwer Academic Publishers, 431–432. 61

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press. 48

Qamra, Arun, Belle Tseng, Edward Y. Chang. 2006. Mining blog stories using community-based and temporal clustering. *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*. ACM, New York, NY, USA, 58–67. doi:http://doi.acm.org/10.1145/1183614.1183627. 82

Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. *Proceedings of the Conference on Computer-Supported Cooperative Work, CSCW'94*. 48

Rizzo, W. A., F. D. Frank. 1977. Influence of irrelevant cues and alternate forms of graphic rating scales on the halo effect. *Personnel Psychology* **30** 405–417. 50, 116

Sarkar, P., A.W. Moore. 2005. Dynamic social network analysis using latent space models. *ACM SIGKDD Explorations Newsletter* **7**(2) 31–40. 83

Shardanand, Upendra, Pattie Maes. 1995. Social information filtering: Algorithms for automating word of mouth. *CHI*. 210–217. 47

Shweder, R. A. 1975. How relevant is an individual difference in personality. *Journal of Personality* **43** 455–484. 50, 115

Shweder, R. A., R. G. D'Andrade. 1980. The systematic distortion hypothesis. *New directions for methodology of behavioral science: Fallible judgment in behavioral research* 37–58. 50, 117

Si, Luo, Rong Jin. 2003. Flexible mixture model for collaborative filtering. *ICML*. AAAI Press, 704–711. 47, 48, 50, 53, 54, 60

Smyth, Padhraic. 1996. Clustering Using Monte Carlo Cross-Validation. Evangelos Simoudis, Jia Wei Han, Usama Fayyad, eds., *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press, 126. 15

Steele, F.J. Landy; R.J. Vance; J.L. Barnes-Farrell; J.W. 1980. Statistical control of halo error in performance ratings. *Journal of applied psychology* **65** 501–506. 50, 55, 56, 118

Streeter, L.A., K.E. Lochbaum. 1988a. An expert/expert locating system based on automatic representation of semantic structure. *Proceedings of the fourth conference on artificial intelligence applications*. 345–350. 82

Streeter, L.A., K.E. Lochbaum. 1988b. Who knows: A system based on automatic representation of semantic structure. *RIAO*, vol. 88. 380–388. 82

Swan, R., D. Jensen. 2000. Timemines: Constructing timelines with statistical models of word usage. *KDD-2000 Workshop on Text Mining*. Citeseer, 73–80. 82

Thorndike, EL. 1920. A constant error in psychological ratings. *Journal of Applied Psychology* **4** 25–29. 50, 114, 115

Ungar, L.H., D.P. Foster. 1998. Clustering methods for collaborative filtering. *AAAI Workshop on Recommendation Systems* 112–125. 48

Vivacqua, A. 1999. Agents for expertise location. *Proc. 1999 AAAI Spring Symposium Workshop on Intelligent Agents in Cyberspace*. 9–13. 82

Wang, Xuerui, Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, 424–433. doi:http://doi.acm.org/10.1145/1150402.1150450. 82

Wang, Xuerui, Natasha Mohanty, Andrew McCallum. 2005. Group and topic discovery from relations and text. *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*. ACM, New York, NY, USA, 28–35. doi:http://doi.acm.org/10.1145/1134271.1134276. 83

Wasserman, S., K. Faust. 1994. *Social network analysis: Methods and applications*. Cambridge Univ Pr. 84

Wells, F. L. 1907. A statistical study of literary merit. *Archives of psychology* **1**. 50, 114, 115

William T. Hoyt, Michael-David Kerns. 1999. Magnitude and moderators of bias in observer ratings a meta-analysis. *Psychological Methods* **4** 403–424. 120

Yang, Yiming, Tom Pierce, Jaime Carbonell. 1998. A study of retrospective and on-line event detection. *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM Press, 28–36. doi:http://doi.acm.org/10.1145/290941.290953. 13, 16, 82

Yimam, D. 2000. Expert Finding Systems for Organizations: Domain Analysis and the DEMOIR approach. *ECSCW 99 Beyond Knowledge Management: Management Expertise Workshop*. MIT Press, 276–283. 82

Zhang, Ya-Jun, Zhi-Qiang Liu. 2004. Refining web search engine results using incremental clustering. *International journal of intelligent systems* **19** 191–199. 17

Zhu, Shenghuo, Kai Yu, Yun Chi, Yihong Gong. 2007. Combining content and link for classification using matrix factorization. *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 487–494. doi:http://doi.acm.org/10.1145/1277741.1277825. 83