Midterm Project Report
Social Computing

Kevin Swelsen & John Servaes

Information 290

November 1, 2011

Abstract

This report gives an overview of the project we've been working on since the beginning of the semester. The aim of the project is to build a recommender system for a social networking website called Goodreads.com. Furthermore, the aim is to build a recommender system with and without a social component, and being able to compare both in terms of accuracy and efficiency.

Several steps have already been taken. We started with a proof of concept for the project that was aimed to answer two separate questions. First, are we able to implement an algorithm that can give recommendations to a user? And second, does Goodreads.com provide us with the data we need? For the proof of concept we build an algorithm called Slope One that uses collaborative filtering to give recommendations to users. A collaborative filtering algorithm recommends based on the interests of other people and the user's own interest. The API of Goodreads.com was a bit troublesome. Even though we were able to get from the API, users were not very willing in sharing their profile with us. Also, the API was rather slow, meaning that it would take a long time to actually get all the data off of the website.

For the rest of the semester, we want to implement a second algorithm, this time with a social component. This would allow us to compare both of the algorithms. Also, we have to come up with a solution for the problems with the data. One possible and very likely solution would be to use fake data, and validate it.

Index

Introduction

With so many choices these days, we have a hard time making decisions. Iyengar and Lepper(2000) have demonstrated that indeed, when confronted with too many choices, our satisfaction decreases. As the number of options increase, people become less happy with their choice as they feel that they probably did not pick the best option.

These days, many websites help users make choices by using recommender systems to provide a smaller set of options that are likely to be of interest to that user. Recommender systems have been around for a while. E-commerce companies such as Amazon.com recommends products to their customers based on what others have purchased. Recommender systems are being applied to more than just online experiences; these days they are also implemented in set-top boxes giving users recommendations on what TV-program to watch. By implementing recommender systems, websites can hope to increase choice satisfaction.

The rise of social networking sites like for example Facebook give rise to a lot of new opportunities. Not only do we know with these kind of websites which movies a person likes or which music that person likes, but also we can learn a lot from the relations between that person and other people. A good example of this is a website called Goodreads. Using Goodreads, one can create his or her own profile, and list books that they have read. They can give a rating to books, and they can write comments about that a certain book. But the website also has a large social factor. A user can become friends with other users. This is what we like to exploit in our project. Our project is to build a recommender system for Goodreads. Using the data available through the Goodreads API, we would like to build a recommender system and running an experiment to evaluate whether the adding of the social aspect into the recommender system actually improves the recommendations.

The remainder of this report will be organized as follows. First, we will try to give an overview of research that has been done on recommender systems. Then we will elaborate on our progress and some of the problems we faced already. In the third chapter an overview is given of work that still needs to be done.

1. Literature Review

There are basically two main types of recommender systems. First, Automated Collaborative Filtering (ACF) systems predict a user's affinity for items or information by connecting that person's recorded interests with the recorded interests of a community of people and sharing ratings between like-minded persons (Herlocker &Konstan & Riedl, 2000). This means that when a lot of my friends like a certain movie or book, there is a large probability I like it as well. Also, if I give five movies a high rating, other users who also rated these movies high as well then it is likely I also like other movies which received a high rating from those users.

Two kinds of collaborative filtering algorithms can be distinguished. Memory-based algorithms compute an index for similarity between pairs of users, through a weighted average (). However, memory-based algorithms have some problems (Sarwar & Karypis & Konstan & Riedl, 2001). Recommender systems are often used with large datasets. Users may have only rated one or two percent of all items in that dataset. That amount of data sparsity is a serious problem. Another serious problem with these kinds of algorithms is cold start. When a user has not yet rated anything, it is very difficult to recommend items to such a user. There are two kinds of memory-based algorithms, depending on whether the algorithm focuses on finding similar users (user-based) or similar items (item-based).

In research done by Lemire and Maclachlan (2005) they propose an algorithm called Slope One. Figure 1 below demonstrates the basis of their idea. In this situation user A rated both Item I and Item J whereas User B just rated Item I. Now, Lemire and MacLachlan state that if user A rated Item J half a point higher than Item I, the same is likely for user B. So, User B is expected to give Item J 2,5 points. The algorithm they build and test in the paper is based on this idea, and according to Lemire and MacLachlan it not only gives very good recommendations compared with other algorithms, but is also very easy to implement and efficient. However, like stated earlier, this kind of algorithm can have potential problems with scalability.

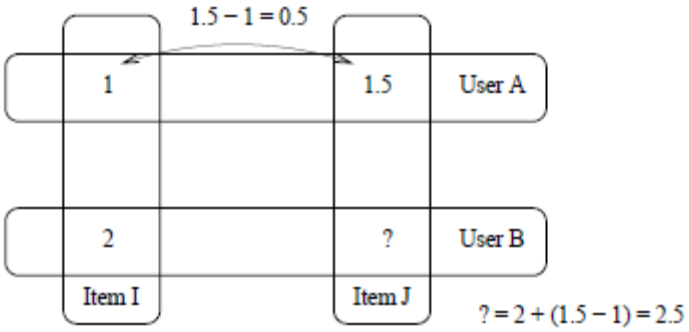

Figure 1: Basis of Slope One algorithm

Model-based algorithms provide recommendations based on a model of user ratings (Herlocker, Konstan & Riedl, 2004). Most of these kind of models are based on techniques like neural networks or bayes methods. One can imagine however that these kind of techniques are usually very difficult and do not perform well when having large datasets.

The other type of recommender system is a so-called content-based system. A content-based recommender system recommends products according to how a user rated associated features of other products (Burke, 2002; Degemmis et al., 2004, as cited by Ochi et al., 2009). For example, if a user has rated Bermuda highly and New York City poorly on a travel destination website, a content-based recommender system could use characteristics of these places such as climate and population density to recommend the Bahamas and not Chicago.

Content-based algorithms have certain disadvantages (Cacheda & Carneiro & Fernandez & Formoso, 2011). First, Items need to be machine-analyzed. This kind of analysis is computational very heavy. Another problem with content-based is that it cannot evaluate quality of a certain item. Obviously, quality is a very subjective feature, and this is very difficult for a machine to analyze. Last, content-based algorithms are not very good in finding items a user is not expected to find.

The rise of social networks like for example Facebook gives new opportunities for recommendation algorithms. Ma, Yang, Lyu and King (2008) therefore introduced a term called social recommendation. The framework they introduce in this paper takes into account not only users and items they rated, but also how these users know each other and how they trust each other. Ma et al. (2008) state that knowing a user's social network can make recommendations more accurate and personal. Their results show that the framework outperforms all of the other collaborative algorithms they have tried and tested. Also, using complexity analysis, they have shown that the algorithm is scalable to even very large datasets. As for social networks, in a lot of these you can also state which users you distrust. Ma et al. (2008) admit that including this kind of information into the algorithm could enhance the algorithm even further and could improve the accuracy of their framework even further.

Besides improving these kinds of algorithms, there is also quite a bit of research done on improving not just the algorithm, but the entire user experience. Knijnenburg, Willemsen and Kobsa (2011) argue that subjective system aspects (SSA), in addition to objective system aspects (OSA), of a recommender system may influence user experience as summarized in figure 2. The inclusion of OSA, such as for example explanations on how the system calculated certain recommendations, influence how people perceive the system (SSA) and therefore their overall experience and subsequent behaviors. In addition, personal characteristics, such as their propensity to trust systems or their persistence in finding the best option, may influence overall user experience as well.

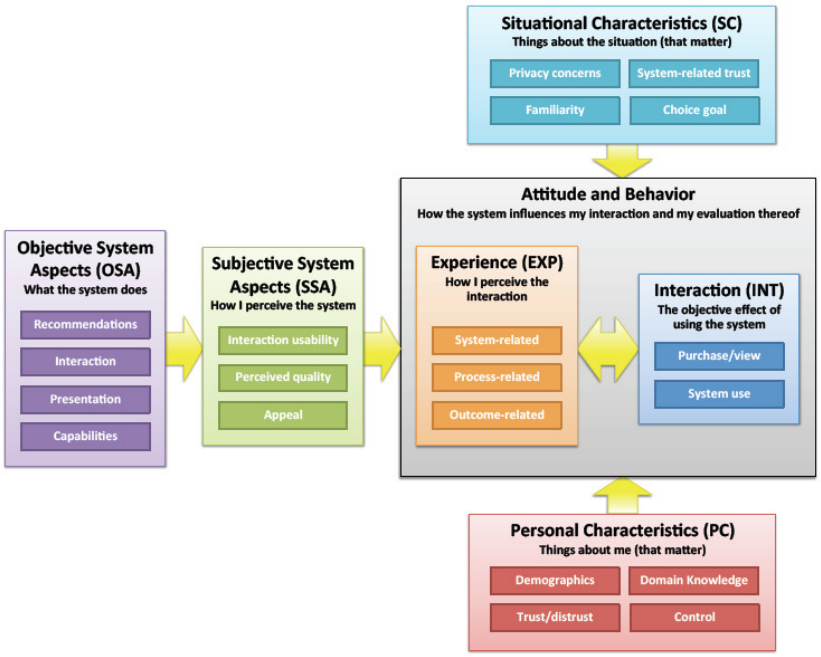

Figure 2. Recommender System Framework (Knijnenburg et al., 2011)

2. Progress & Problems Faced

The starting point of our research began with exploring the available theory that exist on the subject. The idea we had in the beginning was that it might be wise to start with an easy example, implement it fully, and then go further from there.

Our initial idea was to use Facebook and movies as data for our algorithms. This turned out to be difficult. The Facebook API changed a couple of weeks after the beginning of the semester, and the new API made it much more difficult to reach the data we needed to have. Therefore, we started to explore other options. Eventually, a book catalog website called Goodreads.com turned out to do exactly what we wanted. Users have a social network with friends, and they can rate books. Also, it turns out that the users of Goodreads.com are very active and use the website a lot.

For this project we choose to use an ACF algorithm for making recommendations. After doing the literature review, we realized that we needed to use a collaborative filtering algorithm to make full use of the possibilities of a social network. Like Herlocker, Konstan and Riedl (2000) pointed out, a collaborative filtering algorithm uses a person's interests and connects that with that of the rest of the community and like-minded persons. This is exactly what is available through a social networking site like Goodreads.com. Through this website we know the interests of every user (the books that user rated highly) and we know about like-minded persons (the friends of that particular user) in that community. That made a lot of sense for us.

Before we started this project we had two main questions which needed to be answered to give an indication if this project can be successful. Are we able to implement one or even multiple algorithms to give the user an accurate recommendation and are we able to get enough data to verify, test and compare our designed algorithms?

We decided to create a proof of concept to answer each question separately. The goal of the proof of concept was not to see whether we could create an optimized algorithm which gives highly accurate recommendations but to proof we could implement an algorithm and get meaningful output data. Therefore, we decided to implement the Slope One algorithm in Python and use this as a baseline for a more complicated algorithm. So far we have been able to successfully implement the Slope One algorithm as described in an earlier chapter and use a data file to feed simulation data in to the algorithm. A more detailed overview of the implementation of the Slope One algorithm can be found in Appendix I. A lot of time went also into finding a suitable next algorithm to work on. A lot of algorithms exist, and all of them involve a lot of math. It was difficult to really understand and comprehend those. Finally, we've chosen to implement the algorithm by Ma, Yang, Lyu and King (2008). This algorithm not only has a very clear social component, but is also very well scalable to large datasets and is therefore a good choice, we think, to use in this project.

The second question is all about acquiring enough relevant data to test and compare our algorithms. This is important to be able to test and compare different algorithms. We chose the website Goodreads.com to be our source for data because it has a vast majority of users around the world who are interconnected via their personal profiles and the reviews they have written about the books they

have read. A second reason why we chose Goodreads.com as our primary data source is the availability of an API which makes it easy to crawl the website for data. Our main concern is not so much the technical aspect but whether enough users give us approval to use the data on their personal profile so our dataset can become large enough to test the algorithms.

We have created a referral page where Goodreads users can give us permission to scrape their profile information such as book ratings, reviews and connections with other users. When a user gives his/her permission the user profile is accessed via the API and the retrieved information is stored in a MySQL database according to the Goodreads terms of use. As we would have foreseen the implementation of the communication layer around the API has not caused many troubles. Although, the speed with which the data acquired is not as fast as we would have liked. People tend to be wary of giving their permission which might be due to privacy norms or because the Goodreads website already includes a recommendation system. We want to simulate data to address this problem and to make sure we can test and compare our algorithms. Although, we are still hoping to gather enough real life data as a viable back-up plan. A more detailed overview on the implementation of the API is available in Appendix II.

The past weeks we have addressed the two main concerns described in the above paragraph and spend the rest of the time doing literature research. One of the problems we ran into was the acquiring of the right data. We have to schedule time in the next weeks to really investigate if and how we can simulate that data. By acknowledging the possibility of might having to use simulated data instead of real data, we think it is only fair to allocate more time to do the extra research on this topic.

A problem which might have had the largest impact on our process is the fact we initially had third team member. Our original team consisted of three persons instead of two. The third team member decided to drop the course a couple of weeks ago due to the high workload of other courses. This made us rethink about the scope of the project, and what became feasible or not.

3. Next steps

Algorithm

In the coming weeks most of the time is allocated for developing and optimizing our algorithms. As we now have the Slope One algorithm implemented we treat this algorithm as the baseline and try to optimize the accuracy of the recommendations.

The next step is to implement an algorithm which uses a social component. We choose to implement an algorithm proposed by Ma, Yang, Lyu and King (2008) called Social Recommendation and our hypothesis is that the latter will outperform the Slope One algorithm. The first steps have been taken to implement this social algorithm and in the coming weeks the algorithm must be implemented and optimized further.

Validating the algorithms

Our first choice would be to use real data to validate the algorithms. However, like pointed out earlier it might be the case that we would not acquire enough real data to be able to validate the algorithms properly. If this is the case we switch to simulated data or use both datasets.

To test an algorithm we divide the complete dataset into two pieces and randomly assign fifty percent of the data to each of the subset. One of the subsets will be used to train the algorithm where the other subset is used to test the algorithm. The goal is to optimize the algorithm and achieve a high generalization performance. The optimization is a trade-off between the optimization of an individual subset and the subsets combined. By varying the capacity we can estimate the test error and predict which model performs best on the test set.

Schedule

	Week												
	37 Sep 12-18	38 Sep 19 - 25	39 Sep 26 - Oct 2	40 Oct 3 - 9	41 Oct 10 - 16	42 Oct 17 - 23	43 Oct 24 - 30	44 Oct 31 - Nov 6	45 Nov 7 - 13	46 Nov 14 - 20	47 Nov 21 - 27	48 28-Dec-04	49 Dec 5 - 11
Project proposal													
Feedback project proposal													
Project presentation													
Proof of concept: Goodreads data + API													
Proof of concept: Slope One algorithm													
Mid-term project report													
Coding: Social Algorithm													
Coding: GUI													
Final project presentation													
Final project report													?

Summary

In this report we have tried to give an overview of the project we are working on since the beginning of the semester. The aim of this project is to build a recommender system for a website called Goodreads.com. Goodreads.com is a social networking site build around books. Users can create a profile, add friends and give ratings on books. The aim of this project is to experiment with a recommender system that uses a social component and compare this with a recommender system that does not have a social component.

Several kinds of recommender systems exist. Automated Collaborative Filtering (ACF) algorithms use a community of people and their interests to give a recommendation to a particular user. This can be done in two different ways, either by creating a model of user's rating or by looking at similarity between users (user-based). Content-based Algorithms look at recommendations from a different prospective. Content-based algorithms look at features of items. For example, if a user rated the movie The Exorcist really high, and the Titanic really low, the algorithm would, based on the features of both movies, recommend Dracula and not Grease.

Our initial plan for this project was to create a proof of concept. This proof of concept had two answer two questions. First, are we able to implement one or multiple recommender algorithms, and second, are we able to get the right data from Goodreads.com to use in our recommender algorithm. For implementing a first algorithm, we chose an algorithm called Slope One. This is an ACF algorithm, but without a social component. The advantage however of this algorithm is, is that it is relatively easy to implement and can recommend well even compared to more difficult algorithms. We chose an ACF algorithm because these kinds of algorithms actually make use of the community and the data that is available through social networks. By doing experiments with the Goodreads.com API we found that all of the data we needed is available through the API. However, the speed with which the data can be gathered is low, and people are not very willing to give us permission to actually use their data.

For the remainder of the semester we are planning implement at least one algorithm which uses the online social relations of users to recommend a book. This algorithm will be compared with the Slope One algorithm and validated by simulating a social network.

Appendix I: Goodreads API

The Goodreads website provides an API helping developers use the rich source of information which is available on the website of Goodreads. The API uses the OAUTH authentication method to ensure that the user has given permission to use their profile data.

When a user grants permission a CURL request is sent to the Goodreads website requesting the available information of the user.

Goodreads sends back an XML file containing the requested information which is parsed and stored into a MySQL database.

An example of (part of) an XML file that contains a review of a book:

```
<review>
<id>22</id>
<user>
<id>1</id>
<name>
<![CDATA[ Otis ]]>
</name>
<location>
<![CDATA[ Santa Monica, CA ]]>
</location>
<link>
<![CDATA[ http://www.goodreads.com/user/show/1-otis-chandler ]]>
</link>
<image_url>
<![CDATA[
http://photo.goodreads.com/users/1189644957p3/1.jpg
]]>
</image_url>
<small_image_url>
<![CDATA[
http://photo.goodreads.com/users/1189644957p2/1.jpg
]]>
</small_image_url>
</user>
<book>
<id type="integer">2</id>
<isbn>0439358078</isbn>
<isbn13>9780439358071</isbn13>
<text_reviews_count type="integer">8593</text_reviews_count>
<title>
<![CDATA[
Harry Potter and the Order of the Phoenix (Harry Potter, #5)
]]>
</title>
<image_url>http://photo.goodreads.com/books/1293473437m/2.jpg</image_url>
<small_image_url>http://photo.goodreads.com/books/1293473437s/2.jpg</small_image_u
rl>
<link>
http://www.goodreads.com/book/show/2.Harry_Potter_and_the_Order_of_the_Phoenix
</link>
<num_pages>870</num_pages>
<average_rating>4.32</average_rating>
<ratings_count>463960</ratings_count>
```

Appendix II: Slope one algorithm

In order to build an algorithm, the first thing that is needed is data. To get some data to feed into the algorithm, a script was build to generate such data. At the moment this script is very simple. It generates 500 users, who each rated 10 items. These 10 items come out of a pool of also 250 items. Even though we haven't done any tests on the validity of this test-data, it at least makes a bit of sense. Users will probably rate only a very small subset of the possible items.

So, in our data a user would look like the following:

```
498:dict(
id12=0.2,
id216=0.2,
id224=0.6,
id133=0.6,
id152=0.4,
id211=0.7,
id171=0.4,
id195=0.5,
id76=0.7,
id145=0.8)
```

This small piece of our dataset basically says that user with id 498 rated those 10 items on a scale from 0 to 1.

Then, in our actual algorithm we define the user we want to give the recommendation to, with the items that this user rated. Based on this and based on the ratings of the other users, the algorithm calculates the actual recommendation. At this moment the output (the actual recommendations) are still outputted by the command line. A short piece of the output would look like the following:

```
{'id49': 0.10000000000000003, 'id2': 0.69999999999999996, 'id44': 0.15000000000
00002, 'id43': 0.0, 'id42': 0.29999999999999999, 'id40': -0.19999999999999996,
```

Eventually, this would be shown in a web interface, and only the first few items will be shown.

References

Cacheda, F., & Carneiro, V., & Fernández, D., & Formoso, V. (2011). Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web*, 5(1), 2.

Herlocker, J.L., & Konstan, J.A., & Riedl, J. (2000). Explaining collaborative filtering recommendations. *Proc.CSCW2000*, 241–250.

Iyengar, S. S., &Lepper, M. R. (2000). When Choice is Demotivating: Can One Desire Too Much of a Good Thing? *Journal of Personality and Social Psychology*, 79, 995-1006.

Knijnenburg, B.P., & Willemsen, M.C., & Kobsa, A. (2011). A Pragmatic Procedure to Support the User-Centric Evaluation of Recommender Systems. *Short paper accepted to the ACM Conference on Recommender Systems (RecSys)*.

Lemire, D., & Maclachlan, A. (2005), Slope One Predictors for Online Rating-Based Collaborative Filtering. *Proc. SIAM Data Mining (SDM '05).*

Ma, H., & Yang, H., & Lyu, M.R., & King, I. (2008). SoRec: Social recommendation using probabilistic matrix factorization. *Proc. of CIKM '08*, 931–940.

Ochi, P., Rao, S., Takayama, L., &Nass, C. (2009). Predictors of user perceptions of web recommender systems: How the basis for generating experience and search product recommendations affects user responses. *International Journal Human-Computer Studies,* 68, 472–482.

Sarwar, B., &, Karypis, G., & Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. *WWW '01: Proceedings of the 10th international conference on World Wide Web*, 285–295.