Amir H. Hajizamani
St. John's College
ahh29

Computer Science Tripos Part II Individual Project Proposal

Recommender Systems for Social Networks

20 October 2010

Project Originator: Amir H. Hajizamani

Project Supervisor: Cecilia Mascolo

Director of Studies: Robert Mullins

Overseers: Ann Copestake and Robert Harle

Resources Required: See attached Project Resource Form

1 Introduction and Description of the Work

In this project I will explore and implement techniques for making recommendations to members of online social networks. The motivation behind this idea is that the value of data about an individual person or some digital content (text, audio, video, etc.) is increased by knowledge of its relation to other such entities. This is true for the user, whose experience is enriched, and the service provider, whose cultural and financial success will depend on behaviour-based analysis and the growth and interactions of its users.

Motivation

With the growing number of available social network services that individuals can choose to invest time and effort into, it is imperative that there is an incentive to do so. We take this investment by the user to consist of: providing personal information, creating and sharing content, making social connections with other users, and interacting with other users and content in the myriad ways that are possible and observed in real world implementation such as Facebook, YouTube, LinkedIn, Twitter and Foursquare, to name a few. One way of providing this incentive is by making it easier for users to make *discoveries* – to find other individuals of interest (real-world friends, people with similar characteristics) or content. That is, make recommendations to users and keep them interested in exploring what the social network has to offer. Recommender systems aim to solve the problem of *information overload* in areas where the users of a service struggle to make good choices on what content to consume or what social activities to take part in, simply because the options are overwhelming.

If we treat the users of a social network as a community, it is clear that the community will need the investment mentioned above from its users so that it reaches critical mass – the point at which it is valuable for others to join or contribute to the network because the majority of their friends have, or so much of the content they consume is accessible via the service. This will continually increase the richness and quality of the service the community receives (assuming the service provider keeps up and adapts to changes!) and increase the lifetime and utility of the service for its users, too. Of course, here we are concerned with social networks that aim to reach the mainstream and are not inherently limited to a select number of users. For instance, a social network for the community of world experts on network congestion analysis is bound to hit its maximum membership limit very quickly and not be much more useful to its users than email or more traditional communication methods.

From a commercial perspective, which is arguably the more important one for the survival of a service, having users remain active on the network and incentivise others to do so, too, is paramount. It is well-understood that social network service providers are not

expected to be profitable[1] until they reach an "audience of scale" to enable the adoption of a sensible monetisation model. For instance, Facebook only became profitable in 2009[2] after reaching 300 million users. This means that the investors who enable such ventures at the start need to be convinced that the service has the potential to reach popularity and therefore provide a return on investment. Of the many strategies that can be used to maintain the growth of a social network, implementing a good recommender system is standard practice. It should be noted that recommendations are difficult, if not infeasible, to make at the very beginning of a network's lifetime due to the *cold start* problem, that is, when there is insufficient data to base recommendations on. Therefore, the focus of this project is on stages of a network's lifetime after this issue has been overcome.

Below I describe my chosen dataset and the reasons for this choice over other available datasets.

The *Mixcloud* Dataset

I will work on data from Mixcloud.com whom I worked for over the summer of 2009. They are a steadily-growing and successful content distribution website, aiming to be the "YouTube of Radio". They provide a platform for user-generated audio content, ranging from podcasts to DJ mixes, to be easily accessed by anyone. They have been operating for over a year and have in the order of 100,000 users and a similar number of so-called "Cloudcasts" (uploaded audio content). They provide their users with social networking features such as Twitter-style following, Facebook-style activity updates, favouriting of Cloudcasts and commenting, and a record of listening history. The user-uploaded Cloudcasts are also heavily annotated with tags, categories, text descriptions, tracklists where appropriate, metadata about recent listeners and more.

This wealth of metadata about their content and users is available via their public API[3] and I shall be gathering the data I need from it. The fact that the API is public means that I am able to use the data in my project and the results will be publishable in the project dissertation. I have had further confirmation from the staff at Mixcloud Ltd. about this.

At the time of writing, Mixcloud implement some simple mechanisms for *item recommendations* and *personalised recommendations* – recommending Cloudcasts similar to the current one, and recommending Cloudcasts matching the user's personal listening history. I would like my recommender system to primarily aim to provide the functionality of *social recommendations* – suggesting other users whom the current user may be interested in following or listening to based on their shared social connections, activity similarities and other metrics. The problem of recommending content to users based on the listening pattern of their own social connections (as opposed to that of their entire community) can also

[1] `http://www.makeuseof.com/tag/how-do-social-networks-make-money-case-wondering/` – explains the risks behind the business of social networks

[2] `http://www.pcpro.co.uk/news/351646/facebook-eyes-profit-as-it-hits-300-million-members`

[3] `http://api.mixcloud.com`

be called *social recommendation* and would likely use similar underlying data and metrics. I should point out that the social aspect of the Mixcloud community is more similar to that of Twitter, where people follow users who produce good content as well as real life friends, than that of Facebook where social links are more often initiated physically.

Following the motivations described earlier in the last subsection for recommender systems, I understand that Mixcloud have prioritised the implementation of Cloudcast recommendations on their website because their main focus has been on building up the library of content on their network and promoting themselves as content distributors. However, the social networking layer on top of their distribution layer is naturally their next focus, whilst they improve the recommendation algorithms they currently use, too. They can leverage the high standard of their content metadata and existing social connections to make social recommendations. So essentially, the quality and completeness of the dataset, despite its organic nature, and the utility of a social recommender system for Mixcloud and their community are good reasons for picking it. I also compare the Mixcloud dataset with other ones traditionally used to explore recommender system techniques below.

A working system would likely draw on concepts from all three types of recommendation mentioned so far to account for inherent peculiarities in the dataset. For example, two users may have common taste in music but not be well-connected in the social graph or vice-versa. An effective system would take advantage of the rich social graph and indirect relationships between users and content to overcome these issues and enable further linkage in the dataset, hopefully with the effect of aiding the community and commercial objectives outlined so far.

Comparison with other Available Datasets

The most widely used datasets in discussions on recommender systems are the MovieLens, EachMovie, Book-Crossing and Jester Jokes datasets[4] which contains items and user ratings on their respective titular item types. Though the quality of these datasets is proven, they have limitations for my intended project goals. In particular, there is no social connectivity data between users in these datasets and the links between entities are limited to ratings by users on some items. With the Mixcloud dataset I am able to explore the social graph of the users. Furthermore, the Mixcloud dataset provides much richer metadata about its entities, as described above, than any of the traditional datasets which give only enough data to build item-item and item-user matrices corresponding the ratings. The Mixcloud dataset has scope for more interesting analysis and recommender system implementations.

The Mixcloud dataset also has the potential for analysis in the temporal dimension, which is the extension idea for this project and is described later in this document.

[4] `http://www.grouplens.org/node/12` – all of these datasets are available via this link

Wider Context

The Web is rife with successful recommender systems, with companies such as *Amazon, Netflix*[5] and *Pandora* (radio service backed by *The Music Genome Project*[6]) investing a lot of time and money in improving their algorithms and data quality to better recommend items of interest to their users. On the other hand, more directly community-driven websites such as *del.icio.us* and *flickr* have harnessed the collective intelligence of their users and their tagging of their content with keywords. Collaborative tagging leads to an organic categorisation and annotation of data which has been termed *folksonomy*[7]. In a sufficiently large and active community, the folksonomy that can be extracted becomes a good source of input for the algorithms in a recommender system.

With the accelerating growth of social networks such as Twitter and Facebook, the value of social recommendations which aim to increase the connections between individuals is more important that it used to be: the items of interests are now other users not products on sale (though from an advertising revenue perspective the difference is blurred). Recommender systems are nearing maturity and will soon be, if they are not already, commodity technologies as the "social connections layer" of virtual communities is completed[8]. This means that an analysis of such systems is essential to the understanding of social networking and the Web as it experiences a paradigm shift from a *search* platform (e.g. Google) to a *discovery* platform.

2 Starting Point

I am familiar with the RESTful[9] API provided by Mixcloud which I have already begun obtaining data from. At the time of writing, I have collected data on over 20000 users by performing a breadth-first search on the social connections of a few hand-picked seed users. This is a a significant proportion of the Mixcloud user-space and I am confident that I will have all the data I need available to me in time.

What I have learned from some of the Computer Science Tripos courses such as Artificial Intelligence I, Digital Communications I and Databases will likely help me along the way.

The type of algorithms I will be using in this project will be new to me and my only exposure to them is in reading the relevant literature.

[5] In 2006 Netflix challenged programmers to improve their movie recommendation engine by 10%.

[6] `http://www.pandora.com` – The MGP uses the"genealogy" of music to link songs and artists

[7] `http://www.vanderwal.net/folksonomy.html` – a portmanteau of *folk* and *taxonomy*

[8] The "game layer" is the next stage in building virtual communities, the founder of SCVNGR argues – `http://www.ted.com/talks/seth_priebatsch_the_game_layer_on_top_of_the_world.html`

[9] Representational State Transfer-style access to data over HTTP/1.1. This particular API responds to requests in the JSON data interchange format.

3 Substance and Structure of the Project

The data for the project will be obtained over the Mixcloud API as JSON objects and stored in the document-oriented database, MongoDB. This will involve writing a crawler application to explore the social and content graph of Mixcloud via its API and save the relevant result in a useful way. As mentioned above, I have already begun doing this, writing a wrapper for the API in Python to use in my crawler application.

As I mentioned in the above section, I have no prior experience of the types of processing and algorithms used in building recommender systems and therefore I will have to spend some time examining the existing literature and open source code.

Building a recommender system involves the following broad stages:

- Identifying appropriate distance metrics to measure the similarity or relevance of item-item, item-user and user-user pairs;
- Computing the above metrics for the available data;
- Make recommendations with appropriate classification and filtering alogirthms, using these metrics.

Data for Distance Metrics

A common notion in recommender system design is that of "ratings" for a particular entity on some scale. Within the Mixcloud dataset, ratings are uniformly binary, that is, the existance of a connection between two entities signifies an implicit or explicit preference rating, and the lack of a connection means no implicit or explicit rating is available. These so-called ratings between entities can be used to measure the similarity or relevance, which I shall collectively call distance, of entities in the social network. The techniques used to measure distance between two entities of the same type can be similar whether the entities are Cloudcasts (items) or users. These can then be used to calculate the mixed-type distances, that is, the relevance of a Cloudcast to the active user and vice-versa.

A distance metric for measuring item-item distance could use the following ratings and objects as it's input data:

- The sets of users who have listened to/favourited/commented on/uploaded the two items in question
- The sets of tags and other metadata carried by the two items

Similarly, a metric for measuring social relevance – whether a user-user pairing should exist between two individuals – can use the following:

- Association: the two users' sets of current social links – followers, followees, profile commentors.
- Taste and Activity: the two users' sets of listening history/favourites/Cloudcast comments.
- In-Depth Taste: the metadata carried by the users' uploads/favourites/listening history.

The actual distance metrics that I will use will be subject to other implementation details.

Classification and Filtering

The fundamental process that my proposed recommender system will perform is to classify a relationship between two entities as either existing or not existing, taking into account the distance metrics discussed above. Given two users, two Cloudcasts, or a user and a Cloudcast, should a recommendation be made for linking them? The decision to be made is a binary one and the family of machine learning techniques used for this kind of processing are *binary classifiers*. These are a class of algorithms which employ probabilistic models derived from some training data to make prediction about further unseen data.

Reading the relevant literature, the classic approach to making recommendations is to use *Collaborative Filtering* (CF) techniques, which aim to make predictions for a given user or item based on data available for similar users or items. CF techniques can be split into two main types: memory-based and model-based. I am more interested in the model-based solutions to CF because I wish to build a recommender system that identifies complex patterns in data using machine learning techniques, as mentioned above. The memory-based approach is simple to implement but limited – essentially large sparse matrices of entity relationships/distances are stored and for a given entity the highest ranking corresponding entities are given as recommendation. This naturally has the problem that for changes in the set of entities recomputation of the matrices is required and also with large data sparsity, the computational performance of the system suffers.

4 Success Criterion

A recommender system attempts to predict and then facilitate the next interaction of a user [4]. How well it does this can be measured on a scale where the two extremes are the performance of a naive system where random entities are recommended, and at the other end a recommender system that reproduces the known ground truths exactly.

The challenge for me is evaluating my system without access to the real users of the system who would provide genuine ground truths regarding their behaviour[2].I can withhold some

of the links (ground truths) in the dataset and see how well the system is able to recover them.

Therefore the main method of evaluation I will use to judge the success of my recommender system will be cross-validation of its performance against a simple baseline. This is a common technique for assessing the performance of a supervised learning system which aims to make predictions on some test data, given some training data. This process would involve splitting the dataset into some number n of disjoint partitions and for each partition removing some of the links between entities (nodes in the network). The system is then optimised for making recommendations that repair the broken links in some $k, (k < n)$ subset of these partitions (the training data) and comparing its performance on the remaining $(n - k)$ partitions (the test data).

In the context of the Mixcloud dataset, only the presence of a link carries meaningful information about two entities relationship, and the lack of one does not necessarily entail a negative relationship. Some of the appropriate metrics that can be used to measure the performance of the system are the commonly used *precision* and *recall* rates [1]:

- Precision - the proportion of the produced results that recover the withheld ground truths
- Recall / Sensitivty - the proportion of the recovered ground truths to the total number of withheld ground truths

5 Extension Ideas

The Temporal Dimension

If obtain more than one snapshot of the social network across time, perhaps at 3-4 week intervals, I could use the snapshots as a means of evaluation by comparing recommendations made for the network at time (snapshot) $t-1$ with the changes made to the network at time t. This method of evaluation of the recommender system will compare the performance of the recommender system against the organic growth and changes in the real network since a previous snapshot.

6 Resources Required

See attached Project Resource Form. I have included a summary below:

- Mixcloud.com user and content data from their API – a cautious estimated size of 2GB

- PWF Filespace – Up to 500Mbytes
- SRCF Filespace – Up to 500Mbytes
- My own Laptop PC (Intel Core 2 Duo T6400 2.0GHz, 4GB RAM, 500GB HHD)

7 Timetable and Milestones

From the official beginning of this project on Fri 22nd October until the submission deadline of Fri 20th May, there are 30 weeks which I have timetabled as follows:

22 Oct – Submission of this Proposal

23 Oct – 5 Nov

Have a usable sample of the dataset gathered from the Mixcloud API and be comfortable with manipulating the database programmatically. Put in place backup strategies and project management tools and practice (version control system, mirrors, backup scripts, etc.).

6 Nov – 26 Nov

Implement the basic methods for calculating distance metrics. Ensure I have an understanding of the mathematical properties of the metrics and their respective suitability for my application.

27 Nov – 10 Dec

Implement some classifiers and Collaborative Filtering algorithms with appropriate distance metrics - these should produce meaningful output given small simple inputs. At the end of this time, I should be comfortable with the various data structures, algorithms and techniques available to me to implement a recommender system. I also need to ensure that I understand all the evaluation methods available for these techniques in detail.

11 Dec – 7 Jan

Using knowledge gained so far, build a recommender system that produces meaningful results given a large, realistic input, combining and experimenting with different techniques as discussed above.

8 Jan – 21 Jan

Start planning out the dissertation, with the Introduction and Preparation chapters being nearly complete and the Implementation chapter beginning to take form.

22 Jan – 3 Feb

Prepare for the progress report by testing the working recommender system and doing preliminary evaluation on it. Producing some visualisations and graphics from the data

and the output will also be helpful for this, as well as for the final dissertation. Produce the presentation for progress report.

4 Feb – Submission of Progress Report

5 Feb – 18 Feb

Work on any feedback from progress report. Begin work on more advanced techniques or novel approaches that I may have come up with after using the data, e.g. Temporal Analysis. Move onto extensions in earnest, if time is available. Set out a clear plan for evaluating the performance of the recommender, including the type of measurements to be taken and comparisons to draw.

19 Feb – 4 Mar

Make evaluation measurements on the performance of the recommender system, using appropriate methods and metrics explored previously, and draw comparisons between different approaches. Final clean up of code and Evaluation chpater of dissertation should be taking form. Dissertation document is regularly updated during this time and the Implementation chapter has taken form.

5 Mar – 25 Mar

Dissertation is comfortably on target in terms of content and word limit. Identify weaknesses in code and documentation, and address them.

26 Mar – 13 May

Final proof reading of dissertation and aim to submit a week before deadline.

20 May – Submission of Dissertation

References

[1] R. Eisner, Basic Evaluation Measures for Classifier Performance – `http://www.cs.ualberta.ca/~eisner/measures.html` – this page describes the terms *recall, precision, accuracy and specificity.*

[2] J. L. Herlocker et al., Evaluating Collaborative Filtering Recommender Systems, ACM Transactions on Information Systems, Vol. 22, No. 1, January 2004

[3] X. Su et al, A Survey of Collaborative Filtering Techniques, Advances in Artificial Intelligence, Volume 2009

[4] E. Vozalis et al., Analysis of Recommender Systems' Algorithms