Information Retrieval with Bayesian Sets
and Extensions
by
Will Youzhi Zou (F)
Fourth-year undergraduate project in Group F, 2006/2007

"I hereby declare that, except where specifically indicated, the work submitted herein is my own original work."

Will Youzhi Zou

Date_____

CONTENTS

Introduction

In 2002 alone, the human world produced 5 exabytes (10^{18} bytes) [1] of information, equivalent to all the words ever spoken by human beings. The information in this world is growing at mind-blowing speed in the form of text, films, and computer data storage.

Does producing information mean advance in human knowledge? It is not hard to recognise that the amount of information produced has no direct correlation to significance, and more information without a way to find what's useful, can only disorient human minds. Information overload, the problem of overwhelming amount of unclassified information, has become a serious issue even to the life of average people, who has learned to use the modern communication tools such as the World Wide Web.

The field of Information Retrieval tries to solve the problem of information overload. Given some small amount of query information provided by the user, an automated information retrieval system tries to find the exact piece of information required, from a mass information storage base. However, the best tools available to us, computing machines, have storage specifications which are not optimal for structuring, processing and retrieving information needed by the human user. The field of information retrieval have developed comprehensive computational algorithms, and applied advanced interdisciplinary techniques to this demanding area.

In this report, we will combine the perspectives of information retrieval with machine learning, and discuss a specific machine learning algorithm developed for information retrieval, Bayesian Sets (Z.Ghahramani and K.A. Heller, 2005) [2]. This algorithm is proposed to solve an interesting problem of clustering-on-demand, or finding similar items given some examples. In the next section, the report will explore theoretical aspects of Bayesian Sets in detail. Two extensions to the original Bayesian Sets algorithm are introduced, and we will look at applying the algorithm in different ways with different types of data.

The form of information needs by the human user is hardly easily understood by the computer. For instance, the request for a piece of 'soothing' music cannot be comprehended by a machine unless pieces of music are marked with tags of 'soothing', 'invigorating' in binary form. It will cost enormous human effort to label these data, analogous to program a microprocessor by assembly code. Is there a way to have the computer comprehend these concepts? In this report, we try to do this using a collaborative dataset, i.e. a dataset formed

by human opinions. We use the Bayesian Sets algorithm in the collaborative, or a trivial 'human interest' data-space of multi-dimensional binary data, to retrieve similar items to those specified by the user query and have the retrieval system pick up common similarities in the items' content features. In Section 3, the report will discuss implementation of Bayesian Sets and extensions on a recently published collaborative database, the Netflix Movie dataset [3].

In Section 4, a number of experiment designs is specified and implemented using MATLAB. Using movie content information extracted from the Internet Movie Data Base (IMDB) [4] (www.imdb.com), we show by experimentation the effectiveness and the advantages of Bayesian Sets used in information retrieval of collaborative data. We compare results to other retrieval algorithms and evaluate a new implementation which improves the original Bayesian Sets algorithm.

Theoretical Results and Derivations

It is essential to introduce and derive, at the start of this report, the key theoretical results that will be used as the basis of implementation and experimentation. We will show in this section:

- The Boolean formulation for information retrieval
- The basic Bayesian Sets algorithm for information retrieval
- An extension to Bayesian Sets applied to Negative Relevance Feedback (NRF)
- An implicit method of applying NRF with content information.

1.1 Information Retrieval and its Basic Formulations

"Information Retrieval is finding materials of an unstructured nature that satisfy information need from within large collections." [5]

To introduce our formulations of information retrieval, it is useful to look at the common problem of retrieving documents with the text query. Suppose there are a large number of documents. The retrieval task is find relevant documents to a query text string. Given the query, say "Cambridge", the simplest form of search that we can perform is:

```
For all stored documents,
      Search for the matching string "Cambridge"

Return all the documents that have one or more matches
```

If the document base is large, the text search can be computationally expensive. To perform real-time retrieval by request, we could use a form of pre-computation, or indexing. A simple way to index the raw data can be performed if we obtain the following Boolean matrix:

	Doc 1	Doc 2	Doc 3	Doc 4
"Cambridge"	0	1	1	0
"London"	1	0	0	1
"Paris"	0	0	1	0
"New York"	1	1	0	1

Figure 1.1: Term-document incident matrix. Element (i,j) is 1 if document in column j contains the term in i

The matrix given in Figure 1.1 is called a "term-document incident matrix", where the incidence of a term in a document is represented by a binary '1'. Provided with a complete vocabulary of words, this matrix is pre-processed from the documents and stored in computer memory. Now to perform retrieval by request, we can simply search through rows for '1's and return the documents accordingly. By solving the above simple retrieval problem, we have assumed a trivial definition for 'relevance'. A document is '*relevant*' to the query if it *contains* the term given in the query.

From the above simple example, it can be seen that the problem of information retrieval can be broken down into basic components: the raw data, the indexed/processed data, the query and the algorithm that defines relevance.

In our formulation, we use the above Boolean model with collaborative data:

	Movie 1	Movie 2	Movie 3	Movie 4
Katie	1	1	0	0
Phil	0	1	0	1
Ken	1	0	1	1
Ruth	0	0	0	1

Figure 1.2: Collaborative incident matrix. Element (i,j) is 1 if user i likes movie j

Figure 1.2 shows the form of pre-processed collaborative data on which we apply the Bayesian Sets algorithm in the scope of this report. Each element (i,j) is a binary indicator of whether user i likes item j. In the pre-processed data-space of U users and I items, each item (movie) is represented by a I-dimensional binary vector [0,1,1,0,1…….].

Unlike the text retrieval example, searching through a certain user's preferred movies is not informative. This is because the knowledge that one user likes a certain movie give us little information about the movie. However, the collaborative opinion of *all* the users toward this movie gives us much richer information. We use the formulation that movies, represented by binary vectors of user opinion, form the elements of both query and target of retrieval. Using the collaborative information, we'd like the retrieval to system to answer this question: given a query 'set' of some items (movies), which other items in the dataset are 'similar' to the query? The definition of an item's relevance to the query is the key to solving this problem. This definition is clarified by introducing the Bayesian Sets algorithm.

1.2 Defining Relevance with Bayesian Sets

The Bayesian Sets method defines a general framework for finding data that belongs to the same concept/cluster as the specified query. Consider the following data from previously defined formulation:

A query set formed by two binary vectors:

$$Q = \{0000111000111, 0001100001011\}$$

Items (movies) in the collaborative dataset:

$$\mathbf{x}_1 = 1111000011000,\ \mathbf{x}_2 = 0000011010110, \ldots\ldots \mathbf{x}_n = 0010001010000$$

In order to retrieve the most relevant items, it is expected to rank the items $\mathbf{x}_1, \mathbf{x}_2 \ldots\ldots \mathbf{x}_n$ by a measure of relevance with Q.

To define similarity, one could measure the number of matching bits between elements of Q and $\mathbf{x}_i$, or use other intuitive measures. However, a more systematic approach is to perform probabilistic modeling. We assume the data points all come independently and identically distributed (i.i.d) and use the Bernoulli distribution to model them:

$$p(\mathbf{x}_i \mid \theta) = \prod_{j=1}^{J} \theta_j^{x_{ij}} (1-\theta_j)^{1-x_{ij}} \qquad \textbf{(1.1)}$$

Take the query data as been 'observed'; we arrive at an inference problem: having observed data Q, how likely are we to observe the data $\mathbf{x}_i$?

A number of techniques could be applied to solve this problem, such as performing maximum likelihood estimation for the model parameters. But for Bayesian Sets, we apply a full Bayesian approach.

We return to the general perspective. Consider a dataset where each data item is described by a certain data type, for instance, a vector of real numbers. The user has supplied a query set Q consisting of items that come from the same concept, and we'd like to find other items in the dataset that also belongs to this concept.

Suppose we have assumed a probabilistic model that governs the distribution of data, to evaluate how well an item may fit into the concept defined by Q, the basic measure is given by the probability p(x|Q), i.e. the probability of observing data x having observed data Q. In order to make this quantity sensible as a relevance measure, the basic measure is divided by the marginalised probability of observing the new data p(x), i.e. the probability of observing this data a priori.

The expression is given as:

$$score(\mathbf{x} \mid Q) = \frac{p(\mathbf{x} \mid Q)}{p(\mathbf{x})} \qquad \textbf{(1.2)}$$

which can also be written in the form:

$$score(\mathbf{x} \mid Q) = \frac{p(\mathbf{x}, Q)}{p(\mathbf{x})p(Q)} \qquad \textbf{(1.3)}$$

The above expression can be interpreted as a ratio of the probability of data x and Q being generated by the same model with the same set of parameters, to the probability that data x and Q are generated by models with different sets of parameters.

The score can be computed once a probabilistic model is assumed for the data. This can be shown by marginalising the numerator and denominators of the score expression in (1.2):

$$p(\mathbf{x} \mid Q) = \int p(\mathbf{x} \mid \theta) p(\theta \mid Q) d\theta \qquad \textbf{(1.4)}$$

$$p(\mathbf{x}) = \int p(\mathbf{x} \mid \theta) p(\theta) d\theta \qquad \textbf{(1.5)}$$

Returning to binary data, we have assumed our data to be i.i.d: each binary vector (J dimensions) has an independent Bernoulli distribution given in equation (1.1). The conjugate prior for the parameters of a Bernoulli distribution is the Beta distribution:

$$p(\theta \mid \alpha, \beta) = \prod_{j=1}^{J} \frac{\Gamma(\alpha_j + \beta_j)}{\Gamma(\alpha_j)\Gamma(\beta_j)} \theta^{\alpha_j - 1} (1-\theta)^{\beta_j - 1} \qquad \textbf{(1.6)}$$

Assume the query Q has N data points, i.e. $\mathbf{x}_i \in Q$ (i=1,2,…N). Equation (1.4) can be further derived:

$$p(\mathbf{x} \mid Q) = \int p(\mathbf{x} \mid \theta) \frac{p(Q \mid \theta) p(\theta)}{p(Q)} d\theta = \int p(\mathbf{x} \mid \theta) \frac{\prod_{i=1}^{N} p(\mathbf{x}_i \mid \theta) p(\theta)}{\prod_{i=1}^{N} p(\mathbf{x}_i)} d\theta \qquad \textbf{(1.7)}$$

Using equation (1.1), (1.5), (1.6) and (1.7), the scoring expression given in equation (1.2) can now be expressed with Bernoulli and Beta forms. In fact the derivation arrives at an expression of gamma functions:

$$score(\mathbf{x} \mid Q) = \frac{p(\mathbf{x} \mid Q)}{p(\mathbf{x})} = \prod_j \frac{\dfrac{\Gamma(\alpha_j + \beta_j + N)}{\Gamma(\alpha_j + \beta_j + N + 1)} \dfrac{\Gamma(\alpha_j + x_{\cdot j})\Gamma(\beta_j + 1 - x_{\cdot j})}{\Gamma(\alpha_j)\Gamma(\beta_j)}}{\dfrac{\Gamma(\alpha_j + \beta_j)}{\Gamma(\alpha_j + \beta_j + 1)} \dfrac{\Gamma(\alpha_j + x_{\cdot j})\Gamma(\beta_j + 1 - x_{\cdot j})}{\Gamma(\alpha_j)\Gamma(\beta_j)}} \qquad \textbf{(1.8)}$$

This expression can be simplified using gamma function property $\Gamma(x) = (x-1)\Gamma(x-1)$

(for x>1).

The element $x_{.j}$ can be either 0 or 1:

If $x_{.j}$=1, the score expression simplifies to $\frac{\alpha_j + \beta_j}{\alpha_j + \beta_j + N}\frac{\tilde{\alpha}_j}{\alpha_j}$

If $x_{.j}$=0, the score expression simplifies to $\frac{\alpha_j + \beta_j}{\alpha_j + \beta_j + N}\frac{\tilde{\beta}_j}{\beta_j}$

From arguments above, equation (1.8) becomes:

$$score(\mathbf{x} \mid Q) = \prod_j \frac{\alpha_j + \beta_j}{\alpha_j + \beta_j + N} \left(\frac{\tilde{\alpha}_j}{\alpha_j}\right)^{x_{.j}} \left(\frac{\tilde{\beta}_j}{\beta_j}\right)^{1-x_{.j}} \quad \textbf{(1.9)}$$

Taking log on both sides:

$$\log score(\mathbf{x} \mid Q) = c + \sum_j q_j x_{.j} \quad \textbf{(1.10)}$$

where

$$c = \sum_j \log(\alpha_j + \beta_j) - \log(\alpha_j + \beta_j + N) + \log \tilde{\beta}_j - \log \beta_j \quad \textbf{(1.11)}$$

$$q_j = \log \tilde{\alpha}_j - \log \alpha_j - \log \tilde{\beta}_j + \log \beta_j \quad \textbf{(1.12)}$$

If we form all the data into one matrix $\mathbf{X}$ such that the data points are rows of the matrix, the log score vector across all data points can be computed using one matrix-vector multiplication:

$$\mathbf{s} = c + \mathbf{X} \cdot \mathbf{q} \quad \textbf{(1.13)}$$

The above derivation for Bayesian Sets is given with reference to Z.Ghahramani, K.A.Heller *"Bayesian Sets"*(NIPS 2005)[2].

1.3 Interpretation of the Algorithm and Data Types

With the Bayesian Sets scoring expression given in the previous section, we can now efficiently apply the Bayesian Sets algorithm to binary data in the form defined in Section 1.1 using the Beta-Bernoulli framework. We are able compute the Bayesian Sets score for all

the data items in the dataset, and rank the relevance of items to the given query set, *in terms of* how well the item fit into the concept defined by the query.

We could define a space spanned by the dimensions of the data being used. In this space, data points in the dataset are distributed in certain clusters. Give a query, the Bayesian Sets algorithm, essentially performs a search in the data space and finds elements of the cluster that the query data belongs to. In other words, given we've observed the query data as elements of one cluster, the algorithm ranks other items by the criterion of whether an item belongs to this cluster. The needs of the user are satisfied if the retrieved cluster or 'concept' is in line with what he/she requires. Given more number of data points provided by the user, the system is able to find, more accurately, the cluster he/she is looking for.

The algorithm can be applied to data of other origins. In a movie retrieval system, the above process can be performed with both collaborative and content data. With content data, each movie can also be represented by a binary vector. The dimensions of the data vector are given by the enumeration of a specific feature, e.g. for movie genre: action, romance, horror etc. each element representing whether the movie has the corresponding content feature (e.g. genre).

Querying the collaborative data using a set of movies, it is very possible that returned results have similar content features, as some features are closely correlated to user preference. If we're in fact looking for similar content, wouldn't it be better if we applied the algorithm on the content data?

However, genre data for movies, for example, cannot easily *generalise* to other content features of a movie such as cast, director, language, year etc. There is little correlation between different content features. Allowing Bayesian Sets to generalize is one of the reasons why a collaborative data set is used as the primary data to run Bayesian Sets. Given enough example movies of a same concept, the algorithm is able to generalise to the required data cluster with collaborative data. The user does not need to specify the content information he/she requires, which is implicit with the query provided.

Nevertheless, a content dataset is collected for the implementation described in this report. We use the content data for two important purposes. First, the content data is used to evaluate the performance of the Bayesian Sets algorithm and extensions. Secondly, the content data is used to implement a Bayesian Sets extension to refine the retrieval process on collaborative data. This is performed using the Negative Relevance Feedback framework introduced below, applying the method implicitly. The implicit Negative Relevance Feedback is discussed in detail in Section 1.5.

1.4 Negative Relevance Feedback

We rewrite equation (1.3) for of the Bayesian Sets scoring criterion

$$score(\mathbf{x} \mid Q) = \frac{p(\mathbf{x}, Q)}{p(\mathbf{x})p(Q)}$$

The criterion is the ratio of the joint probability of observing x and Q, to the probability of independently observing x and Q. An interpretation is that the ratio compares two probabilities:

1. The probability of x and Q being generated by the same model with the same parameters θ
2. The probability of x and Q being generated by models with different parameters θ and θ'

The score compares the probability of the hypotheses that the data is generated by the two graphical models illustrated in Figure 1.3.

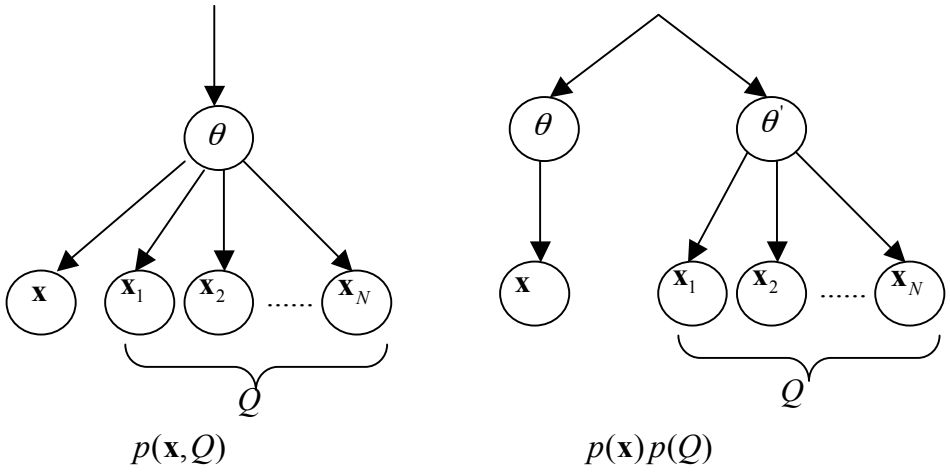

Figure 1.3: Comparison of hypotheses that the data was generated each of the two graphical models

In order to derive an expression for the negative relevance feedback of the retrieval algorithm, we study this formulation in more detail.

Suppose we have been supplied *another* set by the user: the Negative Relevance Set I, which specifies the items that belongs to the same concept, but a concept the user does not wish to search for and does not belong to the same concept as the query set Q. We would like to find the items which fit into the same cluster as Q *while* do not belong to the same concept as I.

From the given information that Q and I belong to different concepts. We have a fixed

graphical framework given by Figure 4.

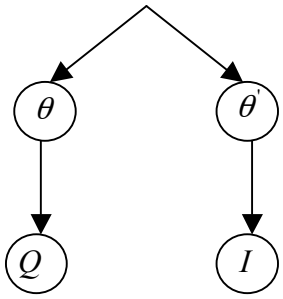

Figure 1.4: Initial graphical model framework for NRF

Given that Q and I belong to different concepts, what are the possible models that could generate data x, Q and I?

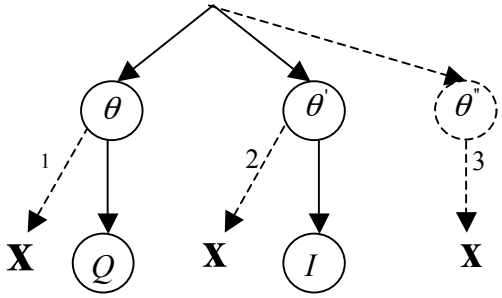

Figure 1.5: Illustration of all possible model assumptions

There are three possible hypotheses as shown on Figure 1.5, and the goal is to find data that is likely to fit hypothesis 1. Similar to the probabilities shown in Figure 1.3, we can define the probability criterion for how likely data is generated by models defined by each of the hypotheses. Analogous to the basic Bayesian Sets criterion, our new criterion that takes account of the negative relevance set should measure the ratio of these two quantities:

1. The probability of x and Q being generated by the same model with the same parameters θ, while {x, Q} and I are generated by models with different parameters
2. A sum of the following probabilities:
   a. The probability of x and I being generated by the same model with the same parameters θ, while {x, I} and Q are generated by models with different parameters
   b. The probability of x, Q and I being respectively generated by models with different parameters

The above is interpreted as:

$$scoreNRF_I(\mathbf{x} \mid Q) = \frac{p(\mathbf{x}, Q)p(I)}{p(\mathbf{x}, I)p(Q) + p(\mathbf{x})p(I)p(Q)}$$

This is in fact comparing the probabilities of different hypotheses given data (assuming equal priors for the hypotheses):

$$scoreNRF_I(x \mid Q) = \frac{p(H_1 \mid D)}{p(H_2 \mid D) + p(H_3 \mid D)} = \frac{p(D \mid H_1)}{p(D \mid H_2) + p(D \mid H_3)} = \frac{p(\mathbf{x}, Q)p(I)}{p(\mathbf{x}, I)p(Q) + p(\mathbf{x})p(I)p(Q)}$$

$$= \frac{p(\mathbf{x} \mid Q)p(Q)p(I)}{p(\mathbf{x} \mid I)p(I)p(Q) + p(\mathbf{x})p(I)p(Q)}$$

$$= \frac{p(\mathbf{x} \mid Q)}{p(\mathbf{x} \mid I) + p(\mathbf{x})}$$

$$= \frac{1}{score(\mathbf{x} \mid Q)^{-1} + \frac{score(\mathbf{x} \mid I)}{score(\mathbf{x} \mid Q)}} \qquad \textbf{(1.14)}$$

The final expression shows that the negative relevance feedback criterion given query Q and NRF set I can be expressed in terms of basic Bayesian Sets scores with regards to query Q and query I. The computation for calculating the NRF is simply twice the basic Bayesian Sets scoring. This is reasonable as two sets of information are provided by the user.

This expression is easily extended to multiple NRF sets $I_1, I_2 ... I_M$:

$$scoreNRF_{I_1, I_2 ... I_M}(\mathbf{x} \mid Q) = \frac{1}{score(\mathbf{x} \mid Q)^{-1} + \frac{score(\mathbf{x} \mid I_1)}{score(\mathbf{x} \mid Q)} + \frac{score(\mathbf{x} \mid I_2)}{score(\mathbf{x} \mid Q)} + ... \frac{score(\mathbf{x} \mid I_M)}{score(\mathbf{x} \mid Q)}} \qquad \textbf{(1.15)}$$

The direct application of NRF is user specified NRF. The user may feedback information through the application user interface, giving the system sets of items that he/she thinks is irrelevant, may as well be some he/she's just seen from the retrieval ranking. The NRF can be then applied to refine the search and return a more informative result.

The NRF can be applied in more interesting ways through a user interface. This is discussed in more detail in the Section 4.2 of this report.

1.5 Implicit Negative Relevance Feedback (iNRF) with Content Information

Having derived the basic Bayesian Sets algorithm and the Negative Relevance Feedback method, we propose a framework to apply Negative Relevance Feedback with implicitly decided NRF sets given some content training data.

Suppose we have the following data:

- The complete collaborative dataset (e.g. a complete movie-user dataset)
- A small knowledge(training) set containing specific content information of a fraction of the data items (e.g. genres of a small set of movies)

Both of these data can be made into 'incidence' binary data matrices in the way described in Section 1.1. Once we have the data matrix, we can run Bayesian Sets algorithm on both data matrices as illustrated by equation (1.13).

Given a query containing a few items, the content based iNRF, is able to perform a focused search for the specified feature corresponding to the knowledge (training) set in the way described as follows:

1. Query the training set matrix C with Bayesian Sets using some query Q to obtain a list ranked by relevance with regards to the specific content.
2. Fill a negative relevance set I with k lowest scoring items in the retrieved results, which are least likely to have the same content as Q, specified by C.
3. (Optional) Eliminate elements of I that match Q with content information. This step can be taken to use a supervised process.
4. Query the collaborative data with query set Q and NRF set I, using the NRF method.
5. Score and obtain the final ranking.

This process is illustrated in Figure 1.6.

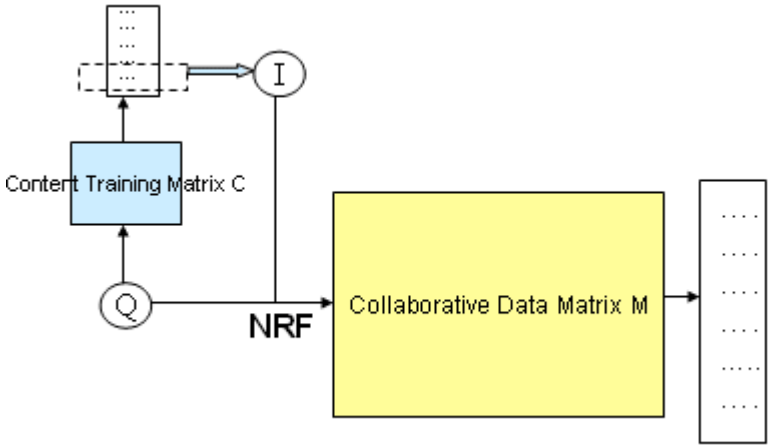

Figure 1. 6: iNRF illustration

The above formulation can be performed with a number of content sub-features such as (for movies) genre, cast, plot in parallel, each corresponding to a different NRF set $I_1, I_2 ... I_M$.

This is made possible with the NRF formulations we have described (equation 1.15). Using

this implementation, the collaborative information and content information are combined to improve the performance of the basic algorithm.

The iNRF is useful because the method generalizes information contained in the small training matrix to the retrieval process with the collaborative dataset. For instance, if we have a collaborative data set of people's preferences for movies, and the content training matrix is formed by incidences of genres with a small number of movies. The result of applying iNRF is supplying with a NRF set least likely to have the same genre as the query set. This NRF set (which can be chosen as small or large to tune the influence) references to collaborative data that defines the cluster in the collaborative data space of least likely genres. With the NRF framework previously derived, the scoring criteria ranks the data by how well they fit into the NRF hypothesis, thus allowing the genre information to be generalized to the much larger collaborative dataset in the retrieval process.

Furthermore, we can extend the idea of iNRF to building hybrid retrieval processes with different types of data. For example, the iNRF can be applied in reverse where content data is abundant whereas there is a small amount of collaborative data, and an implicit feed back from the collaborative data can be given to the prominent content search. This is applicable in the case where the user of the retrieval application has found some like-minded people whose opinions the user wishes to take into account in his/her search, give their collaborative data is accessible to the system.

With the movie retrieval implementation of this report, an implicit Negative Relevance Feedback method is constructed using genre information of movies. Given a specific query set of movies, and a particular need to look for the similar genre, the system is able to feed the algorithm with implicitly found negative relevance items. The implementation of this method is given in Section 2.5 and in Section 3 we experimentally evaluate the effectiveness of iNRF.

System Implementation

The implementation of this project is divided into three major areas: raw data specification and processing; MATLAB implementation, on which experimentation is performed; and the C++ implementation, the basis for an efficient application of Bayesian Sets.

2.1 Collaborative Data

In the scope of this report, a Beta-Bernoulli model is adopted to perform retrieval using Bayesian Sets. Ideal data should be formed into a sparse binary matrix of the size number of users by number of movies, where the element M_{ij} corresponds to a logical 1 or 0 representing whether user i likes movie j. We use a predefined orientation, users by movies, with the data matrix throughout this report for consistence.

Implementation of Bayesian Sets associated with this report is mainly based on the collaborative data obtained from the internet company Netflix. The data was made public available on 2nd Oct. 2006 for the Netflix Prize[3], a competition for 1 million USD prize for a Collaborative Filtering (user rating prediction) system that out-perform 10% of the current system used by Netflix.

The original Netflix dataset contains the user rating data of 17,770 movies/DVDs across 480,000 random users. The data is given in 17,770 text files, each containing rating information for the corresponding movie. Rating information is given in the form as user id, rating (from 1 to 5 stars), and date of rating. The size of the complete dataset is around 700 MB.

The titles, indices and year of production of the movies are given as a separate text file. This is extracted to form a dictionary for the movie titles.

The data was imported into a MATLAB sparse matrix of the size number of users by number of movies in many segments. The sparse matrix underwent significant pre-processing of the following steps:

1. Eliminate the movies which are rated by a number of users that is less than a threshold (e.g. 100). This process corresponds to collapsing the columns of the matrix
2. Eliminate the users which have number of ratings less than a threshold. This

corresponds to collapsing the rows of the matrix

3. Binarise data so that ratings of larger than 3 stars are translated into a logical 1 in the sparse binary matrix
4. Normalise the data matrix by each movie to account for movie popularity, i.e. all the binary rating data of a movie are divided by the total number of ratings of that movie

Data matrices of different sizes have been obtained, e.g. 13052x5080, 7772x2000, by different criteria for data pre-processing steps 1 and 2. These data matrices are all processed, tested by runs of the algorithm and ready for the use of further building an application. The large matrices, despite sparse, incur heavy computation overloads. The matrix used in the experimentation and illustrations of this report is of the size 5387x5751, formed to retain the least sparse parts of the data. Appendix A gives the form of Netflix raw data and the processed sparse matrix.

2.2 Content Genre Data

The content information of movies serves important purposes in experimentation, and implementation of the implicit Negative Relevance Feedback method. The content dataset was obtained from the Internet Movie Database (IMDB) [4]. The IMDB offers a wealth of content information for movies including genre, year, cast, country, language, etc. The content data obtained for this report is specifically genre information obtained from a text file in the format given in Appendix A. There are in total 28 genres.

The text files are read into MATLAB, matching for the titles of 5751 movies in our illustration matrix is performed. We choose to perform relatively strict matching criteria for the reliability of experiments. The titles for Netflix data have more condensed titles, so for all the 5751 movies:

1. If title shorter than 35 characters, find match for entire Netflix title in the IMDB data. If longer than 35 characters, match the first 35 characters.
2. Search for the year specified in the Netflix data in the IMDB title
3. If both cases satisfy, mark a match

Using the above method, 3163 out of 5751 movies have been found in the IMDB content data. The reasons for unmatched items are that the titles in the Netflix database are practically DVD/video titles, while the IMDB is a database of films and TV. There are non-overlapping items, and the strict criteria may miss titles in unconventional form. Nevertheless, the matched 3163 movies have good chances of being correct, and they have full genre information. This is sufficient for experiment use and implementing the content based iNRF.

From the IMDB genre data, we have formed a second binary data matrix of the size 28 by 3163. The Bayesian Sets algorithm can be run directly on this matrix. This matrix is an important basis for implementing the iNRF method, and essential for experimentation.

2.3 Direct Implementation of Bayesian Sets in MATLAB

As shown by equation (1.2) in the theory derivations, the algorithm can be implemented in a straightforward manner using MATLAB vectors and matrices:

$$\mathbf{s} = c + \mathbf{X} \cdot \mathbf{q}$$

The MATLAB code for the direct implementation can be found in Appendix B

The implemented Bayesian Sets algorithm can be run directly on the collaborative and genre data matrices. Examples of results are given in Figure 2.1 (top 10 most relevant).

Q={'Saving Private Ryan'}	Q={'Home Alone', '102 Dalmatians'}	Q = {'The Matrix', 'Starship Troopers', 'Aliens'}
'We Were Soldiers'	'Cinderella II'	'Final Fantasy: The Spirits Within'
'U-571'	'Home Alone 2: Lost in New York'	'Alien 3'
'Enemy at the Gates'	'The Kid'	'Event Horizon'
'Black Hawk Down'	'Beethoven's 3rd'	'Alien: Resurrection'
'Hart's War'	'Lady and the Tramp II'	'Screamers'
'Band of Brothers'	'Jack Frost'	'Wing Commander'
'The Program'	'The Flintstones'	'The Animatrix'
'K-19: The Widowmaker'	'The Jungle Book'	'Krull'
'Windtalkers'	'The Santa Clause'	'Soldier'
'Tears of the Sun'	'Max Keeble's Big Move'	'Battlefield Earth'

Figure 2.1: Example Results from the Bayesian Sets Implementation

2.4 Negative Relevance Feedback

The NRF is implemented using the expression given in equation (1.14) and (1.15) by calculating the basic Bayesian Sets multiple times. The user has the option of specifying a number of sets $I_1, I_2 ... I_M$ with movies that he/she thinks is irrelevant to the retrieval task.

The MATLAB code for performing NRF is given in Appendix B.

We try to show the intuitive effectiveness of NRF with an example run on our implemented system. The input process references movies by index, so takes seconds to complete. The first query is given by Q on the left below.

Q = { 'Finding Nemo', 'Shrek 2' }

Top returned items:

'Shark Tale' 'Brother Bear'

'Just Married' 'Racing Stripes'

'Cheaper by the Dozen' 'Robots'

'Alex and Emma'

'Raising Helen'

'The Longest Yard'

'How to Lose a Guy in 10 Days'

The user may think that the movie

'How to Lose a Guy in 10 Days'

is irrelevant to the query of children's animation movies, and puts this into a NRF set I1 together with a similar romance movie

'When Harry Met Sally'

The system runs NRF with

Q = { 'Finding Nemo', 'Shrek 2'}

I1 = {'How to Lose a Guy in 10 Days', 'When Harry Met Sally'}

The result for the NRF is:

'The Incredibles'
'Shrek '
'A Bug's Life'
'Monsters; Inc.'
'Toy Story'
'Aladdin'
'The Lion King'
'Shark Tale'
'Ice Age'
'Brother Bear'
'Lord of the Rings: The Return of the King'

The results have been refined as the user has added more negative relevance information.

If the user is yet not satisfied finding that

'Lord of the Rings: The Return of the King'

is not a typical children's animation and puts it into another NRF set I2

The system runs again with

Q = { 'Finding Nemo', 'Shrek 2'}

I1 = {'How to Lose a Guy in 10 Days', 'When Harry Met Sally'}

I2= {'Lord of the Rings: The Return of the King: Extended Edition'}

The system returns the following much more refined results with a clear hint of similarity:

'Brother Bear (Theatrical Widescreen Version)' 'Shark Tale' 'Ice Age' 'Shrek '
'Monsters; Inc.''Lilo and Stitch' 'Finding Nemo' 'A Bug's Life' 'Robots'
'The Lion King: Special Edition' 'The Incredibles' 'Elf' 'Mulan'
'Spirit: Stallion of the Cimarron' 'The Emperor's New Groove' 'Aladdin'
'Beauty and the Beast' 'The Fox and the Hound' 'Brother Bear (Home Viewing Version)'
'Patch Adams' 'Pocahontas'

2.5 iNRF

The iNRF implementation with the collaborative and genre data for movies is given as follows:

- A knowledge (training) matrix C for iNRF is obtained by randomly selecting 500 movie vectors from the genre matrix (28x3163) obtained in Section 2.2. We use a smaller matrix to test the system's ability to generalize.
- For each query of iNRF, the query set is scored with regards to matrix C. Lowest scoring items are formed into a negative relevance set. We adopt an unsupervised feedback process i.e. the step 3 in Section 1.5 is not taken. It is proved in the experimentation section that this gives stronger generalization results.
- The collaborative data matrix is queried using NRF to obtain final score and ranking

The significance of implementing the iNRF is two fold. Firstly, iNRF is evaluated with experimentation to test the effectiveness of Negative Relevance Feedback. Secondly, the iNRF is a useful retrieval method itself and shows good performance as evaluated in the experimentation section.

2.6 Clustering

If the query is not well classified, i.e. it is a combination of movies without a clear common feature, the results returned from the algorithm can be uninformative. This can occur when for instance, when a user inputs all the movies he/she likes, but these movies do not have obvious similarity, or in fact, the pattern is too complex for the probabilistic model assumed.

This can be remedied using clustering, which is a way to raise the model complexity.

Clustering can be applied in two different ways. Firstly, clustering can be applied in a conventional information retrieval approach, where all the data is subject to this process and a classification of data can be obtained using an iterative algorithm such as K-means. Moreover, a more complex process called hierarchical clustering can be applied, and a classification hierarchical structure can be obtained. These predefined classification information can be used to filter and refine result, or to better inform the user. The second approach, which we implemented, is clustering the query set.

It has been discussed that the Bayesian Sets algorithm essentially finds clusters of data in the data-space it is applied to. It is therefore straightforward to adopt a simple clustering algorithm to cluster the query data points.

As clustering is not related to the main results of this project and report, the details of theoretical aspects are not discussed. A copy of MATLAB implementation code is included in Appendix B. This is for the reference of future implementation of the Bayesian Sets retrieval system, as clustering should be a useful aspect in implementing an application.

2.7 C++ Implementation of Bayesian Sets

The purpose of implementing the Bayesian Sets algorithm is to work towards a hard-coded application. This report discusses the implementation of a C++ program to load data matrices and perform the basic Bayesian Sets scoring computations, which can be the basis of further implementation. We have chosen the C++ language for speed considerations. The algorithm needs to be applied on large matrices, and speed is crucial for an information retrieval application.

The implementation of the Bayesian Sets algorithm in C++ is performed in a Linux Environment with the GNU g++ compiler. As the data required by the algorithm needs to be stored and manipulated as sparse matrices, a sparse matrix library SparseLib++ [6] is used.

With the C++ implementation, an average of 420 milliseconds of query time is obtained for with a sparse binary matrix of the size 10099x17770 processed from the Netflix data. This is compared to more than 2 seconds of query time for the same matrix in MATLAB.

Experimentation

The experiments presented in this report focus on evaluating the performance of the Bayesian Sets information retrieval algorithm and extensions. We compare to a number of competing algorithms. A standard experimentation method for information retrieval, the precision-recall curve, is used through out this section. In designing the tests, this report provides detailed aims, specifications and implementation to justify our results.

The method used in our experimentation is data oriented in line with the evaluation practice of information retrieval. We do not use user tests due to the difficulty of sustaining objectivity.

In highlight, the experimentation discussed in this report provides evidence for the basic algorithm's ability to:

1. Retrieve items similar in *content* with the query by querying collaborative data, i.e. ability to generalise content from collaborative information.
2. Generalise the data clusters specified by *common content features* in the query.

Also, we further evaluate a content-collaborative hybrid method, the implicit Negative Relevance Feedback with comparison to competing algorithms. The results are evidence of

1. Performance of Negative Relevance Feedback framework derived in Section 1.4.
2. The merits of the iNRF implementation for content focused search.

3.1 Evaluation in Information Retrieval

The evaluation of information retrieval methods is based on concrete test designs. Before experiments are designed to evaluate our retrieval methods. It is useful to look at the basic concepts in evaluating information retrieval: Precision and Recall.

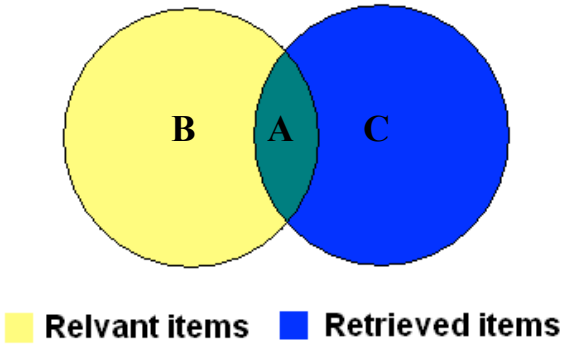

Figure 3.1: Classification used to define Precision and Recall

Given a query, and if it is possible to define a criterion that can categorize which items in the dataset are relevant and which are not, a classification of items as shown in Figure 3.1 can be obtained for each run of a specific algorithm.

In the illustration of Figure 3.1:

- A is the set of items that are both retrieved by the algorithm and relevant to the query
- B is the set of items that are relevant to the query, but not retrieved by the algorithm
- C is the set of items that are retrieved but not relevant to the query.

Once the sets A, B and C are defined, Precision and Recall are defined as the ratios:

$$Pr = \frac{A}{A+C} \qquad \text{Re} = \frac{A}{A+B} \qquad \textbf{(3.1)}$$

Precision = Number of retrieved, relevant items/the total number of retrieved items
Recall = Number of retrieved, relevant items/ the total number of relevant items to the query

For a given retrieval algorithm that gives ranked results, and a specific definition of relevance, Precision is decreasing function of Recall. This is illustrated in the following example.

Given a query, the corresponding ranked retrieval results can be classified as either relevant or irrelevant, represented by 1's and 0's. We can also evaluate the total number of items in the database that's relevant to the query. In the following example as shown in Figure 3.2, assume that we've determined the relevance of the ranked results, and found the total number of relevant items in the dataset is 10. We can calculate Precision and Recall pairs for each ranking index.

Rank	Relevant?	A	A+C	A+B	Precision	Recall
1	1	1	1	10	1	0.1
2	1	2	2	10	1	0.2
3	0	2	3	10	0.67	0.2
4	1	3	4	10	0.75	0.3
5	0	3	5	10	0.6	0.3

Figure 3.2: Calculation or Precision and Recall

The plot of Precision against Recall is in the form of Figure 3.3, where Precision jumps for every correctly retrieved item. It is desirable to reduce the sudden changes in direction and the standard way to do this is interpolate precision at any recall level R as the maximum precision found for any recall level r that's higher than R:

$$\text{Pr}_{\text{int}}(R) = \max_{r>R} \text{Pr}(r) \qquad \textbf{(3.2)}$$

The interpolated Precision values are shown as red in Figure 3.3.

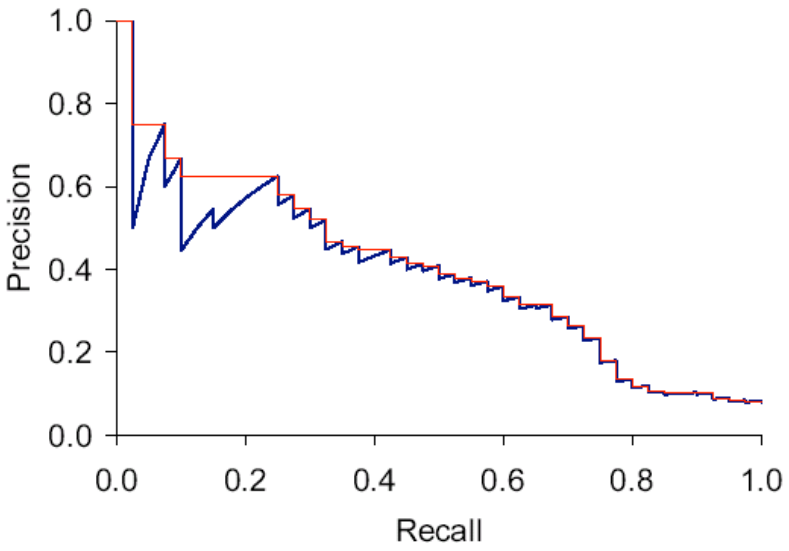

Figure 3.3: Example Precision-Recall Curve for a single query. ([3] Page 119)

The performance of an information retrieval algorithm is evaluated by the best trade off between precision and recall. As shown in Figure 3.4, the 'higher' the curve is, the better the system performance. Given a certain fixed value, a superior algorithm gives a higher precision.

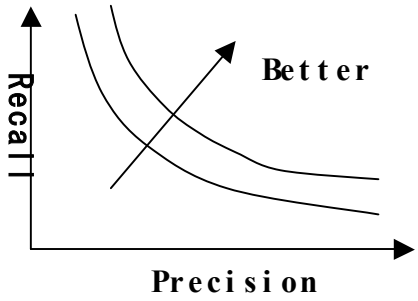

Figure 3.4: Evaluation of performance using P-R curves

We will use the evaluation method of Precision-Recall curves to evaluate the Bayesian Sets algorithm and its extensions.

3.2 Competing Algorithms

Experiments are performed in comparison of the following sets of algorithms:

- Bayesian Sets Basic Algorithm
- Bayesian Sets implicit Negative Relevance Feedback (iNRF)

- K-Nearest Neighbour (KNN)
- K-Nearest Neighbour Mean(KNNM)
- GroupLens Algorithm for Collaborative Filtering with Netnews [7]

The guidelines for choosing comparable algorithms are:

1. Select algorithms that define relevance for Information Retrieval or one that can be modified from a Collaborative Filtering method
2. Compare the Bayesian Sets algorithm to the geometric algorithms, such as KNN

We now give the details for implementations of KNN, KNNM, and the GroupLens algorithm. The implementation of Bayesian Sets and iNRF were given in then previous sections.

Implementation of KNN on Binary Collaborative Data

The data is represented by multi-dimensional binary vectors. The K-Nearest Neighbour algorithm is implemented so that the data vectors are ranked by their Euclidean distances to the query set.

The data matrix is firstly normalized so that:

$$M_{norm}(i,j) = \frac{M(i,j) - \mu_i(j)}{\sigma_i(j)} \qquad \textbf{(3.3)}$$

Where μ_i is the mean vector across user i, indexed by movie j, and σ_i is the standard deviation across user i.

Given a query Q, the KNNM algorithm is given by:

```
Compute the mean vector m_Q across all items in Q;
Calculate and rank the Euclidean Distance d between every item in the data set and m_Q;
Rank items by descend sort of d;
```

The KNN algorithm is given by:

```
For all the data-vectors x in the data set:
{
Calculate and rank the Euclidean Distance d between x and every data-vector in Q, x_Q;
Choose the shortest distance d to elements in set Q, as the distance D from x to Q
}
Rank items by descend sort of D;
```

The KNNM and KNN algorithms are straightforward to implement in MATLAB.

Implementation for GroupLens Algorithm

The GroupLens Algorithm we refer to in this report is the Collaborative Filtering method presented by P.Resnick et.al (1997) [7]. The algorithm is considered to be suitable for comparison because it was developed for user-rating collaborative data. It is also a geometric algorithm suitable to perform retrieval with our collaborative movie data. The GroupLens Algorithm is essentially an improvement on KNN.

Its original implementation is explained in [7] and was applied to discrete user rating data (e.g. rating from 1-5). Using a user item matrix of ratings, we'd like to predict the rating user a would give for item p.

First calculate the Pearson Correlation for all the users with regard to a, which is a measure of how likely are user a and b to agree with each other [7]:

$$r_{ab} = \frac{\mathrm{cov}(a,b)}{\sigma_a \sigma_b} \qquad \textbf{(3.4)}$$

The prediction of user a's rating to item p is given by:

$$R_{ap} = \mu_a + \frac{\sum_i (R_{ip} - \mu_i) r_{ai}}{\sum_i |r_{ai}|} \qquad \textbf{(3.5)}$$

where μ_i is the average rating for user i across all items.

This algorithm can be easily modified to apply to our pre-processed and normalised collaborative data by creating a pseudo-user (vector with number of movie dimensions) who has preferences for only the movies in the query. We then try to predict the pseudo-user's ratings for all the other movies, and obtain a ranking result. The computation can also be formulated into a vector-matrix multiplication, although a full matrix rather than sparse needs to be stored. The MATLAB code for performing this algorithm is given in Appendix B.

3.3 The General Genre Test with Single Item Query

Figure 2.1 shows example results returned from the Bayesian Sets algorithm. We propose an experimental method using genre information to evaluate these results.

QUERY	TOP RESULT
NAME:	NAME:
'Saving Private Ryan'	'We Were Soldiers'
YEAR	YEAR
'1998'	'2002'
GENRES:	GENRES:
'Drama'	'Drama'
'Action'	'Action'
'War'	'War'
	'History'

Figure 3.5: Detail Comparison between Query and Retrieval Result

Figure 3.5 gives the detailed genre information for the query of example 1 in Figure 2.1 and the movie ranked as most relevant. It can be observed that a useful measure for the similarity of movies, given any two movies, is their genre patterns. In this example, three of the genres match, and because the two are both 'War', 'Action' and 'Drama' movies, they can be seen as similar by a major opinion. With this perspective, we aim to evaluate the algorithm's effectiveness in satisfying users' requests to find 'similar' items.

Having obtained the genre information from IMDB, we use this data to define 'relevance' in order to obtain the classification shown in Figure 3.1 required to compute Precision and Recall.

Aim of Experiment

Evaluate the Bayesian Sets, iNRF algorithms in their general performance of returning similar movies measured by Precision-Recall.

Specification

Only the 3167 movies with genre information (described in Section 2.2) out of 5751 are used so that each item's relevance to the query can be evaluated. The test queries contain only 1 element and are drawn from a filtered list of movies having more than 3 genres. These queries are run on the collaborative data.

Relevance measure between movies is approximated by genre match. We assume the user making queries is looking for genre matches, and define the relevance criterion:

A movie in the database is relevant to the query if it has
Test 1: at least 1 match of genre
Test 2: at least 2 matches of genre
Test 3: at least 3 matches of genre

In order to compute Precision-Recall, it is required to find the number of all the relevant movies in the database. This is done by matching genres and can be achieved in one vector-matrix multiplication in MATLAB.

All the competing algorithms are evaluated in this test. Note for single items as query, KNN and KNNM are equivalent.

Implementation

The experiment is run on an automated program in MATLAB. For each query, a *single* query is draw from the test set**.** The results returned by the algorithms are judged with the genre relevance criteria and a binary vector indicating relevance can be obtained. Having obtained this vector, the method described in Figure 3.2 can then be used to compute Precision and Recall. The Precision values are interpolated using equation 3.2 for a fix grid of recall. These Precision values are averaged over a sufficient number of runs.

Results

All results are obtained averaging over 50 runs of each algorithm.

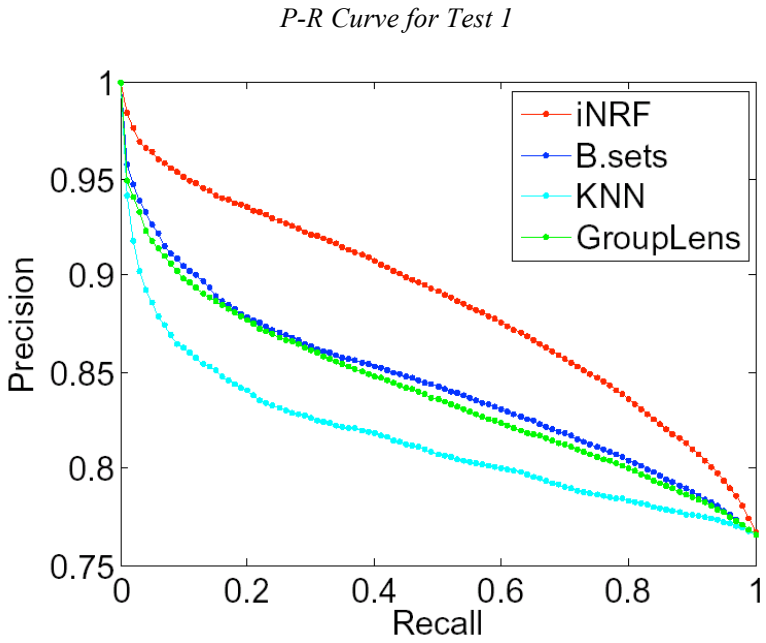

Figure 3.6: Precision Recall for General Genre Test (Match 1)

P-R Curve for Test 2

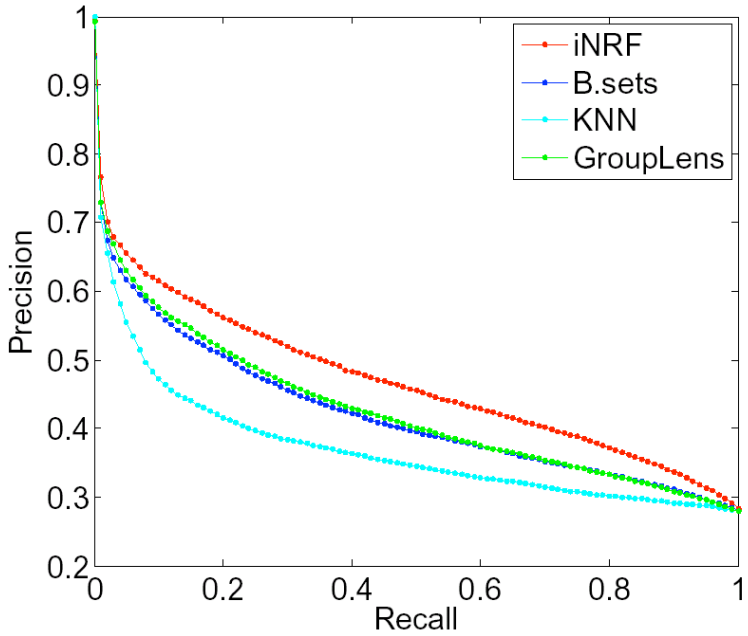

Figure 3.7: Precision Recall for General Genre Test (Match 2)

P-R Curve for Test 3

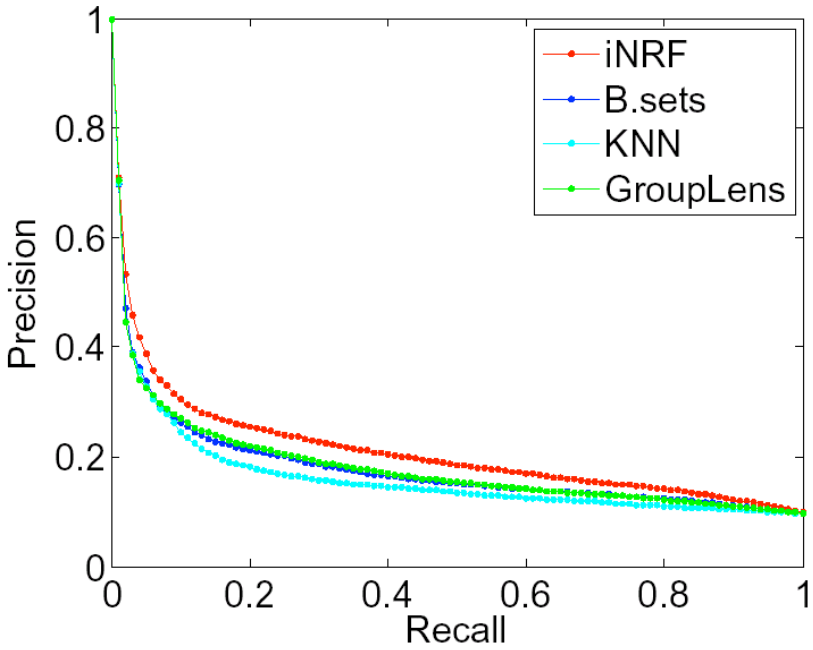

Figure 3.8: Precision Recall for General Genre Test (Match 3)

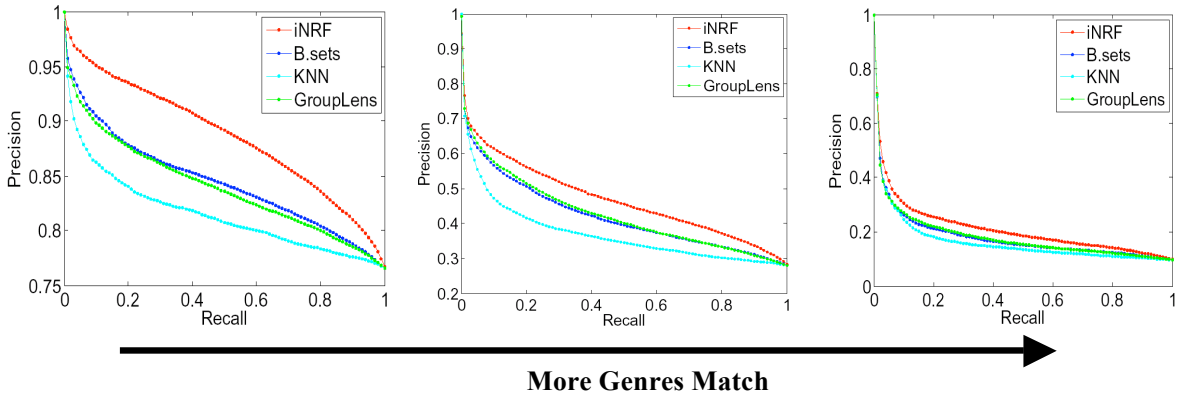

Figure 3.9: Alignment of results for General Genre Test

Observation and Discussion of Results

- The relevance criteria in the three tests are made stricter as the demanded number of matched genres increase. This results in the decrease of precision give a certain recall for all algorithms.

- The Precision at the ending point of P-R curve gives the fraction of number of relevant movies to the total number of movies. For Test 1, this fraction is near 0.77. The two other tests give fairly small retrieval fractions, suggesting the retrieval problem is harder and the system is retrieving a small number of relevant results for the user.

- The Bayesian Sets and iNRF algorithms outperform the KNN algorithm significantly in all three tests. The higher P-R curves indicate that given a fixed recall, or a specific ranking index, Bayesian Sets and iNRF return more relevant items. Therefore Bayesian Sets and iNRF should be more useful than KNN to a user's retrieval task.

- For the single query and all three tests, the Bayesian Sets algorithm matches closely with the GroupLens algorithm. However, for single queries, i.e. a single example data point, the system is under-informed and differences will emerge once given more data points. This is shown in the next experiment.

- The iNRF outperforms GroupLens and basic Bayesian Sets algorithm significantly in all three tests. This give indication that

    1. Given the negative relevance set selected for iNRF is reasonable, the Negative relevance method is effective in refining results and improving algorithm performance. This result for NRF is obtained implicitly without the need for a user trial.

    2. The iNRF uses the genre information of 500 hundred movies to improve the retrieval in genre of 3000 movies. The feedback is also unsupervised as stated in section 2.5. This shows the iNRF has good generalization with collaborative data.

3. When the Bayesian Sets system is under-informed, the iNRF is able to provide improvement to reinforce the Bayesian Sets algorithm for the case of few query data points.

3.4 The Multiple-element Query Test

It has been shown that with single query element, the GroupLens algorithm matches Bayesian Sets in performance. However, the strength of Bayesian Sets is in multiple element queries. With more example data points in one cluster, the algorithm is able to predict better other data points that belong to the same cluster.

We design the multiple element query test to give evidence for this observation. We try to show that from collaborative data, the Bayesian Sets algorithm and its extension iNRF are capable of generalizing common content features in the query set.

Aim of Experiment
Evaluate the performance of algorithms in identifying the similarities shared in the query set.

Specification
The data and test set of movies are as defined in the last experiment. We use a step-up comparison of K=1, 3, 6, 10 data points in the query. For every value of K, first randomly choose a fixed genre G from the list of 28. K movies are then chosen randomly to form the query from this same genre. They may have other genres apart from the chosen, but all K movies have 1 genre G in common.

Relevance criterion: a retrieved movie is considered relevant to the query if it has the genre G. This experiment assumes the type of user that requires the system to return items with the common similarity found in query.

Implementation
Again the experiment is performed with a MATLAB program. The interpolation method for Precision described in the previous experiment is also used.

Results
All results are obtained averaging over 50 runs of each algorithm

K = 1 (this result is added as a benchmark against higher values of K)

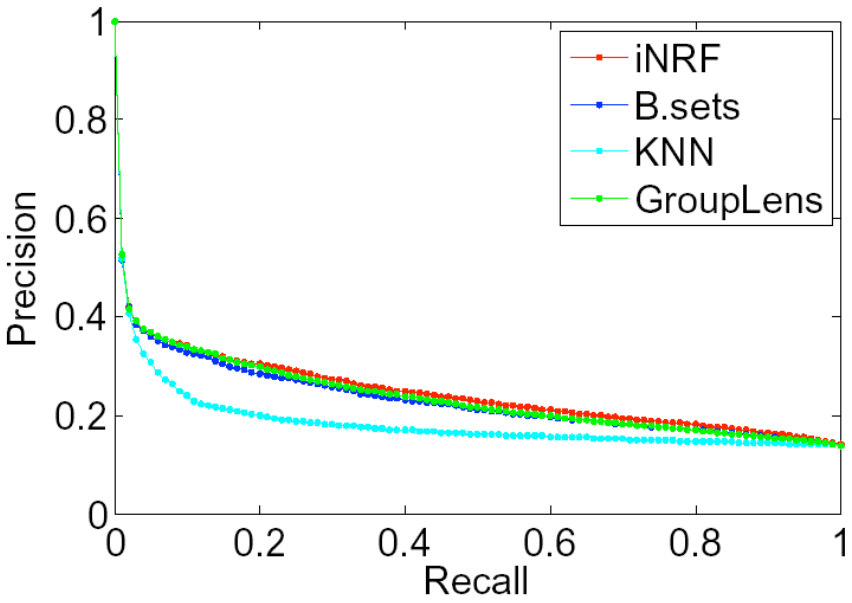

Figure 3.10: Result for 1 element query to retrieve movies with a chosen genre

K=3

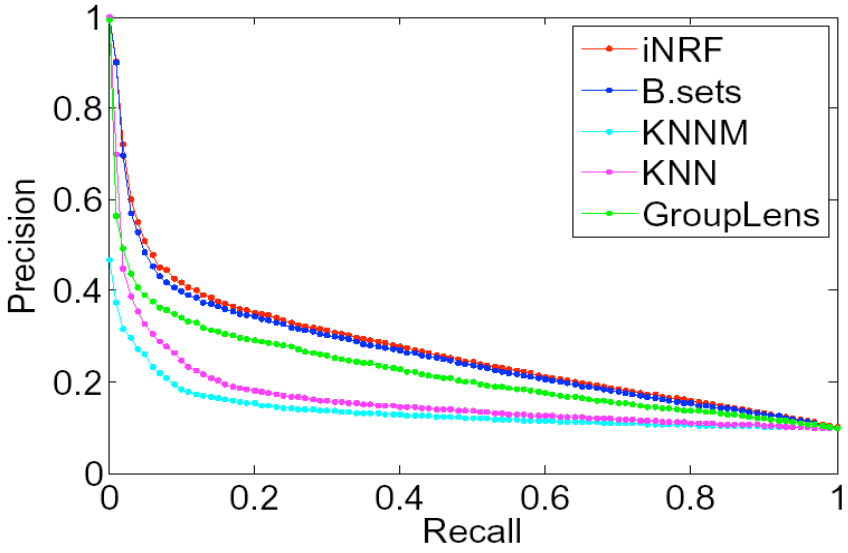

Figure 3.11: P-R curves for 3 query elements to retrieve movies with the chosen common genre

K=6

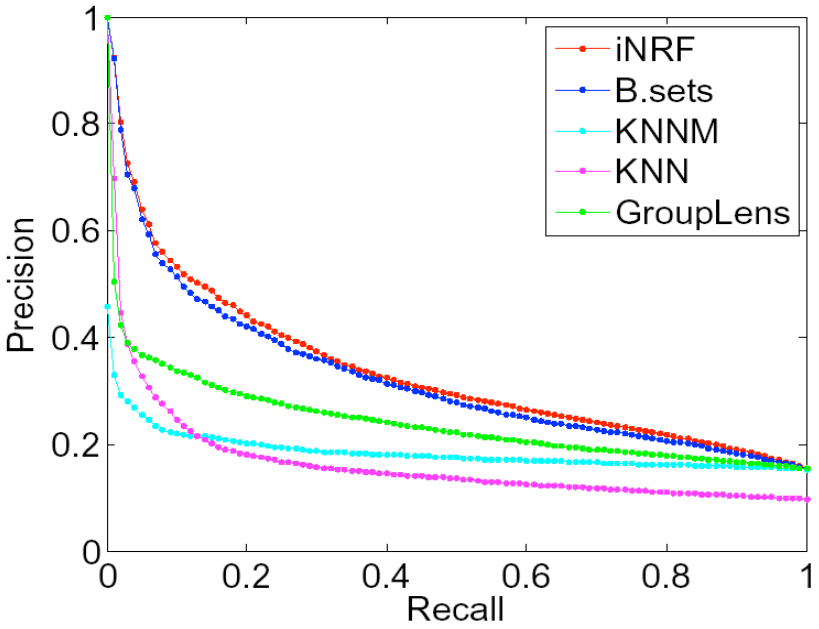

Figure 3.12: P-R curves for 6 query elements to retrieve movies with the chosen common genre

K=10

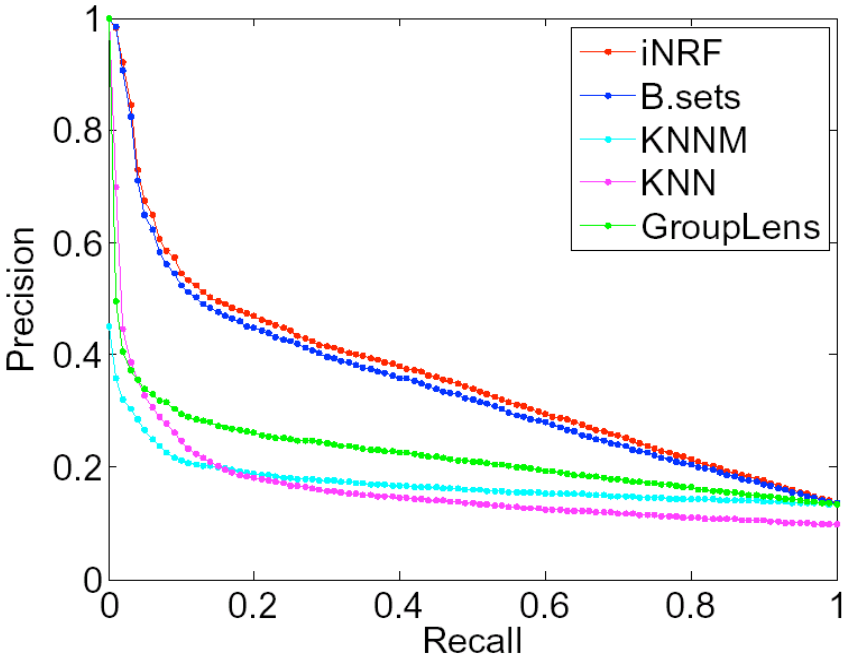

Figure 3.13: P-R curves for 10 query elements to retrieve movies with the chosen common genre

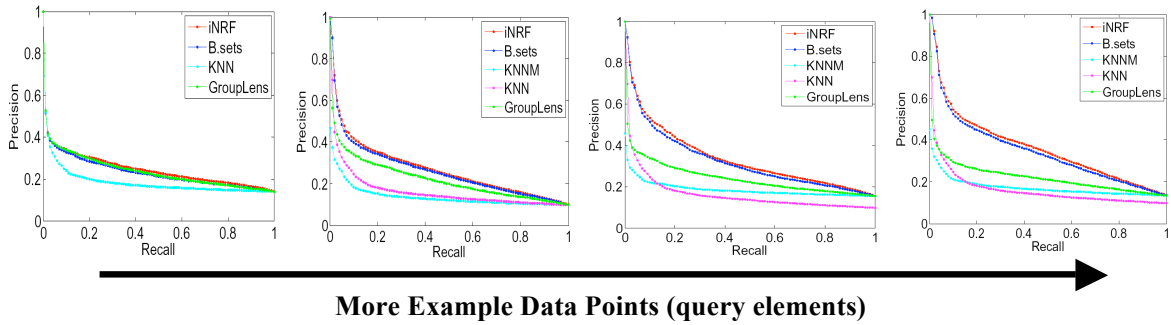

Figure 3.14: Aligned Results for Multiple Element Query

Observation and Discussion

- From Figure 3.10, the three algorithms GroupLens, Bayesian Sets and iNRF start from matching performance. As the number of data points in the query increases, Bayesian Sets and iNRF quickly pick up the cluster defined in the multiple element query and improve performance in retrieving movies with the common genre in query set.
- Contrast to the previous experiment, given more query elements, Bayesian Sets exhibits superior retrieval results against GroupLens algorithm in retrieval precision for this task, and further outperforms KNNM and KNN.
- It is observed from Figure 3.14 that the Bayesian Sets algorithm's performance of generalizing the common query concept improves with the increasing number of data-points, while the geometric algorithms stay unaffected. This proves our theoretical results and discussion in Section 1 of the algorithm's strength on finding clusters.
- The iNRF retains its improvements in performance with regard to Bayesian Sets. The improvements are not as significant as the previous experiment. This is because the more strict relevance criterion of matching one specific genre. Nevertheless, with negative relevance feedback, the iNRF retains ability to generalize common query concept yet still raises precision with respect to Bayesian Sets.

Conclusions and Future Work

4.1 Conclusions

In this report, we have referenced and introduced the Bayesian Sets algorithm for information retrieval, derived two extensions to the basic algorithm, implemented the binary data formulation for the basic algorithm and its extensions using the Netflix movie collaborative data and IMDB movie genre data. We also performed experimentation using information retrieval evaluation methods to justify the performance of our theoretical work.

The conclusions of this report and the project are summarized as follows:

Theory

- A comprehensive study of the Bayesian Sets algorithm has been conducted, and a good understanding of the underlying theory is obtained.
- An extension of the Bayesian Sets algorithm, the negative relevance feedback, has been developed through this project.
- A specific method for building hybrid retrieval systems with data of different origins, the implicit Negative Relevance Feedback method was introduced through this project.

Dataset

- The Netflix collaborative dataset of 480,000 users by 17770 movies was processed and used throughout implementation. Pre-processed matrices of different sizes are ready to be used in applications.
- Genre data was obtained for 3167 movie titles in the Netflix dataset from IMDB.
- A combined processed dataset used in this project will be made available for research use.

Implementation

- The basic Bayesian Sets algorithm have been implemented in both MATLAB and C++
- All versions of all the methods related to Bayesian Sets, negative relevance feedback, data and query clustering, iNRF, have been implemented and tested in MATLAB
- Around 150 pieces of MATLAB code have been produced during this project

Experiment Results

- From the results to Section 3.3 and 3.4, we can conclude that, in generalizing content from collaborative data, the basic Bayesian Sets algorithm outperforms the K-Nearest Neighbour algorithm with both single and multiple element queries. The basic Bayesian Sets algorithm matches with the performance of our implementation of GroupLens algorithm with under-informed single element queries, but significantly outperforms GroupLens with multiple element queries. This illustrates the strength of Bayesian Sets

at finding data clusters or item 'concepts'.

- In both the results to Section 3.3 and 3.4, the iNRF method outperforms all the algorithms we have studied in generalizing content from collaborative data. The iNRF significantly outperforms the basic Bayesian Sets algorithm with under-informed queries. We can conclude that iNRF is effective in retrieving with a content feature focus, and is especially useful when the system supplied with few data points in the query. The performance of iNRF illustrates the effectiveness of the Negative Relevance Feedback Framework.

4.2 Future Work: Proposed Interfaces for Retrieval Application

The Bayesian Sets algorithm is well equipped with methods to apply in a retrieval interface: negative relevance feedback, clustering and hybrid retrieval with content information. Also, the data is readily available (Netflix and IMDB) and an initial C++ implementation has been made. The next step of this project might be to develop an interface for the retrieval application. A few of the interface ideas are given below.

NRF

As illustrated in Section 2.4, NRF is a useful method to add 'reasoning' to information retrieval. With 2 NRF sets, the system can return clearly oriented results. However, it is troublesome for the user to type in the movie items. It would be helpful to produce a click-and-drag interface with a number of windows to which the user can drag items from his/her collection. The user is able to trial and error with different combinations of NRF sets, and will be able to quickly find the correct model hypothesis. An illustration is given in Figure 4.1.

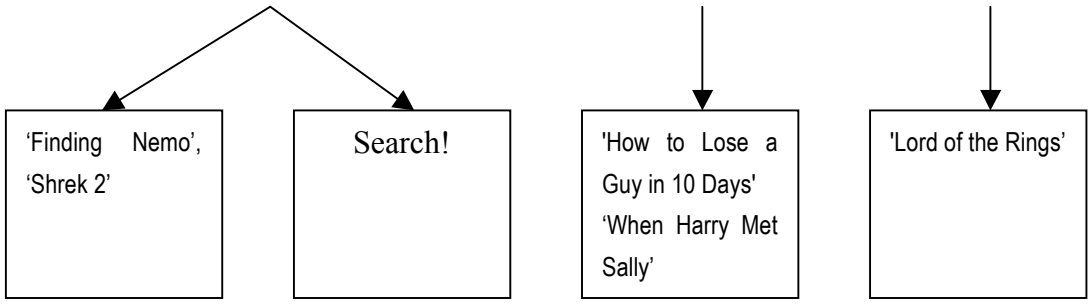

Figure 4.1: Proposed interface for NRF

Clustering

As discussed in Section 2.6, clustering can be applied to the query if the query is large and/or without a clear common feature. Suppose the query is processed into M clusters. In each cluster elements should be relevant to each other and form reasonable queries. The retrieval results can be given by a parallel display of the respective query results of the M clusters

individually. The user is informed of which results are given from which of the M clusters.

iNRF

Consider a movie retrieval system for instance, with iNRF, one can tell the system to search for films similar to 'The Godfather' with a preference direction to 'cast' if say the user likes the movie because of the actors/actresses. The system uses only a small amount of data, the knowledge dataset of casts for a fraction of all the movies, to apply an implicit feedback to the search in the complete dataset. Thereafter, the system will rank high the movies with a similar cast as ‘The Godfather’ on top of the score from collaborative data. This will achieve higher precision toward the user's needs, as the user has specified more information. Note the system does this without knowing any content information about the movies it is trying to score.

As mentioned before, using the Negative Relevance Feedback formulations we have derived, our system should be able to perform iNRF with a multitude of content features by specifying a multitude of NRF sets.

Moreover, with some further implementation, the user is able to specify to what extent he/she may emphasize on a certain feature. The system can enable this by linking this specific 'extent' quantity with the amount of relevance feedback the system receives, e.g. the number of items in each negative relevance set. The user is then able to use his/her own search initiatives to filter through results and change search preferences.

References

[1] P.Lyman et al., *How Much Information? 2003*, University of California Berkeley

[2] Z.Ghahramani and K.A.Heller, *Bayesian Sets*, NIPS, 2005

[3] The Netflix Prize, www.netflixprize.com

[4] Internet Movie Database, www.imdb.com

[5] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008.

[6] J.Dongarra, A.Lumsdaine, R.Pozo, K.Remington, *A Sparse Matrix Library in C++ for High Performace Architectures*, Preceedings of the Second Object Oriented Numerics Conference, 1994

[7] P.Resnick, N. Lacovou, M.Suchak, P.Bergstrom, J.Riedl, *GroupLens:An Open Architecture for Collaborative Filtering of NetNews*, Preceedings of ACM 1994 Conference on Computer Supported Cooperative Work

Appendix A: Format of Raw Data

Netflix Collaborative Data: In the format of user index, rating and rating date.	**Pre-processed and Normalised Sparse Matrix for the Bayesian Sets Algorithm**	**IMDB Genre Data Example**	
1488844,3,2005-09-06	(96,83) 0.0004	Matrix Revolutions, The (2003)	Action
822109,5,2005-05-13	(97,83) 0.0004	Matrix Revolutions, The (2003)	Sci-Fi
885013,4,2005-10-19	(25,84) 0.0016	Matrix Revolutions, The (2003)	Thriller
30878,4,2005-12-26	(52,84) 0.0016	Matrix, The (1999)	Action
823519,3,2004-05-03	(90,84) 0.0016	Matrix, The (1999)	Thriller
893988,3,2005-11-17	(97,84) 0.0016	Matrix, The (1999)	Sci-Fi
124105,4,2004-08-05	(25,85) 0.0018	"Matroesjka's" (2005)	Crime
1248029,3,2004-04-22	(29,85) 0.0018	"Matroesjka's" (2005)	Drama
1842128,4,2004-05-09	(33,85) 0.0018	"Matroesjka's 2" (2007) (mini)	Drama
2238063,3,2005-05-11	(38,85) 0.0018		
1503895,4,2005-05-19	(41,85) 0.0018		
2207774,5,2005-06-06	(45,85) 0.0018		

Appendix B: MATLAB Codes

The Basic Bayesian Sets IR Scoring Algorithm

```
%The matrix to run the algorithm is given in TempM
%Query set is given by Q(1xn vector), alpha, beta are pre-computed
%Please refer to the expressions derived in Section 1.
QueCut= TempM(:,Q);
CompleteSum=size(Q,2);
QueSum = sum(QueCut, 2);
alphatilde=alpha+QueSum;
betatilde=beta+CompleteSum-QueSum;

c=sum(log(alpha+beta)-log(alpha+beta+CompleteSum)+log(betatilde)-log(beta)
q=log(alphatilde)-log(alpha)-log(betatilde)+log(beta);

s= c+(TempM')*q;
scr=exp(s);
% Sort score
[final, IX]=sort(scr);
%Display movie names
Names(IX);
```

The Query-Clustering Algorithm

```
%The query to cluster is given in Qc, data matrix is TempM

qmovlist = 1:size(Qc,2); % This variable stores what cluster each element currently belongs to
liststore = zeros(1, size(Qc, 2));
change =1;
iteration=0;
%Find the limit for stopping the clustering algorithm, as the minimum of the maximum score with any
other data vector so first need to find the maximum list for all query elements
testMaxStore=zeros(1,size(Qc,2));
for i = 1:size(Qc,2)
    i
    if(sum(TempM(:,Qc(i)))==0)

        qmovlist(i)=0;
    else
        mov = Qc(i);
        record = qmovlist(i);
        qmovlist(i)=0;
```

```
        clear Q;
        clear scrboard;
        for j = 1:size(Qc,2)
            binlist = (qmovlist == j);
            Q = Qc(reflist(binlist));
            if(Q)
                scoresingle;
             %Score and store the scores across all other items in the query in scoreboard
                scrboard(j)=scr;
            else
                scrboard(j)=0;
            end
        end
        [Max,Clu]=max(scrboard);
        Max
        testMaxStore(i)=Max;
        qmovlist(i)=record;
    end
end
while(change)  %stop if there is no change of clusters for all the elements
    change = 0;

    iteration = iteration +1;
for i = 1:size(Qc,2)
    i
    if(i>1)
        qmovlist(i-1)
    end
    if(sum(TempM(:,Qc(i)))==0)
        qmovlist(i)=0;
    else
        mov = Qc(i);
        record = qmovlist(i);
        qmovlist(i)=0;
        clear Q;
        clear scrboard;
        for j = 1:size(Qc,2)
            binlist = (qmovlist == j);
            Q = Qc(reflist(binlist));
            if(Q)
                scoresingle;
                scrboard(j)=scr;
            else
                scrboard(j)=0;
```

```
            end
        end
        [Max,Clu]=max(scrboard);
        Max
        if(Max>min(testMaxStore))
            qmovlist(i)=Clu;
        else
            qmovlist(i)=record;
        end
        if(qmovlist(i)~=record)
            change=change+1;
        end
    end
end
end
liststore = cat(1, liststore, qmovlist);
end
```

The GroupLens Algorithm

```
%load stdstore (standard deviation across movies, an element for each user)
%load M (offsetMatrix minus mean across movies)

queryusr = zeros(size(M,2),1);
for i =1:size(Q)
    temp = 1/nnz(TempM(:,Q(i)));
    queryusr(Q(i))=temp;
end

r = zeros(size(TempM,1),1);
rvector=queryusr-mean(queryusr);
rstd=(var(queryusr)^0.5);

r=(M*rvector)./(rstd*stdstore);

scr = r'*M./(r'*r);

[final,IX]=sort(scr');
```

Negative Relevance Feedback

```
%Please refer to NRF expression given in report Section 1
%Query is given in Qc, NRF sets I1, I2, I3
Q = Qc;
```

```
score;
rel = scr;
if(I1~=-1)
   Q=I1;
   score;
   irel1= scr;
   if(I2(1)~=-1)
      Q = I2;
      score;
      irel2=scr;

      if(I3(1)~=-1)
         Q = I3;
         score;
         irel3=scr;

         scr = 1./(1./rel+irel1./rel+irel2./rel+irel3./rel);
      else
         scr = 1./(1./rel+irel1./rel+irel2./rel);
      end
   else
      scr = 1./(1./rel+irel1./rel);

   end
else
   scr = 1./(1./rel);
end

[final,IX]=sort(scr);
```