DOCTORAL DISSERTATION PROPOSAL

THREE ESSAYS ON ENTERPRISE INFORMATION SYSTEM MINING FOR BUSINESS INTELLIGENCE

A dissertation proposal 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

February 2, 2009
Abstract

This dissertation proposal 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 document 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. It encodes the text of the conversation and the participating actors in a tensor. With the help of blog data collected from a large IT services firm it shows that by tensor factorization we are able to identify significant topics of conversation as well as the important actors in each. In addition it proposes three extensions to this study: 1) Evaluation of the tensor factorization approach by measuring its accuracy in topic discovery and community discovery, 2) Extension of the study by incorporating the blog reading data which is unique because it measures consumption of post topics, and 3) Study the interdependence of reading, posting, citation activity at a blog social network.
# Table of Contents

Abstract .................................................. 2

Table of Contents ........................................ 3

List of Tables ............................................ 5

List of Figures ........................................... 6

1 Introduction ............................................. 7

2 Discovering topic hierarchy in document streams .......... 10
   2.1 Introduction ........................................ 10
   2.1.1 Contribution of this research .................... 11
   2.2 Literature review ................................... 12
   2.3 Text Documents and word distributions ............. 21
      2.3.1 Models based on Poisson distribution .......... 21
      2.3.2 Katz’s K-mixture model ......................... 24
      2.3.3 Fitness comparison .............................. 26
   2.4 Algorithms for text .................................. 27
      2.4.1 COBWEB: when attribute values follow Katz’s distribution 27
      2.4.2 COBWEB: when attribute values follow Negative Binomial distribution 28
   2.5 Cluster Evaluation Methods ........................... 29
      2.5.1 Evaluating the clusters ......................... 29
      2.5.2 Evaluating the hierarchy ....................... 31
   2.6 Experiment setup and results ......................... 32
      2.6.1 Reuters-RCV1 .................................... 32
      2.6.2 OHSUMED (88-91) ............................... 36
   2.7 Conclusion ........................................... 41

3 Multi-component Rating Collaborative Filtering .......... 43
   3.1 Introduction ......................................... 43
   3.2 Multi-component rating recommender system .......... 48
      3.2.1 Data description and preliminary analysis .... 49
      3.2.2 Modeling component ratings for collaborative filtering 50
      3.2.3 Parallels ......................................... 53
      3.2.4 Model estimation using EM algorithm .......... 54
   3.3 Results and discussion ................................ 58
      3.3.1 Predicting the Overall rating ................. 58
      3.3.2 Filling-in missing component ratings .......... 64
   3.4 Conclusions .......................................... 71
TABLE OF CONTENTS

4 Socio-temporal analysis of conversations in intra-organizational blogs 73
   4.1 Introduction ......................................................... 73
   4.1.1 Objective ....................................................... 75
   4.2 Theme Identification ................................................ 75
       4.2.1 Background ................................................... 75
       4.2.2 Blog topic development ....................................... 75
       4.2.3 Blog conversation development .............................. 77
       4.2.4 Comparison with the existing methods ....................... 77
       4.2.5 Data description and representation .......................... 77
   4.3 Results and Discussion ............................................... 78
   4.4 Proposed next steps ................................................ 80
       4.4.1 Task based evaluation of tensor factors ....................... 80
       4.4.2 Analysis of blog Reading behaviour .......................... 81
A Document Clustering .................................................... 83
   A.1 MLE of Katz’s distribution parameters ............................ 83
B Multi-component Rating Collaborative Filtering ....................... 85
   B.1 Derivation of marginal distributions .............................. 85
       B.1.1 $P(U, I, O)$ for the model with dependency among the ratings 85
       B.1.2 $P(U, I, O)$ for the model with independent component ratings 86
       B.1.3 $P(U, I, O)$ for the model with only the overall ratings .......... 86
       B.1.4 $P(U, I, S, O)$ from the complete joint distribution ............. 87
   B.2 Halo in multi-criteria movie rating ................................ 87
       B.2.1 Halo Effect ..................................................... 87
       B.2.2 Halo in movie rating data .................................... 91
References ......................................................................... 95
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Likelihood comparison</td>
<td>27</td>
</tr>
<tr>
<td>2.2</td>
<td>RCV1 dataset</td>
<td>33</td>
</tr>
<tr>
<td>2.3</td>
<td>Cluster quality comparison on RCV1 data</td>
<td>35</td>
</tr>
<tr>
<td>2.4</td>
<td>Evaluation of the cluster hierarchy using RCV1 data</td>
<td>36</td>
</tr>
<tr>
<td>2.5</td>
<td>OHSUMED dataset (88-91)</td>
<td>37</td>
</tr>
<tr>
<td>2.6</td>
<td>Cluster quality comparison on OHSUMED dataset</td>
<td>38</td>
</tr>
<tr>
<td>2.7</td>
<td>Evaluation of the cluster hierarchy using OHSUMED data</td>
<td>39</td>
</tr>
<tr>
<td>3.1</td>
<td>Examples of Collaborative Filtering based recommender systems</td>
<td>44</td>
</tr>
<tr>
<td>3.2</td>
<td>An example of multi-component rating. Ratings are on a scale of 0 to 4.</td>
<td>48</td>
</tr>
<tr>
<td>3.3</td>
<td>Correlation among components of rating</td>
<td>50</td>
</tr>
<tr>
<td>3.4</td>
<td>Principal components</td>
<td>50</td>
</tr>
<tr>
<td>3.5</td>
<td>Factor loadings after quartimax rotation</td>
<td>51</td>
</tr>
<tr>
<td>3.6</td>
<td>Partial correlations controlling for Overall rating</td>
<td>53</td>
</tr>
<tr>
<td>3.7</td>
<td>Increase in dataset size after filling in missing components</td>
<td>65</td>
</tr>
<tr>
<td>3.8</td>
<td>Comparison of different methods filling in missing rating components</td>
<td>68</td>
</tr>
<tr>
<td>4.1</td>
<td>Topics of conversation</td>
<td>74</td>
</tr>
<tr>
<td>4.2</td>
<td>Blog data description</td>
<td>78</td>
</tr>
<tr>
<td>B.1</td>
<td>Correlation among rating components</td>
<td>91</td>
</tr>
<tr>
<td>B.2</td>
<td>Partial correlation given Overall. $R_i, R_j \in {S, A, V, D}$</td>
<td>91</td>
</tr>
<tr>
<td>B.3</td>
<td>Eigen values of the correlation matrix</td>
<td>92</td>
</tr>
<tr>
<td>B.4</td>
<td>Eigen values of the partial correlation matrix</td>
<td>93</td>
</tr>
<tr>
<td>B.5</td>
<td>Factor loadings after quartimax rotation</td>
<td>93</td>
</tr>
<tr>
<td>B.6</td>
<td>Principal component analysis of rating components</td>
<td>94</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>COBWEB control structure</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>COBWEB illustrated</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>COBWEB: After first two items are added</td>
<td>18</td>
</tr>
<tr>
<td>2.4</td>
<td>COBWEB: Two partitions of the root cluster</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Merge and split operations illustrated</td>
<td>20</td>
</tr>
<tr>
<td>2.6</td>
<td>Word distribution</td>
<td>23</td>
</tr>
<tr>
<td>2.7</td>
<td>Hierarchical evaluation illustrated</td>
<td>31</td>
</tr>
<tr>
<td>2.8</td>
<td>Three classification hierarchies</td>
<td>33</td>
</tr>
<tr>
<td>2.9</td>
<td>Cluster quality comparison on RCV1 data</td>
<td>34</td>
</tr>
<tr>
<td>2.10</td>
<td>MeSH labels file</td>
<td>39</td>
</tr>
<tr>
<td>2.11</td>
<td>Sibling precision: using 25 and 75 words</td>
<td>40</td>
</tr>
<tr>
<td>2.12</td>
<td>Sibling precision: using 125 and 175</td>
<td>40</td>
</tr>
<tr>
<td>3.1</td>
<td>Aspect model</td>
<td>47</td>
</tr>
<tr>
<td>3.2</td>
<td>Flexible mixture model</td>
<td>47</td>
</tr>
<tr>
<td>3.3</td>
<td>Skewed user distribution</td>
<td>49</td>
</tr>
<tr>
<td>3.4</td>
<td>Extreme ratings are more frequent</td>
<td>50</td>
</tr>
<tr>
<td>3.5</td>
<td>Flexible Mixture model for component rating collaborative filtering</td>
<td>51</td>
</tr>
<tr>
<td>3.6</td>
<td>Discovered structure in the sub ratings</td>
<td>52</td>
</tr>
<tr>
<td>3.7</td>
<td>Flexible Mixture Model with component rating dependency structure</td>
<td>54</td>
</tr>
<tr>
<td>3.8</td>
<td>Flexible mixture model with independent component ratings</td>
<td>56</td>
</tr>
<tr>
<td>3.9</td>
<td>Plot of errors by amount of training data used, for different models</td>
<td>61</td>
</tr>
<tr>
<td>3.10</td>
<td>Precision-Recall curve for three methods</td>
<td>62</td>
</tr>
<tr>
<td>3.11</td>
<td>Rating prediction mean absolute error</td>
<td>64</td>
</tr>
<tr>
<td>3.12</td>
<td>Three algorithms compared by using data in time order for training</td>
<td>65</td>
</tr>
<tr>
<td>3.13</td>
<td>Precision-Recall curves using prior data for training and later data for testing</td>
<td>66</td>
</tr>
<tr>
<td>3.14</td>
<td>Records with partial information</td>
<td>67</td>
</tr>
<tr>
<td>3.15</td>
<td>Error distributions</td>
<td>68</td>
</tr>
<tr>
<td>3.16</td>
<td>Accuracy of collaborative filtering after filling in missing components</td>
<td>69</td>
</tr>
<tr>
<td>3.17</td>
<td>Precision recall curves for filled in data</td>
<td>70</td>
</tr>
<tr>
<td>4.1</td>
<td>Trends of topics and important bloggers in each</td>
<td>79</td>
</tr>
<tr>
<td>4.2</td>
<td>of topics and topic specific hub and authority score</td>
<td>79</td>
</tr>
</tbody>
</table>
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 that use only one component.

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, methods developed and results obtained so far in the intra-organizational blog data analysis. It also outlines the proposed next steps towards the completion of the dissertation.
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 \textsc{Cobweb} that has been used in non-text domain and its variant \textsc{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 paper we demonstrate methods to carry out incremental hierarchical clustering of text documents. Specifically, the contributions of this work are:

1. A \textsc{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 Schütze (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 \( O(n^2) \). 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 \( \rho m \) 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 \( O(mn) \). 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 online 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 document to match an older query. When a new document $d_j$ is compared to a past query $q_i$, the threshold is computed as $0.4 + p \times \left( \text{sim}(q_i, d_i) - 0.4 \right) + tp \times (j - i)$, 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 $q_i$s. 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
2.2. LITERATURE REVIEW
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.

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

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

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

$V_{ij} = j$th value of the $i$th attribute (say, $V_{12}$ indicates “female”),

$P(C_k)$ = the probability of a document belonging to cluster $k$, given that it belongs to the parent cluster $p$.

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 0.25 fraction of the time and C 0.04 fraction of the time. A good cluster, in which the attributes of the items take similar values, will have high $P(A_i = V_{ij} \mid C_k)$ 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
2.2. LITERATURE REVIEW

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.

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 \( C_2 \).

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...
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](c45b7c76.png)

Figure 2.2: COBWEB illustrated

\[
C_3 \quad P(C_3) = 1 \\
(\text{a and b}) t = A, t = B
\]

\[
C_1 \quad P(C_1) = 0.5 \\
(\text{a}) t = A
\]

\[
C_2 \quad P(C_2) = 0.5 \\
(\text{b}) t = B
\]

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

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 clusters are \( C_1, C_2 \) and \( C_4 \). The category utility of this configuration is:

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

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

\[
\text{CU}^2 = \sum_{k=\{4,2\}} P(C_k) \left[ \sum_{A_t} \sum_{t=\{A,B,C\}} P(t|C_k)^2 - \sum_{A_t} \sum_{t=\{A,B,C\}} P(t|C_3)^2 \right]
\]

or

\[
\text{CU}^1 = \sum_{k=\{4,2\}} P(C_k) \left[ \sum_{A_t} \sum_{t=\{A,B,C\}} P(t|C_k)^2 - \sum_{A_t} \sum_{t=\{A,B,C\}} P(t|C_3)^2 \right]
\]

\[
\text{CU}^2 = \frac{1}{2} \left[ \frac{2}{3} \left\{ \left( \frac{1}{2} \right)^2 + \left( \frac{1}{2} \right)^2 \right\} - \left( \frac{1}{3} \right)^2 + \left( \frac{1}{3} \right)^2 + \left( \frac{1}{3} \right)^2 \right] + \frac{1}{3} \left( 1^2 - \left( \frac{1}{3} \right)^2 + \left( \frac{1}{3} \right)^2 + \left( \frac{1}{3} \right)^2 \right)
\]

\[
\text{CU}^2 = \frac{1}{6}
\]

Since, $\text{CU}^1 > \text{CU}^2$ 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

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

where,

- \(\sigma_{ip}\) = standard deviation of the value of the attribute \(i\) in parent node \(p\), and
- \(\sigma_{ik}\) = 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 distri-
butional 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 Schütze have discussed several models to characterize the occurrence of words across different documents (Manning and Schütze 2000).

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

\[
P(k) = \frac{\lambda^k e^{-\lambda}}{k!}
\]

(2.2)

where, \(\lambda\) 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

\(^1\)Through out this paper 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”.


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:

\[ P(k) = \alpha \frac{\lambda_1^k e^{-\lambda_1}}{k!} + (1 - \alpha) \frac{\lambda_2^k e^{-\lambda_2}}{k!} \]  

(2.3)

where, \( \alpha \) 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,

\[ P(k) = \binom{k + r - 1}{r - 1} p^r (1 - p)^k, \]  

(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.
Figure 2.6: The occurrence of a typical word ("result") across different documents in our test collection.
The probability density function of Zero Inflated Poisson distribution is given by

\[ P(k) = (1 - \alpha)\delta_k + \alpha \frac{\lambda^k e^{\lambda}}{k!}, \quad k = 0, 1, 2 \ldots \]  

(2.5)

where,

\[ \delta_k = \begin{cases} 1, & \text{if} \ k = 0 \\ 0, & \text{otherwise} \end{cases} \]

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\(^2\).

\[ P(k) = (1 - \alpha)\delta_k + \alpha \left( \frac{\beta}{\beta + 1} \right)^k \]  

(2.6)

\[ \delta_k = 1 \text{ if } k = 0 \text{ and } 0 \text{ otherwise}. \]

The MLE estimates of parameters \( \alpha \) and \( \beta \) are:

\[ \beta = \frac{cf - df}{df} \]  

(2.7)

\[ \alpha = \frac{1}{\beta} \times \frac{cf}{N} \]  

(2.8)

\( cf = \text{collection frequency} = \) 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.

\( df = \text{document frequency} = \) number of documents in the entire collection that contain the word \( i \).

From (2.6) it follows that

\[ P(0) = 1 - \alpha + \frac{\alpha}{\beta + 1} \]

\[ = 1 - \frac{df}{N} \]  

(2.9)

\[ = 1 - \Pr(\text{the word occurs in a document}) \]

\[ = \Pr(\text{the word does not occur in a document}) \]

\(^2\)In this section we shall discuss the case of one word, the \( i \)th word. Hence, we shall drop the subscript \( i \) from the equations and expressions.
Also, it follows that

\[ P(k) = \frac{\alpha}{\beta + 1} \left( \frac{\beta}{\beta + 1} \right)^k, \quad k = 1, 2, \ldots \]  

(2.10)

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

\[ P(k) = \alpha(1 - p)p^k \]  

(2.11)

Let’s define a parameter \( p_0 \) as

\[ p_0 = P(0) \]  

(2.12)

using (2.7) we find that

\[ p = \frac{cf - df}{df} = \frac{cf - df}{cf} \]  

(2.13)

\begin{align*}
&= \frac{\Pr(\text{the word repeats in a document})}{\Pr(\text{the word occurs in a document})} \\
&= \frac{\Pr(\text{the word repeats} \cap \text{the word occurs})}{\Pr(\text{the word occurs})} \\
&= \Pr(\text{the word repeats} \mid \text{the word occurs})
\end{align*}

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

\[ P(0) = (1 - \alpha) + \alpha(1 - p) \]  

\[ = 1 - \alpha + \alpha - \alpha p \]

Hence, \( \alpha \) in terms of \( p_0 \) and \( p \) is

\begin{align*}
p_0 &= 1 - \alpha p \\
\Rightarrow \alpha p &= 1 - p_0 \\
\Rightarrow \alpha &= \frac{1 - p_0}{p}
\end{align*}  

(2.14)

Expression (2.11) can now be written as

\[ P(k) = (1 - p_0)(1 - p)p^{k-1} \]  

(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)

\[
\hat{p}_0 = 1 - \frac{df}{N} \tag{2.16}
\]

and

\[
\hat{p} = \frac{cf - df}{cf} \tag{2.17}
\]

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 \( \alpha \) and \( \lambda \) are

\[
\hat{\lambda} = \frac{\text{Var}(X)}{\bar{X}} + \bar{X} - 1 \tag{2.18}
\]

and

\[
\hat{\alpha} = \frac{\bar{X}}{\lambda} \tag{2.19}
\]

**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

$$
\frac{1}{K} \sum_k P(C_k) \left[ \sum_i \sum_j (P(A_i = V_{i,j}|C_k)^2 - P(A_i = V_{i,j}|C_p)^2) \right]
$$
Notice that for each attribute indexed $i$ we need to compute
\[ \sum_j \left( P(A_i = V_{i,j}|C_k)^2 - P(A_i = V_{i,j}|C_p)^2 \right) \]  
(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
\[ CU_{i,k} = \sum_{f=0}^{\infty} P(A_i = f|C_k)^2 \]  
(2.23)
Let’s use $CU_{i,k}$ 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
\[ CU_{i,k} = \sum_{f=0}^{\infty} P(A_i = f|C_k)^2 = \frac{1 - 2p_0(1 - p_0) - p(1 - 2p_0)}{1 + p} \]  
(2.24)
Substituting estimates of $p_0$ and $p$ from Expressions (2.16) and (2.17) in Expression (2.24), and simplifying, we get
\[ CU_{i,k} = \sum_{f=0}^{\infty} P(A_i = f|C_k)^2 = 1 - \frac{2 \times df}{N - \frac{cf \times df}{2 \times cf - df}} \]  
(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
\[ CU_p = \frac{1}{K} \sum_k P(C_k) \left[ \sum_i CU_{i,k} - \sum_i CU_{i,p} \right] \]  
(2.26)
where, $CU_{i,k}$ 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
\[ P(x) = \binom{x + r - 1}{r - 1} p^r (1 - p)^x \]  
(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

\[
CU_{i,k} = \sum_{x=0}^{\infty} \left( \frac{(x + r - 1)!}{x!(r - 1)!} \right) p^r (1 - p)^{x-1} \left. \right)^2
\]  
(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.

\[
CU_{i,k} = \frac{p^2 r^2 F_1 (r, r, 1, (1 - p)^2)}{(1 - p)^2}
\]  
(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

\[
p_k = \max_c \left\{ CF_k(c) \right\}
\]

(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

\( CF_k(c) = \) 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, \( \max_{c} \{ CF_k(c) \} \) 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.

\[
r_c = \frac{\max_{k} \{ CF_k(c) \}}{N_c}
\]

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 \( P \) and \( R \) values:

\[
\frac{1}{F} = \frac{1}{2} \left( \frac{1}{P} + \frac{1}{R} \right)
\]  (2.31)
2.5. CLUSTER EVALUATION METHODS

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

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
Class & Frequency \\
\hline
$C_1$ & 2 \\
$C_2$ & 1 \\
$C_{1,1}$ & 1 \\
\hline
\end{tabular}
\end{table}

\(^3\)At the lowest level each cluster has only one document and its class can be read from the data directly.
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

\(^4\)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 \textit{split} and \textit{merge} operators.
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.
2.6. EXPERIMENT SETUP AND RESULTS

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.

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 $cf \times \log (N/df)$ 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 our 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
## 2.6. EXPERIMENT SETUP AND RESULTS

<table>
<thead>
<tr>
<th>V</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K</td>
<td>N</td>
<td>K</td>
<td>N</td>
<td>K</td>
<td>N</td>
<td>K</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>micro</td>
<td>0.46</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>macro</td>
<td>0.31</td>
<td>0.43</td>
<td>0.33</td>
<td>0.42</td>
<td>0.36</td>
<td>0.42</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>K</td>
<td>N</td>
<td>K</td>
<td>N</td>
<td>K</td>
<td>N</td>
<td>K</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>micro</td>
<td>0.83</td>
<td>0.81</td>
<td>0.85</td>
<td>0.79</td>
<td>0.84</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>macro</td>
<td>0.60</td>
<td>0.74</td>
<td>0.67</td>
<td>0.74</td>
<td>0.69</td>
<td>0.76</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>K</td>
<td>N</td>
<td>K</td>
<td>N</td>
<td>K</td>
<td>N</td>
<td>K</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>micro</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>macro</td>
<td>0.31</td>
<td>0.43</td>
<td>0.33</td>
<td>0.42</td>
<td>0.36</td>
<td>0.42</td>
<td>0.30</td>
</tr>
</tbody>
</table>

Table 2.3: Cluster quality comparison on RCV1 data
Table 2.4: Evaluation of the cluster hierarchy using RCV1 data

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

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.

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

\(^5\)This can be obtained from (Lewis et al. 2004) Appendix 2.
2.6. EXPERIMENT SETUP AND RESULTS

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

Table 2.5: OHSUMED dataset (88-91)

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 \( cf \times \log \left( \frac{N}{df} \right) \) 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.

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

\(^6\)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.
clear that the best K-nN-Classist and the best Normal-Classist have been highlighted by gray cells. The cells where Normal-Classist performs better are marked with a √, and the cells where the K-nN-Classist performs better are marked with an ×. The K-nN-Classist figures in the table are F-score×100. K stands for Katz-Classist, N for the original Classist, and W in the smaller figures in the table are F-score×100. K stands for Katz-Classist, N for the original Classist, and W in the smaller figures in the table are F-score×100. K stands for Katz-Classist, N for the original Classist, and W in the smaller

Table 2.6: Cluster quality comparison on OSHMED data at different number of clusters (k) and vocabulary size (V).

<table>
<thead>
<tr>
<th>k</th>
<th>V</th>
<th>000</th>
<th>100</th>
<th>00</th>
<th>10</th>
<th>0</th>
<th>1</th>
<th>00</th>
<th>0</th>
<th>0</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>10</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>20</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>50</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>100</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
</tbody>
</table>

The table shows the F-score for different values of k and V. The best performance is highlighted in gray.
2.6. EXPERIMENT SETUP AND RESULTS

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

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

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

Table 2.7: Evaluation of the cluster hierarchy using OHSUMED data

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 existing 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.
2.6. EXPERIMENT SETUP AND RESULTS

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

Figure 2.12: Tracing the cluster sibling precision over the height of the tree. Vocabulary sizes 125 and 175.
when we look at the nodes occurring 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 performance 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. 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 we think we have made in 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 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. We evaluate the algorithm using data from Yahoo! Movies. Use of multiple components leads to significant improvements in recommendations when we have small amount of training data—a case very common in real life implementations. However, when there is more training data we gain little from using additional components of the ratings. 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.

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.
3.1. INTRODUCTION

<table>
<thead>
<tr>
<th>Item type</th>
<th>Commercial</th>
<th>Non Commercial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Music</td>
<td>iTunes, Last.fm, Yahoo! Music</td>
<td>iRATEradio.com, mystrands.com</td>
</tr>
<tr>
<td>Movies</td>
<td>Netflix.com, blockbuster.com</td>
<td>movielens.umn.edu, filmaffinity.com</td>
</tr>
<tr>
<td>Books</td>
<td>StoryCode.com</td>
<td>gnooks.com</td>
</tr>
<tr>
<td>Dating</td>
<td>reciprodate.com</td>
<td>GiveALink.org, StumbleUpon.com</td>
</tr>
<tr>
<td>Webpages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aggregated</td>
<td>Amazon.com, half.ebay.com</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Examples of Collaborative Filtering based recommender systems

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). The 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.
3.1. INTRODUCTION

Contributions of this paper

In this paper 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 train the model using multi-component rating data from Yahoo Movies and use the trained model to generate movie recommendations. The test results show a significant improvement from the use of multiple components when very little training is possible. 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, although the idea of multi-aspect rating collaborative filtering has been motivated in the literature before (Adomavicius and Tuzhilin 2005). 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 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 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
3.1. INTRODUCTION

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.
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.

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

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>story</th>
<th>acting</th>
<th>visuals</th>
<th>direction</th>
<th>overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>$m_1$</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$m_1$</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

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 make more accurate recommendations.
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

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 ($A^+, A^-, B^+, B^-, C^+, C^-, D^+, D^-, F$). We recoded them to a scale of 0 to 4 ($\{A^+, A^-, B^+, B^-, C^+, C^-, D^+, D^-, F\}$ → 4, $\{B^+, B^-, C^+, C^-, D^+, D^-, F\}$ → 3, $\{C^+, C^-, D^+, D^-, F\}$ → 2, $\{D^+, D^-, F\}$ → 1, $\{F\}$ → 0), so that there will be enough data points in each rating bucket. Ratings on a set of 5628 movies were collected. Although, there were 691,496 records 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 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.
3.2. MULTI-COMPONENT RATING RECOMMENDER SYSTEM

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

Table 3.3: Correlation among components of rating

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>A</th>
<th>D</th>
<th>V</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>1.00</td>
<td>0.79</td>
<td>0.82</td>
<td>0.74</td>
<td>0.87</td>
</tr>
<tr>
<td>A</td>
<td>0.79</td>
<td>1.00</td>
<td>0.81</td>
<td>0.73</td>
<td>0.83</td>
</tr>
<tr>
<td>D</td>
<td>0.82</td>
<td>0.81</td>
<td>1.00</td>
<td>0.79</td>
<td>0.88</td>
</tr>
<tr>
<td>V</td>
<td>0.74</td>
<td>0.73</td>
<td>0.79</td>
<td>1.00</td>
<td>0.80</td>
</tr>
<tr>
<td>O</td>
<td>0.87</td>
<td>0.83</td>
<td>0.88</td>
<td>0.80</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 3.4: Principal components

<table>
<thead>
<tr>
<th>Component</th>
<th>One</th>
<th>Two</th>
<th>Three</th>
<th>Four</th>
<th>Five</th>
</tr>
</thead>
<tbody>
<tr>
<td>% variance explained</td>
<td>84.5</td>
<td>5.7</td>
<td>4.4</td>
<td>3.2</td>
<td>2.2</td>
</tr>
</tbody>
</table>

Table 3.5: Principal components

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 removing the effect of the overall component tells us how the user evaluates aspects of the item. Traditionally, only the overall compo-
3.2. MULTI-COMPONENT RATING RECOMMENDER SYSTEM

<table>
<thead>
<tr>
<th>Factor 1</th>
<th>Factor 2</th>
<th>Uniquenesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0.91</td>
<td>0.23</td>
</tr>
<tr>
<td>A</td>
<td>0.87</td>
<td>-0.02</td>
</tr>
<tr>
<td>V</td>
<td>0.93</td>
<td>-0.08</td>
</tr>
<tr>
<td>D</td>
<td>0.84</td>
<td>-0.12</td>
</tr>
<tr>
<td>O</td>
<td>0.95</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Table 3.5: Factor loadings after quartimax rotation

Figure 3.5: Flexible Mixture model for component rating collaborative filtering

Component has been used to carry out collaborative filtering, e.g., in (Si and Jin 2003). In 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 independent assertion 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. 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.

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 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 in-

\(^{1}\)Mutual information is a metric to measure the strength of the dependence between two variables (MacKay 2003)
dependent. 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

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.

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.

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. MULTI-COMPONENT RATING RECOMMENDER SYSTEM

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 like the EM algorithm (Dempster et al. 1977). The EM algorithm has 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 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**

\[
P(Z_u, Z_i | \vec{X}) = \frac{P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i, Pa_{R_j}\right)}{\sum_{Z_u} \sum_{Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\prod_{j=1}^{5} P\left(R_j|Z_u, Z_i, Pa_{R_j}\right)} \tag{3.1}
\]
3.2. MULTI-COMPONENT RATING RECOMMENDER SYSTEM

\textbf{M-step}

\begin{align*}
P(Z_u) &= \frac{1}{L} \sum_{l} \sum_{Z_i} P(Z_u, Z_i | \overrightarrow{X}(l)) \quad (3.2) \\
P(Z_i) &= \frac{1}{L} \sum_{l} \sum_{Z_u} P(Z_u, Z_i | \overrightarrow{X}(l)) \quad (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 | \overrightarrow{X}(l))} \quad (3.4) \\
P(I|Z_i) &= \frac{\sum_{l: I(l) = I} \sum_{Z_u} P(Z_u, Z_i | \overrightarrow{X}(l))}{\sum_{l} \sum_{Z_u} P(Z_u, Z_i | \overrightarrow{X}(l))} \quad (3.5) \\
P(R_j|Z_u, Z_i, \text{Pa}_{R_j}) &= \frac{\sum_{l: R_j(l) = R_j \& \text{Pa}_{R_j}(l) = \text{Pa}_{R_j}} P(Z_u, Z_i | \overrightarrow{X}(l))}{\sum_{l: \text{Pa}_{R_j}(l) = \text{Pa}_{R_j}} P(Z_u, Z_i | \overrightarrow{X}(l))} \quad (3.6)
\end{align*}

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. $R_j \in \{S, A, V, D, O\}$
- $\text{Pa}_{R_j}$ = 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 $l$
- $\text{Pa}_{R_j}(l)$ = rating variable $\text{Pa}_{R_j}$ 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
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 the above model 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.

**Figure 3.8: Flexible mixture model with independent component ratings**

\[
P(Z_u, Z_i | \tilde{X}) = \frac{P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u) \prod_{j=1}^{5} P(R_j|Z_u, Z_i)}{\sum_{Z_u} \sum_{Z_i} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u) \prod_{j=1}^{5} P(R_j|Z_u, Z_i)} 
\] (3.7)
3.2. MULTI-COMPONENT RATING RECOMMENDER SYSTEM

M-step

\[
P(Z_u) = \frac{1}{L} \sum_l \sum_{Z_i} P(Z_u, Z_i | \overrightarrow{X}(l)) \quad (3.8)
\]

\[
P(Z_i) = \frac{1}{L} \sum_l \sum_{Z_u} P(Z_u, Z_i | \overrightarrow{X}(l)) \quad (3.9)
\]

\[
P(U | Z_u) = \frac{\sum_{t:U(t)=U} \sum_{Z_i} P(Z_u, Z_i | \overrightarrow{X}(l))}{\sum_l \sum_{Z_u} P(Z_u, Z_i | \overrightarrow{X}(l))} \quad (3.10)
\]

\[
P(I | Z_i) = \frac{\sum_{t:I(t)=I} \sum_{Z_u} P(Z_u, Z_i | \overrightarrow{X}(l))}{\sum_l \sum_{Z_u} P(Z_u, Z_i | \overrightarrow{X}(l))} \quad (3.11)
\]

\[
P(R_j | Z_u, Z_i) = \frac{\sum_{t:R_j(t)=R_j} \sum_{Z_u} P(Z_u, Z_i | \overrightarrow{X}(l))}{\sum_l P(Z_u, Z_i | \overrightarrow{X}(l))} \quad (3.12)
\]

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

E-step

\[
P(Z_u, Z_i | U, I, O) = \frac{P(Z_u)P(Z_i)P(I | Z_i)P(U | Z_u)P(O | Z_u, Z_i)}{\sum_{Z_u} \sum_{Z_i} P(Z_u)P(Z_i)P(I | Z_i)P(U | Z_u)P(O | Z_u, Z_i)} \quad (3.13)
\]

M-step

\[
P(Z_u) = \frac{1}{L} \sum_l \sum_{Z_i} P(Z_u, Z_i | U(l), I(l), O(l)) \quad (3.14)
\]

\[
P(Z_i) = \frac{1}{L} \sum_l \sum_{Z_u} P(Z_u, Z_i | U(l), I(l), O(l)) \quad (3.15)
\]

\[
P(U | Z_u) = \frac{\sum_{l:U(l)=U} \sum_{Z_i} P(Z_u, Z_i | U(l), I(l), O(l))}{L \times P(Z_u)} \quad (3.16)
\]

\[
P(I | Z_i) = \frac{\sum_{l:I(l)=I} \sum_{Z_u} P(Z_u, Z_i | U(l), I(l), O(l))}{L \times P(Z_i)} \quad (3.17)
\]

\[
P(O | Z_u, Z_i) = \frac{\sum_{l:O(l)=O} \sum_{Z_u} P(Z_u, Z_i | U(l), I(l), O(l))}{\sum_l P(Z_u, Z_i | U(l), I(l), O(l))} \quad (3.18)
\]
3.3 Results and discussion

3.3.1 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 CPTS 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:

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

(3.20)
Although the expression for all three models is 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 over-
all rating, namely,

$$P(O|u_a, i) = \frac{P(u_a, i, O)}{\sum_O P(u_a, i, O)}$$  \hspace{1cm} (3.21)

The mode of this distribution of $O$ is predicted as the output$^2$.

**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 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.

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

$^2$Use of expectation of $O$ to predict did not change the conclusions of the experiments.
of precision at 11 standard recall levels (0%, 10%, 20%, ..., 100%) is commonly used to compare the performance of different systems (Baeza-Yates and Ribeiro-Neto 1999).

**Accuracy in rating prediction**

Mean absolute error is computed as:

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

where,

- $L_{\text{test}}$ = the number of records in the test data
- $o_l$ = the true rating
- $\hat{o}_l$ = the predicted rating

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. 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. 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.

However, the most interesting question is “Can we better predict the Overall rating using more components of ratings?” Error-plots of the two algorithms show that there is an advantage when training set is small—up to about 40% of the available dataset for training in this case. This is an important scenario in any real world collaborative filtering system. Often there are lots of new users in the system who have rated only a few items. Most of the existing collaborative filtering systems perform badly for these new users. However, for these users using a set of richer ratings can help. But, when there is enough training data, using only the overall 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 $t$ - 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 items**

Three algorithms were trained as described in the Section 3.2.4 at different training set sizes. Then the expected ratings for each user, item pair in the test set were computed.
3.3. RESULTS AND DISCUSSION

Figure 3.9: Plot of errors by amount of training data used, for different models.
Figure 3.10: Precision-Recall curve for three methods
3.3. RESULTS AND DISCUSSION

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 expected rating. The goal 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 difference between the three models is the most pronounced. The model with discovered structure gives the highest precision at each recall level followed by the model using only the Overall component. The models with independence assumption among the component rating returns lowest precision. However, as we use more training data the difference between these models 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, 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 have an opportunity here to compare our precision-recall figures with those obtained in (Adomavicius and Kwon 2007) as they have used data from the same source, i.e., Yahoo Movies!. The precision they got at recall level less than 10% using 90% of the records for training are 72%–75%—very similar to the precision we see in the last plot of Figure 3.10.

**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 above experiment 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
3.3. RESULTS AND DISCUSSION

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, ..., 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 3.11).

The conclusion of the earlier comparison of the models is still valid in this more realistic test (Figure 3.12 and 3.13). 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 well. When there are more than 11 training instances using only overall rating 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 inconsistency is also reflected in the precision and recall curves (Figure 3.13). However, there is an advantage of using five dependent components as can be observed from Figure 3.13.

3.3.2 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
3.3. RESULTS AND DISCUSSION

![Graph showing MAE vs training fraction for different rating orders.](image)

Figure 3.12: Three algorithms compared by using data in time order for training

<table>
<thead>
<tr>
<th>With unfilled components</th>
<th>Filled in components</th>
<th>% Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of User</td>
<td>1058</td>
<td>1680</td>
</tr>
<tr>
<td>Number of Item</td>
<td>3430</td>
<td>3926</td>
</tr>
<tr>
<td>Number of Records</td>
<td>45892</td>
<td>74110</td>
</tr>
</tbody>
</table>

Table 3.7: Increase in dataset size after filling in missing components

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 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:
3.3. RESULTS AND DISCUSSION

Figure 3.13: Precision-Recall curves using prior data for training and later data for testing
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 3.7 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:

\[
P(U, \vec{R}, I) = \sum_{Z_u, Z_i} P(Z_u) P(Z_i) P(I|Z_i) P(U|Z_u) \prod_{j=1}^{5} P(R_j|Z_u, Z_i, Pa_{R_j})
\] (3.22)

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 3.22:

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 \( o \).
3.3. RESULTS AND DISCUSSION

<table>
<thead>
<tr>
<th>Method</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-component FMM</td>
<td>0.353</td>
</tr>
<tr>
<td>SPSS EM</td>
<td>0.368</td>
</tr>
<tr>
<td>SPSS Regression</td>
<td>0.569</td>
</tr>
<tr>
<td>CF predicting Components</td>
<td>0.701</td>
</tr>
</tbody>
</table>

Table 3.8: Comparison of different methods filling in missing rating components.

The operations result in the following

\[ P(U, I, S, O) = \sum_{Z_u} P(Z_u) \sum_{Z_i} P(Z_i) P(O|Z_u, Z_i) P(I|Z_i) P(U|Z_u) P(S|Z_u, Z_i, O) \]

\[ \Rightarrow P(u, i, S, o) = \sum_{Z_u} P(Z_u) \sum_{Z_i} P(Z_i) P(o|Z_u, Z_i) P(i|Z_i) P(u|Z_u) P(S|Z_u, Z_i, o) \]

\[ \propto P(S|u, i, o) \]

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. Only \( U, I \) and \( O \) variable values were used from the test data to predict the 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

\(^3\)Experiments with 2, 3 and 5 fold cross validations yield similar results
We compared our results with the performance of the Missing Value Analysis (MVA) routines of SPSS. The Error values are given in Table 3.8.

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 they 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 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.

1. Predict missing values of the incomplete records using linear regression model with co-efficients estimated so far,

2. Estimate the regression coefficients using the complete data and predictions for the missing data.

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 3.15). 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 3.16 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 3.16, 3.17). 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
3.3. RESULTS AND DISCUSSION

Training and testing in time order

Training and testing in random order

Figure 3.17: Precision recall curves for filled in data
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. This new algorithm has improved prediction when there is very little training data available—a common problem faced by the real world collaborative filtering based recommender system. However, when there is ample training data there is little benefit of using additional components of ratings. In fact, in such a scenario when our objective is to predict the ratings that a user might give to an unseen item, using only the Overall rating leads to higher accuracy. But, when our objective is to retrieve the highest rated items for an user, using multiple components with the discovered dependency structure leads to higher precision.

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.
Our work suggests several future research directions. One of the 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, especially in the organizational context, 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 paper, 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 themes of conversation using keywords, people and timestamps of the posts. We represent the data as a tensor and show that higher order factorizations of the tensor can separate themes of conversation, identify important people in each such theme, and determine the activity level of each theme over time. We evaluate this approach by applying it to a dataset extracted from a 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. We propose to extend this analysis by including blog reading data which can reveal dominant topics that are being read. We also propose to study the dependency between blog reading, posting, and citation.

4.1 Introduction

Increasingly organizations are creating private blogs to promote peer-to-peer communication among employees. The activities in the blog network permit monitoring employee opinion, identify leaders or experts in different areas 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
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...

Table 4.1: Some of the topics in a blog network along with posting pattern of people behind them.

<table>
<thead>
<tr>
<th>Id: xxx081 Date: 2007-09-05</th>
<th>Id: xxx991 Date: 2007-11-09</th>
<th>Id: xxx368 Date: 2007-10-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>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. ...</td>
<td>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 ...</td>
<td>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 ...</td>
</tr>
</tbody>
</table>

(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”)
4.2. THEME IDENTIFICATION

In this paper we work with the intuition that a theme of posts can be considered significant if it is posted by authorities in the topic, has endured for a period of time, and has significant content. Such importance scores are not available a priori. However they can be determined from the occurrence patterns of authors and keywords over time.

4.1.1 Objective

The objective we pursue in this paper is to detect important developments in conversations taking place in a social network along with the important actors who were responsible for those developments and important time periods when the developments occurred.

4.2 Theme Identification

4.2.1 Background

Singular value decomposition of an adjacency matrix \((from \times to)\) of a network results in the hub and authority scores of the nodes (Pagerank, 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 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.

However, 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 there is a timestamp associated with each tie, 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. One example is TOPHITS that extends the HITS algorithm by associating anchor text with the hyperlinks (Kolda and Bader 2006). We extend the application of tensor decomposition to two higher order cases for blog data analysis.

4.2.2 Blog topic development

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 themes of conversation that are im-
important, we posit that we need to look beyond the word occurrences in the blog posts. Spikes of posts containing same keywords in two separated time periods are often about different subjects, e.g., posts with keywords related to “hurricane” published in last week of Aug ’05 are likely to be about the hurricane Katrina, whereas the posts with similar keywords in the last week of Sep ’05 are likely to be about hurricane Rita. Similarly, posts containing same keywords made by two different people are likely to differ in the perspective offered in the same topic. Therefore, to identify different themes in blog posts, the relevant variables are the authors, timestamps and keywords of the blog posts. This data can be represented as a $\text{author} \times \text{keyword} \times \text{timestamp}$ tensor $X$, where, each cell of the tensor contains $tf-idf$ weighted and length normalized counts of the word occurrences. This value indicates the strength of co-occurrence of the three variables. Consider 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 scores. When we want to identify only one dominant topic, according to our definition the importance of $p$th blogger in this topic can be calculated as:

$$a_p = \sum_q \sum_r x_{pqr}k_qt_r \iff a = X \times_2 k \times_3 t$$ (4.1)

Similarly

$$k = X \times_1 a \times_3 t$$ (4.2)

$$t = X \times_1 a \times_2 k$$ (4.3)

Where, $X \in \mathbb{R}^{a \times k \times t}$ is the data tensor; $a, k,$ and $t$ are the vectors of importance of the authors, keywords and time periods. $X_j$ is the $j$-mode product of a vector with a tensor. Applied iteratively the vectors $a, k,$ and $t$ converge to minimize the error $\|X - a \circ k \circ t\|_F$, where, $\circ$ is the outer product between the vectors (De Lathauwer, De Moor et al. 2000). Thus $a \circ k \circ t$ is the best rank-1 approximation of the tensor $X$. 
4.2. THEME IDENTIFICATION

Extending from one dominant topic to R topics and using a normalizer λ to make each vector of unit length the approximation can be expressed as sum of R rank-1 tensors:

\[ X = \sum_{r}^{R} \lambda_r \times a_r \circ k_r \circ t_r = [\lambda; A, K, T] \quad (4.4) \]

where, A,K, and T are the three modal matrices each with \( R a_r, k_r, \) and \( t_r \) as column vectors respectively. This decomposition is known as Parallel Factorization (PARAFAC) (Harshman 1970). The popular approach to compute this decomposition is based on Alternating-Least-Square error minimization (ALS). The error \( ||X - [\lambda; A, K, T]||_F \) is minimized by successively optimizing one of the three matrices while keeping the remaining 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).

4.2.3 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. Content of the post not only depend on who is making the post but also who is it targeted to. To capture this fact we represent the blog data in a fourth order tensor (\( \text{author} \times \text{recipient} \times \text{keywords} \times \text{timestamp} \)). 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.

4.2.4 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 \( \text{from} \times \text{to} \times \text{term} \) 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.

4.2.5 Data description and representation

The data for this study is collected from a private blog network in a large IT services firm. It contains the blog posts and replies along with timestamps and demographic
### Table 4.2: Blog data description

<table>
<thead>
<tr>
<th>Blog post and reply data</th>
<th>Blogger employment data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date range</td>
<td>Years in the company</td>
</tr>
<tr>
<td>Blog posts</td>
<td>Office location</td>
</tr>
<tr>
<td>Comments</td>
<td>Designation</td>
</tr>
<tr>
<td>Bloggers</td>
<td>Supervisor</td>
</tr>
<tr>
<td>Blogs</td>
<td></td>
</tr>
<tr>
<td>71,000 average length 300 words</td>
<td>4800 Commentors 16,000</td>
</tr>
<tr>
<td>286,000 average length 33 words</td>
<td></td>
</tr>
<tr>
<td>4700</td>
<td></td>
</tr>
</tbody>
</table>

information about the bloggers. We used two subsets of the data for the two methods described in Section 2. For blog topic development analysis we used the text of the blog posts and the comments, author-ids, and the timestamp. Texts of the posts are converted to vectors of term weights by removing the stopwords from them and tokenizing them. Then occurrence counts of the tokens are \(tf-idf\) 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 \(\text{author} \times \text{keyword} \times \text{timestamp}\) tensor. Each cell of the tensor contains total weight of a keyword used by an author on a particular day. This resulted in a \(13665 \times 646 \times 36637\) tensor with \(4,750,874\) nonzero entries \((\text{sparsity} = 1.47 \times 10^{-5})\). For the blog conversation development analysis we used the text in only the comments, 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. We carried out the same transformation of text and the timestamp as done in the case of blog-topic-development, but, this time we stored the data in a \(\text{author} \times \text{recipient} \times \text{keyword} \times \text{timestamp}\) 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 \(12845 \times 2767 \times 646 \times 17317\) tensor with \(1,470,688\) nonzero entries \((\text{sparsity} = 3.7 \times 10^{-9})\).

### 4.3 Results and Discussion

Each tensor was decomposed into 20 rank-1 tensors using PARAFAC. Some of the resulting factors are displayed in 4.1 and 4.2. For each factor the top five keywords are shown along with the importance of the top-authors and the daily intensity of the topic.

As we can see from the first set of results the decomposition is separating activities of different topics. The intensities of the topics over time tell us the nature of the conversation: posts about “software testing” have generated interest for much shorter period compared to the conversations about “new technology companies”. The importance scores of the 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 “software testing” and “Indian political history” topics, though they were active for a shorter period.
Figure 4.1: Trends of topics and important bloggers in each

Figure 4.2: of topics and topic specific hub and authority score
We manually examined the posts made by the top-authors and compared them with some of the other authors. Most of the posts are chatter: jokes, quotes, puzzles, and collections of random news. The chatter did not suggest any expertise on the part of the author and was ignored by the algorithm. The posts made by the top-authors on the topic were persistently focused on the topic and were generally longer. They were rightly detected by the algorithm as being significant. Other chatter by the same authors was ignored. In general the most significant author in a topic was easily identifiable as being an expert in the area. However, the second and third significant author was found to be only loosely related to the topic. Multiple experts on one topic are a rare occurrence in our blog posts dataset.

Analysis of comments on the blog posts reveals a different set of factors. These are usually the factors that generate higher than average amount of reactions from the bloggers, e.g., religion, mythology and law. The effectiveness of the decomposition is illustrated by the latent semantic separation of “religion and food”, “spirituality and mythology” that can be thought of as a part of broader topic of religion. We do not observe multiple authorities in any topic due to the nature of the conversation in the blogs: there are certain blogs that are popular places for discussing certain topics. In a sense these blogs are the harbors for conversations in the topic and they play a central role in defining the ongoing conversations as a topic.

4.4 Proposed next steps

4.4.1 Task based evaluation of tensor factors

The idea behind multi-modal blog data analysis is that by using people and timestamp information in addition to the text we should be able to better identify topics of conversation and significant people in the social network. I propose to do two task based evaluation of blog conversation analysis by tensor factorization.

Topic discovery

The factorization of the tensor representing the entire data set reveals topics of conversation. We evaluate these topics by formulating the factorization as a clustering problem.

First we collect topic labels given to the blog posts by the authors and use them as “gold standards”. Then we proceed to assign each conversation to a factor. An example of a conversation can be the set of responses sent by a group of people to the author of a blog post. We use the proposed tensors to encode individual conversations. The text of these messages, along with the sender, recipient and timestamp information are represented in a \textit{author} × \textit{recipient} × \textit{keyword} × \textit{day} tensor. On the other hand, each of the resulting factor of the four dimensional tensor consists of a set of four vectors of weights over the \textit{authors}, \textit{recipients}, \textit{keywords}, and the \textit{days}. Each of these vectors is multiplied with the tensor representing a single conversation to obtain a score for the particular conversation-factor assignment. This score is computed...
for each factor in turn and the conversation is assigned to the factor with the highest score. Thus the conversations are clustered into several topics with the help of tensor factorization. The quality of these clusters will be measured by following standard cluster evaluation methods such as those presented in Section 2.5.1.

Community discovery

The second test is based on the idea that since we are using the text of the response to describe the tie between two actors in the blog social network, tensor factorization could lead to more accurate identification of the communities of the bloggers than graph partitioning of the social network that uses number of messages between two individuals as a measure of the tie between the two. This test would evaluate the accuracy of the communities identified by the tensor factorization method with the help of the community labels and would compare the accuracy to that of graph partitioning methods that do not take into account any text label on the edges.

4.4.2 Analysis of blog Reading behaviour

Dominant topics as dictated by the reading behaviour

The observed relations between bloggers in the form of citation and response to each other’s post is based on the actor reading the target blogger’s post. Reading activity is usually not observable at the blogs. However, by collecting access logs from the webserver that host the blogs we are able to collect information on who is reading whose blog post along with the timestamp of the reading. Since the reading tie has different characteristic than the response tie, analysis of reading behaviour is likely to yield different dominant topics. Comparison of these topics with those obtained by analyzing blog posts and responses would give us insight into different behaviours of the employee bloggers.

Interdependence of Reading, Posting, and Replying

One of the less researched topics in blog analysis is the relation between reading, posting and replying. Using the collected data I plan to explore the dependence between these three activities. This work will start by measuring the time lag between the reading and posting activity in the dominant topics in the blogs. Such lag can be measured by computing the cross correlation function (CCF) between the time series of the two activities. The time lag and the shape of the CCF will give insight into the nature of the topic.

Link prediction

One task of considerable interest is predicting link between two individuals from the observed behaviour of the two. For example predicting who is reading whose blog,
an activity often unobserved, from the sequence posts by individuals and their timestamps. Since, we have an unique dataset on the reading and posting behaviour of bloggers we have an opportunity to predict and evaluate reading link between bloggers.
Document Clustering

A.1 MLE of Katz’s distribution parameters

The Katz’s distribution is defined as:

\[
P(0) = p_0 \]
\[
P(k) = (1 - p_0)(1 - p)p^{k-1}; \text{ when } k > 0
\]

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 \( df = N - n_0 = \sum_{k=1}^{K} n_k \) and
- collection term frequency \( cf = \sum_{k=1}^{K} kn_k \)

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

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

where, \( \delta(\cdot) \) is the indicator function that is 1 if argument is zero and 0 otherwise.
A.1. MLE OF KATZ’S DISTRIBUTION PARAMETERS

Log of likelihood is

$$\text{LL}(p, p_0) = n_0 \log(p_0) + \sum_{k=1}^{K} [n_k \log(1 - p_0) + n_k \log(1 - p) + n_k(k - 1) \log(p)]$$

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

$$\frac{\partial \text{LL}}{\partial p_0} = \frac{n_0}{p_0} + \sum_{k=1}^{K} \frac{n_k - 1}{1 - \hat{p}_0} = 0$$

$$\Rightarrow \frac{n_0}{p_0} = \frac{1}{1 - \hat{p}_0} \sum_{k=1}^{K} n_k = \frac{1}{1 - \hat{p}_0} (N - n_0)$$

$$\Rightarrow \frac{1 - \hat{p}_0}{p_0} = \frac{N - n_0}{n_0}$$

$$\Rightarrow \frac{1}{p_0} - 1 = \frac{N}{n_0} - 1$$

$$\Rightarrow \hat{p}_0 = \frac{n_0}{N} = \frac{N - df}{N} = 1 - \frac{df}{N} \quad (A.2)$$

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

$$\frac{\partial \text{LL}}{\partial p} = \sum_{k=1}^{K} \frac{n_k - 1}{1 - \hat{p}} + \frac{n_k(k - 1)}{\hat{p}} = 0$$

$$\Rightarrow 0 = \frac{1}{\hat{p}} \sum_{k=1}^{K} n_k(k - 1) - \frac{1}{1 - \hat{p}} \sum_{k=1}^{K} n_k$$

$$\Rightarrow 0 = \frac{1}{\hat{p}} \left( \sum_{k=1}^{K} kn_k - \sum_{k=1}^{K} n_k \right) - \frac{1}{1 - \hat{p}} \sum_{k=1}^{K} n_k$$

$$\Rightarrow 0 = \frac{1}{\hat{p}} (cf - df) - \frac{1}{1 - \hat{p}} df$$

$$\Rightarrow \frac{1 - \hat{p}}{\hat{p}} = \frac{df}{cf - df}$$

$$\Rightarrow \frac{1}{\hat{p}} = \frac{cf}{cf - df}$$

$$\Rightarrow \hat{p} = \frac{cf - df}{cf} \quad (A.3)$$

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.

\[
P(U, I, O) = \sum_{Z_u} \sum_{Z_i} \sum_S \sum_A \sum_V \sum_D P(Z_u, Z_i, U, I, S, A, V, D, O)
\]

\[
= \sum_{Z_u} \sum_{Z_i} \sum_S \sum_A \sum_V \sum_D P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)P(S|Z_u, Z_i, O)\]

\[
P(A|Z_u, Z_i, O)P(V|Z_u, Z_i, O)P(D|Z_u, Z_i, O)P(O|Z_u, Z_i)
\]

\[
= \sum_{Z_u} \sum_{Z_i} P(O|Z_u, Z_i)P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\sum_{S} P(S|Z_u, Z_i, O)\]

\[
= \sum_{Z_u} \sum_{Z_i} P(O|Z_u, Z_i)\sum_{A} P(A|Z_u, Z_i, O)\sum_{V} P(V|Z_u, Z_i, O)\sum_{D} P(D|Z_u, Z_i, O)
\]

\[
= \sum_{Z_u} \sum_{Z_i} P(O|Z_u, Z_i)P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)
\]

\[
= \sum_{Z_u} P(Z_u)P(U|Z_u)\sum_{Z_i} P(O|Z_u, Z_i)P(Z_i)P(I|Z_i) \quad \text{(B.1)}
\]

The conditional probability terms for $S, A, V$ 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.

\[
P(U, I, O) = \sum_{Z_u} \sum_{Z_i} \sum_{S} \sum_{A} \sum_{V} \sum_{D} P(Z_u, Z_i, U, I, S, A, V, D, O)
\]

\[
= \sum_{Z_u} \sum_{Z_i} \sum_{S} \sum_{A} \sum_{V} \sum_{D} P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)P(S|Z_u, Z_i)P(A|Z_u, Z_i)P(V|Z_u, Z_i)P(D|Z_u, Z_i)P(O|Z_u, Z_i)
\]

\[
= \sum_{Z_u} \sum_{Z_i} P(O|Z_u, Z_i)P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)\sum_{S} P(S|Z_u, Z_i)
\]

\[
= \sum_{Z_u} \sum_{Z_i} P(O|Z_u, Z_i)P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)
\]

\[
= \sum_{Z_u} P(Z_u)P(U|Z_u)\sum_{Z_i} P(O|Z_u, Z_i)P(Z_i)P(I|Z_i) \quad \text{(B.2)}
\]

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

\[
P(U, I, O) = \sum_{Z_u} \sum_{Z_i} P(Z_u, Z_i, U, I, O)
\]

\[
= \sum_{Z_u} \sum_{Z_i} P(O|Z_u, Z_i)P(Z_u)P(Z_i)P(I|Z_i)P(U|Z_u)
\]

\[
= \sum_{Z_u} P(Z_u)P(U|Z_u)\sum_{Z_i} P(O|Z_u, Z_i)P(Z_i)P(I|Z_i) \quad \text{(B.3)}
\]
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 metrics 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
B.2. HALO IN MULTI-CRITERIA MOVIE RATING

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

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>a</th>
<th>v</th>
<th>d</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>1.00</td>
<td>0.79</td>
<td>0.82</td>
<td>0.74</td>
<td>0.87</td>
</tr>
<tr>
<td>a</td>
<td>0.79</td>
<td>1.00</td>
<td>0.81</td>
<td>0.73</td>
<td>0.83</td>
</tr>
<tr>
<td>v</td>
<td>0.82</td>
<td>0.81</td>
<td>1.00</td>
<td>0.79</td>
<td>0.88</td>
</tr>
<tr>
<td>d</td>
<td>0.74</td>
<td>0.73</td>
<td>0.79</td>
<td>1.00</td>
<td>0.80</td>
</tr>
<tr>
<td>o</td>
<td>0.87</td>
<td>0.83</td>
<td>0.88</td>
<td>0.80</td>
<td>1.00</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$r_{R_i R_j O}$</th>
<th>$S$</th>
<th>$A$</th>
<th>$V$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>1.00</td>
<td>0.25</td>
<td>0.26</td>
<td>0.15</td>
</tr>
<tr>
<td>$A$</td>
<td>0.25</td>
<td>1.00</td>
<td>0.32</td>
<td>0.22</td>
</tr>
<tr>
<td>$V$</td>
<td>0.26</td>
<td>0.32</td>
<td>1.00</td>
<td>0.33</td>
</tr>
<tr>
<td>$D$</td>
<td>0.15</td>
<td>0.22</td>
<td>0.33</td>
<td>1.00</td>
</tr>
</tbody>
</table>

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 and 0.60 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:

<table>
<thead>
<tr>
<th>Factor</th>
<th>One</th>
<th>Two</th>
<th>Three</th>
<th>Four</th>
<th>Five</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigen values</td>
<td>4.22</td>
<td>0.28</td>
<td>0.22</td>
<td>0.16</td>
<td>0.11</td>
</tr>
</tbody>
</table>

% 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).
### B.2. HALO IN MULTI-CRITERIA MOVIE RATING

<table>
<thead>
<tr>
<th>Factor</th>
<th>One</th>
<th>Two</th>
<th>Three</th>
<th>Four</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigen values</td>
<td>1.77</td>
<td>0.86</td>
<td>0.73</td>
<td>0.63</td>
</tr>
<tr>
<td>% variance explained</td>
<td>44.3</td>
<td>21.6</td>
<td>18.3</td>
<td>15.8</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Factor</th>
<th>Factor 2</th>
<th>Uniquenesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>0.91</td>
<td>0.23</td>
</tr>
<tr>
<td>$A$</td>
<td>0.87</td>
<td>−0.02</td>
</tr>
<tr>
<td>$V$</td>
<td>0.93</td>
<td>−0.08</td>
</tr>
<tr>
<td>$D$</td>
<td>0.84</td>
<td>−0.12</td>
</tr>
<tr>
<td>$O$</td>
<td>0.95</td>
<td>0.03</td>
</tr>
</tbody>
</table>

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)
B.2. HALO IN MULTI-CRITERIA MOVIE RATING

<table>
<thead>
<tr>
<th>Users with # of ratings</th>
<th>fraction of variance explained by components</th>
<th>avg corr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of records</td>
<td></td>
</tr>
<tr>
<td>⩽ 5</td>
<td>346973</td>
<td>0.877</td>
</tr>
<tr>
<td>&gt; 5&amp; ⩽ 10</td>
<td>37473</td>
<td>0.862</td>
</tr>
<tr>
<td>&gt; 10&amp; ⩽ 20</td>
<td>27378</td>
<td>0.848</td>
</tr>
<tr>
<td>&gt; 20&amp; ⩽ 40</td>
<td>18519</td>
<td>0.837</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>25493</td>
<td>0.855</td>
</tr>
</tbody>
</table>

|                         | 0.042 | 0.035 | 0.026 | 0.02 | 0.84 |
| ⩽ 5                     |       |       |       |      |      |
| > 5& ⩽ 10               | 0.048 | 0.039 | 0.029 | 0.021 | 0.82 |
| > 10& ⩽ 20              | 0.053 | 0.042 | 0.033 | 0.024 | 0.80 |
| > 20& ⩽ 40              | 0.057 | 0.045 | 0.036 | 0.026 | 0.79 |
| > 40                    | 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).


Doddington, George, Jaime Carbonell, James Allan, Jonathan Yamron, Umass Amherst, Yiming Yang. 2000. Topic detection and tracking pilot study final report. 11, 14


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. 89


REFERENCES


Koltuv, B. B. 1962. Some characteristics of intrajudge trait intercorrelations. *Psychological Monograph* 76. 48, 89


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. 


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. 


