2012

The Feasibility of Implementing a Collaborative Recommender System for the Brigham Young University Harold B. Lee Library

Derrick Brinton
derrickbrinton@gmail.com

Follow this and additional works at: https://scholarsarchive.byu.edu/studentpub
Part of the Library and Information Science Commons

The Library Student Research Grant program encourages outstanding student achievement in research, fosters information literacy, and stimulates original scholarship.

BYU ScholarsArchive Citation
Brinton, Derrick, "The Feasibility of Implementing a Collaborative Recommender System for the Brigham Young University Harold B. Lee Library" (2012). All Student Publications. 83.
https://scholarsarchive.byu.edu/studentpub/83

This Report is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Student Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
The Feasibility of Implementing a Collaborative Recommender System
for the Brigham Young University Harold B. Lee Library

Derrick Brinton
## Table of Contents

Abstract..................................................................................................................................................iv

Introduction..............................................................................................................................................1

  Challenges Currently Facing the Harold B. Lee Library at BYU.................................................2
  The History of Collaborative Recommender Systems.............................................................2

Implementation........................................................................................................................................3

  General Mechanics of a Recommender System........................................................................3
  BYU Library Recommender System.........................................................................................6

    Formatting the Data......................................................................................................................7

    Massive Amounts of Data.........................................................................................................9

    Updating the Database..........................................................................................................10

    Personalized Recommendations..........................................................................................11

Recommendations..............................................................................................................................14

Future Work.........................................................................................................................................14

Appendix A.........................................................................................................................................15

Appendix B.........................................................................................................................................16

References............................................................................................................................................17
List of Figures

Figure 1: User-based correlations........................................................................................................4
Figure 2: Item-based correlations........................................................................................................5
Figure 3: Screenshot of strange recommendations from Amazon.com..............................................6
Figure 4: Heirarchy of item groupings..................................................................................................9
Abstract

With library users finding their information from other non-library sources, the Harold B. Lee Library at Brigham Young University has great need of a recommender system. The library already has years of checkout data that can be used to jump-start the recommender system's recommendations. Unfortunately, that data is difficult to format in a way that a recommender system can use. With some effort, it can be stored in a graph database and used to generate recommendations for users based on their preferences. Calculating recommendations before they are required and storing them for future use can speed up the amount of time it takes the recommender system to retrieve recommendations.
Introduction

Libraries, institutions that have existed for millennia, are beginning to see a decline in relevance. As society moves into the technological age, libraries seem to be taking their place as an institution of the past. In Emerging Trends in Library Organization: What Influences Change, Sul Lee explains the influence that new technologies can have. “Electronic technology is a powerful tool which has the potential to do things to libraries rather than for libraries (Lee, 1978, p.45).” The problem is that libraries are not integrating well into the digital age we now live in. Instead, they cling to the outdated methods that they used in the past. In Glut, Alex Wright describes the entrenched state of libraries today.

Today, if you walk into almost any library, you will find what is still, at its core, a nineteenth-century institution. The primary feature of most libraries is still a set of long shelves populated by industrially printed books, organized according to a proscribed hierarchical system of call numbers, maintained by specially trained workers laboring in a highly regimented organizational system. The old card catalog may have given way to computer terminals, but the underlying organizational and ontological structures of the modern library have hardly changed at all. Librarians still follow cataloging practices that originated in the 1850s (Wright, p.166).

Libraries seem to be molding new technologies to fit the library system, rather than molding the library system to fit with new technologies. Because of this, some may argue that libraries are doomed to become irrelevant, but others, such as Susan Martin, think libraries are poised to pave the way for the future of information technology. “Libraries and librarians are in an admirable position to take advantage of these increasingly sophisticated technologies (Martin, 1977, p.3).”
Challenges Currently Facing the Harold B. Lee Library at BYU

Brigham Young University’s Harold B. Lee Library is not immune to the technological challenges that face libraries today. At first glance, it would seem to be a haven of research for students. However, though millions of students pass through its corridors each year, fewer and fewer are coming to make use of the library's collections. According to the Harold B. Lee Library’s database of checkout history, half of its more than 4 million items have never been checked out even once. Additionally, the number of checkouts of library materials has been declining steadily for years, yet the library collection continues to grow.

These trends bring into question the continued effectiveness of the BYU Library. One possible solution to this problem is for the BYU Library to provide new technologies to help its patrons search for information. Rocha, Simas, Rechtsteiner, Giacomo, and Luce suggest that a collaborative recommender system in a digital library could do just that. “Today’s web technology allows us to consider new ways of working with (digital libraries). Rather than limiting the users to work in an isolated mode as a (sic) individual with generic capabilities, we can now enable users to work collaboratively and in a personalized manner when desired” (Rocha, 2005, p. 566).

The History of Collaborative Recommender Systems

In 1998, Amazon.com filed a patent for a new technology: the collaborative recommendation system. The collaborative recommendation system uses complex mathematics to calculate the best results to give users who are searching for something (Linden, Jacobi, Benson, 1998). It has proven to increase sales because it makes it easier for users to find what they are looking for. Amazon has enjoyed enough success from this system that many other
companies have also begun using collaborative recommendation systems in their sales efforts.

Statistics show that users are turning away from libraries for information retrieval, likely because it is now comparatively more difficult to find what you are looking for in a library. Consequently, a recommendation system like the one used by Amazon may be exactly what the BYU Library needs in order to get patrons connected to the valuable resources it holds in its collections.

Unfortunately, collaborative recommender systems are not trivial to create and maintain. In this paper, I examine the process and requirements in order to determine the feasibility of implementing a working collaborative recommender system in the Harold B. Lee Library at Brigham Young University. Unless otherwise stated, the feasibility of the processes has been verified by actual implementation.

Implementation

General Mechanics of a Recommender System

Recommender systems typically operate on three pieces of information: users (the people for whom the recommender system is providing recommendations), items (what the recommender system recommends to the users), and the links between users and items. The ultimate goal of a recommender system is to predict the preferences of users for items. That sounds simple enough, but there are many different kinds of recommender systems that all take a different approach to solving that problem.

User-based recommender systems create predictions of user preference by comparing users to each other. As users buy, view, check out, or otherwise interact with items, the system
keeps track of it. The system uses this information to compare two users' histories, thereby
determining how closely related two users' preferences are to each other. As an example, suppose
that two users of an online store both have the same 
purchasing history, except that one of them has
purchased one item that the other hasn't. Because
the rest of their purchases are identical, the system
determines that their preferences are highly related.

From this, the system predicts that the user that didn't purchase the item that the other user
purchased may be interested in purchasing that item. It turns out that these sorts of comparisons
are usually very accurate.

Item-based recommender systems take the opposite approach to predictions as user-based
recommender systems. Instead of comparing the preferences of users based on their interactions
with items, item-based recommender systems compare how closely related items are to each
other based on the users that interacted with them. Two items are determined to be related to
each other in some way if one user interacted with both of them. Beyond just being related,
weights, or quantifications of importance, can be assigned to specify how closely related the
items are. These weights can be calculated on such things as frequency and date proximity.
Frequency means that the more users who interact with both items, the higher the weight given
to the connection between the two items ought to be. For example, if one user purchases two
items, and then a different user purchases the same two items, they ought to be deemed more
closely related than if only the one user had purchased both of them. Date proximity means that
the more time that passed between the interaction that one user had with each item, the lower the
weight given to the connection between the two items ought to be.
For example, if a user purchases two items on the same day, they are
more likely to be related than if he purchased them ten years apart.
Date proximity is not always accurate. For example, sequential
books in a series are clearly related. However, rarely would someone
purchase or check out two books in the same series at the same time. Instead, they will purchase
or check out the first book, read it, then purchase or check out the second. This is a known
limitation of date proximity calculations.

Both user-based and item-based recommender systems are types of collaborative
recommender systems (they both use trends in user behavior to determine recommendations).
Another common type of recommender system, the content-based recommender system, is worth
mentioning because of its potential application in a library setting. Content-based recommender
systems attempt to characterize the items and the users in the system. Recommender systems that
use the collaborative filtering approach base recommendations purely on trends. In contrast, the
content-based approach tries to determine the actual content of an item in order to match it to a
user’s preferences. This does not work in many settings, such as items whose content is difficult
for a programatic algorithm to determine. In a library, however, call numbers and other
categorizations already attempt to give some minimal information about the content of a book or
another item in the library catalog. This means that an effective recommender system in a library
might be able to make use of at least some aspects of the content-based approach to
recommendations. This is confirmed by Huang, Chung, Ong, and Chen who proposed a hybrid
recommender system. “While most existing recommender systems rely either on a content-based

![Figure 2: Item-based correlations – Items 1 and 2 are related because the same user interacted with both of them.](image-url)
approach or a collaborative approach to make recommendations, there is potential to improve recommendation quality by using a combination of both approaches (a hybrid approach)” (Huang et al., 2002). They implemented this idea in a digital library setting and found that a hybrid approach does indeed give better results.

**BYU Library Recommender System**

One of the biggest challenges with starting a recommender system is getting enough data to make it work. In a 2009 paper on digital library recommender systems, Liao, Hsu, Cheng, and Chen explain the problem. “The collaborative filtering recommending method is harder to be applied to the library systems comparing with content-based method. The reason is that acquiring ratings of items can be a problem in practice” (Liao et al., 2009). When there is not enough data present to make a good recommendation, a recommender system will sometimes give strange results. Even with an established recommender system, like the one at Amazon.com, this still happens when gathering a recommendation for a rare item. But when a recommender system doesn't have much data at its fingertips, it will generate strange recommendations for all items. Luckily, the Harold B. Lee Library has been using a digital checkout system since 1999 and has all of the checkout information from then until now in a database. This means that a recommender system for the library could begin with almost 13 years of checkout data to use in making recommendations. This is a huge benefit, especially since sparsity of data is a major concern for library recommendation systems.

Despite the wealth of data available, there are several hurdles to face in order to build an
effective recommender system at the Harold B. Lee Library at Brigham Young University: the data requires significant modification in order to be used for a recommender; once the data is in the correct form, it is massive, requiring immense space and processing power; the data must be constantly updated as patrons of the library make checkouts; and personalizing recommendations requires a computationally-intensive algorithm. I will discuss the requirements of each of these challenges, finding that they are daunting, though not insurmountable.

**Formatting the Data**

Any time that a change happens to the Harold B. Lee Library's digital system, it is recorded in a history file. The history files are gigantic digital files full of text where each line in the file represents something that happened in the system. Checkouts take a particular form that can be filtered out of the history files using regular expressions. A discussion on regular expressions is beyond the scope of this paper (see Watt, 2005). For each checkout, the date, the barcode of the item checked out, and the user who checked out the item are recorded.

In order to convert a checkout to a form that can be used by a recommender system, all of the lines of text that describe a checkout must be isolated from the history files. Next, a list of all the users who have checked out an item must be created. This is done by scanning through all of the checkouts and looking at the unique identifier (ID) of the user who performed the checkout. If the ID is already in the list, then it isn't added again. If it isn't in the list, then it is added. The same procedure is followed to produce a list of barcodes.

At the Harold B. Lee Library, each item has a barcode attached to it. This barcode is unique to that item, meaning that no two items should ever have the same barcode. However, if the library has two copies of the same item, they are each given a different barcode. For the
purpose of a recommender system, it doesn't make much sense to give recommendations that are specific to the copy of the item, rather than the item itself. Additionally, the library may have multiple editions of the same item, which ought to be treated as the same item in a recommender system.

Complicating matters further, one item may have multiple barcodes through the course of its time at the library. The reason for this is that barcode stickers wear down over time and have to be replaced. Rather than printing up a new sticker with the same barcode on it to replace the worn out sticker, the library uses preprinted barcodes that have a longer shelf life. This means that any time an item has a barcode that wears down, it is replaced by a new barcode. Checkouts in the history files don't get updated with the new barcodes. Instead, there is a different line of text that is placed in the history files, indicating that one barcode was replaced with a different barcode. This means that before the data from the history files can be useful, all of the old barcodes have to be replaced with their current version.

Once all barcodes have been replaced, the barcodes must be combined such that different copies and editions of the same book are considered to be the same item. The digital library catalog that the Harold B. Lee Library uses has built-in ways of grouping items. At the lowest grouping level, all identical copies of the same item are grouped. At the highest grouping, items that have the same content are grouped. Each lower level grouping has an ID associated with it, called the catalog key. Each of these keys represents a catalog group. Likewise, the higher grouping has an ID, called a frbr ID, which creates a frbr group. Since the goal is to provide recommendations based on similar content, not on specific editions, all of the barcodes in the checkout data must be replaced with frbr IDs.
Massive Amounts of Data

Once all of the data has been filtered, replaced, and formatted as previously stated, it must be stored in a database, so that the recommender system can access all of it. I chose to use a graph database. Graph databases store nodes (bits of information) and relationships (connections between nodes). For a library recommender system, a graph database should have user nodes and item nodes, with relationships between them being checkouts. In other words, if a relationship exists between a user and an item, then that means that the user checked out the item from the library. This is the basic setup for the graph database.

For the Harold B. Lee Library, about 7.5 million checkouts of 1.4 million items have been performed by 270,000 users since 1999. This means that the database will hold around 1.7 million nodes, with around 7.5 million relationships between them. This requires almost five Gigabytes of storage space. Luckily, technology has advanced to a point where even the cell phones in our pockets are capable of storing that much data. The more pressing concern is computation time.

Because people are no longer accustomed to waiting a long time for a web page to load, a recommender system needs to be able to return recommendations quickly. This means that calculating a recommendation for a user needs to happen in less than a few seconds.

Computation time can be decreased by indexing the results of the computations. Indexing the

---

**Figure 4: Heirarchy of item groupings** – Recommendations operate on the frbr group level.
results means that, instead of waiting for a user to ask for a recommendation, the recommendations are computed beforehand and stored somewhere that they can be accessed later. This requires much more storage space and requires that the recommender system constantly update all of the recommendation computations.

The indexing approach also sacrifices customizability. If the recommendations have all been calculated beforehand, then the many different factors that go into calculating a recommendation cannot be tweaked from case to case. Instead, an ideal formula for the calculation would be used to create all of the indexed recommendations.

In order for a recommender system to be viable for use, it will need to index the recommendations. Otherwise, the computations will take too long to compute. Unfortunately, I have not yet indexed the library recommendations during my research. I have, however, created the algorithm to run through all of the past years of checkout data and index recommendations for them. This will only need to be performed once, but it will take over a month to run. Once the data is indexed, recommendations will be returned in a fraction of a second, allowing the recommender system to run effectively even on an older personal computer.

**Updating the Database**

In order to keep recommendations up-to-date, a recommender system for the Harold B. Lee Library will need to be able to augment its database with checkouts as they happen, or shortly thereafter. This will probably best be performed as a daily operation that happens after the library closes at night. The updater will go through the entire process of scanning the history files to find the checkouts that happened that day, replacing any barcodes that need to be replaced for that day, grouping them into frbr IDs, and then adding the information to the
database. If recommendations are being indexed, then the updater will also need to update the recommendations with the new information.

The update process will be computationally intensive. While the Harold B. Lee Library has access to many computers and servers, the computation time required to perform this daily update will need to be factored in. Other services performed by the library's servers might be compromised by a sudden drain on resources. Luckily, according to the programmers in the Library Information Systems department of the Harold B. Lee Library, the library has plenty of powerful servers available to handle these tasks.

**Personalized Recommendations**

Suppose that a user searches for an item on the Harold B. Lee Library's website. The recommender system's goal is to provide that user with a list of items that may also interest the user. As described previously, recommendations are better equipped to predict a user's preferences if they consider similarities between users. Factoring the similarities between users into the computation requires a time-consuming algorithm.

First, the recommender system finds the item that the user is currently viewing from the graph database. Next, it finds every user who has ever checked out that item. For each of the users who checked out that item, the recommender system determines how closely related their preferences are to the user who is currently viewing the item.

This computation can be done in a number of different ways. One of the most common ways of computing this is with cosine similarity. The checkouts of the two users are constructed as a vector, where each dimension of the vector represents an item in the library catalog. The
Harold B. Lee Library Recommender System

The magnitude of the vector in each dimension is determined by the number of checkouts that the user has made of the item. The dot product of the two vectors is used to compute the cosine of the angle between them. The cosine of the angle between the vector of the first and the second users gives an indication of how closely related the two users are. The result will range from zero to one. If the two users have the exact same checkout history, then they have the exact same vector and the cosine of the angle between the two vectors is one. If they don't share any checkouts of any items, then the cosine of the angle between the two vectors is zero.

There are two alterations that can increase the speed of this computation significantly. First, if we don't care how many times a user has checked out an item, but only care whether or not the user checked the item out at all, then the vector becomes a boolean vector, where every value is either one or zero. This makes the calculation of cosine similarity trivial; the dot product is just the total number of shared items that the two users have checked out. The second way to increase speed is by not converting the dot product to a cosine. This step normally involves one division and one multiplication. This actually solves a limitation of cosine similarity, which is that users who check out tons of items are deemed less similar to other users than users who only check out a few items, even if they share the same number of items. Removing the cosine computation means that the computation just becomes a comparison of checkouts to determine the total number of shared checkouts between the two users.

Once the similarity has been calculated between the current user and all of the users who have checked out the item that the current user is viewing, all of the other items that the users who checked out the item checked out are considered for recommendation. As previously explained, the date proximity is factored in as well as the similarity between the users. For
example, if some user checked out the current item (A), along with another item (B) three days later, the items would be deemed highly related. Then, supposing that the user is also determined to be very similar to the current user, then that recommendation would be high. In other words, the recommendation from that user of item B, based on item A is very high for the current user. This computation is calculated by multiplying the result of the date proximity between the user’s checkouts of items A and B with the similarity computation between the user and the current user. The overall computation of how highly related item B is to item A for the current user is the summation of the computations for each of the users who checked out both items A and B.

The similarity between users can be computed beforehand and indexed to speed later computation using the same procedure as the indexing of the similarity between items. This indexing also requires a large amount of storage space and more than a month to compute for the previous checkout data. Again, this computation only needs to be performed once. The only challenge comes in creating the recommendations later. Without a personalization computation, and with an indexed computation of recommendations, gathering a recommendation for a book requires almost no additional computation. However, including a personalized computation, even if it has been indexed, still requires some computation to generate the final recommendation. Luckily, this computation will not be nearly as time consuming as the indexed computations, meaning that adding personalization to the recommendations is feasible if the similarities between users and items have both been computed beforehand and indexed for quick access.
Recommendations

Implementing a recommender system at the Harold B. Lee Library at Brigham Young University is feasible and needed. Through the course of this research, I have verified the feasibility of each of the required elements of a recommender system for the library. The only exception is that I haven't implemented the indexing of item and user relationships. This feature is key to whether or not a recommender system will be feasible; without it, the recommendations will take too long to compute. Luckily, I have implemented enough of it to know that it will take a little over a month to index all of the existing data. This means that implementing a recommender system at the library at BYU is feasible.

Future Work

Beyond focusing on the feasibility of implementing a collaborative recommender system at the Brigham Young University Harold B. Lee Library, future work should include examining the various kinds of recommender systems. Varying algorithms for computing the similarity between two users, the correlation of items as the time between checkouts increases, and the final weightings alter the ordering of the recommendations (Su & Khoshgoftaar, 2006). Case studies and usability testing can be used to determine which combinations of algorithms work best at the Harold B. Lee Library.
Appendix A

A visualization of the relations between items from the checkout data at the Harold B. Lee Library shows visual research pathways. The following visualization is a single fully-connected graph of relations from the whole graph.
Appendix B

The following is pseudo code for the indexing process.

User relations:

```
Method for indexing all of the user relations.
for each item:
    for each user:
        for each user:
            if same user:
                continue
            if graph contains relation between these users
                continue
            compute weight by cosine similarity
            add relation to graph

-This process is slow. Big O approximately n^4
```

Item relations:

```
Method for indexing all of the item relations.
for each user:
    for each checkout:
        for each checkout:
            if same checkout:
                continue
            if graph contains relation with these specifications:
                if relation contains this user:
                    continue
                compute weight by date proximity
                increment relation weight by this computation
            else:
                compute weight by date proximity
                add relation to graph

-This process is slow. Big O approximately n^5
```
References


