Book-taste clustering in goodreads communities

Akanksha Tiwari (A0123476E)
Antoine Francois Pascal Creux (A0123427M)
Ashish Dandekar (A0123873A)

National University of Singapore,
Singapore

April 3, 2015

1 Detailed Problem Description

Goodreads is a website launched in early 2007, which lets "people find and share the books they like and improve the process of reading and learning throughout the world." It is the world's largest site for readers and book recommendations with a user base of about 30 million members along with 34 million reviews from 900 million books as recorded in 2015 [**?**].

Goodreads provides a multitude of features to its users. It goes beyond the traditional rating and reviewing of books by allowing users to make friends and join and form reading groups based on their literary tastes. Users can not only see what their friends have read, but they can also meet new people with similar reading interests. They can make recommendations to friends, follow authors, track the books they are currently reading, have read and want to read. In addition, goodreads provides personalized recommendations to book readers by analyzing the user data.

Basically, we want to detect and study goodreads communities based on books reviewing. Two people who has given similar rating for the same book share a similar interest, and thus are closed. We also want to improve this homophily by giving a negative score to different rating of the same novel. We wonder if communities based on book tastes are analogous to friends communities, or what the average homophily score is between two friends?

2 Related Work

3 Description of data collection and pre-processing processes

This task is very tedious as users dataset is not available online. Goodreads API is well documented and easily accessible. All we need is a developer key and an OAuth access to have access to all public profiles. Indeed, a goodreader can make its profile private and thus, only accessible from its friends. Given that goodreads users database being huge, we hopefully have enough public profile to run our analysis.

The data scraping is laborious and time consuming, but its execution is vital in our study. Besides, a biased and wrongly data scraping may take a lot of time and return fragment and

unusable data. The main issue is how we can get enough data while keeping it unbiased. Random issues can be queried but first sample of reviews show that no reviews were sharing the same reader or the same book. All reviews written by a goodreader is available by using the API point *owned_books.list* while we cannot retrieve all reviews from a book.

We figured out two ways to collect the data:

- Retrieve users id from a group.
  Goodreads enables virtual groups where people share similar interests or just get along with each other. Some groups are around a specific journal while others just give to all members a reading schedule so that everyone can share its opinions with this group. One of the biggest group is Goodreads Authors/Readers. According to goodreads, this group is dedicated to connecting readers with Goodreads authors. It is divided by genres, and includes folders for writing resources, book websites, videos/trailers, and blogs. Thus, we can query all the members, and then use their first connection to get more users while keeping our dataset unbiased.

- Retrieve users id from reviews of books.
  All the book genres are available at **??**. We can select the genres whose number of reviews is the biggest, and retrieve the reviews from the best-sellers of each top category. One complication was raised as there is no API that retrieves reviews from a book id. Anyway, a meticulous analysis of *Javascript* call when browsing the goodreads Web pages show that requests are made to *book/reviews/*. It is a *Javascript* code that refresh the reviews elements of the HTML page. We can retrieve the update information, aka the user_id and rating given, by parsing the *Javascript* callback using regular expressions.

We have decided to use the group-based scraping. We are sure we will not get biased data, and using the friends, we should get enough data points to run the analysis.

4 Description of the method

5 Discussion of project progress

5.1 Data Collection

We have queried the first reading group entirely. The statistics computed based on this first scraping pass are:

- 567461 ratings
- 7815 public profiles
- 2564 private profiles
- 7203 profiles without a review
- 1948 profiles

We did not know we would get so many fake or private profiles. Within a group of 20000 members, we got less than 8000 users.

We decided to scrape all the friends of the users we have found to get more data. Among those 128213 friends, we have removed the profiles already in the group, which are private or do not have

any review to store a list of 95503 active profiles who have likely written more than 5 million ratings.

To accelerate the scraping, we use multiprocessing and we set up a Tor connection, even if we do not knwo we will used it.
While getting the last data, we worked on the algorithm part.

5.2 Data processing

5.3 Algorithm

We focus on the study of communities detection based on the user graph we have generated.

The most common

Community detection is a widely discussed topic in the social networking literature[1][**?**].

Based on the sample we have, the graph is very dense. With less than 8000 nodes, we have more than 16 million edges. The density is about ???.

Based on [**?**], we wanted to test which community detection algorithm could be used. At first, we were supposed to use the Girvan-Newman algorithm as it was the most common community detection algorithm[**?**]. The GirvanNewman algorithm uses an "edge betweenness" of an edge as the number of shortest paths between pairs of nodes that run along it. Edges with high betweeness are very likeyly the connection between communities. Thus, removing the edges in the decreasing order of between should reveal the dendogram associated, and thus the communities. This compute takes $O(ne)$ running time, and thus is too long for our graph.

G-N can be limiting as any node has to be placed in one unique community. Since, we deal with a bipartite graph, we can we may try to find complete bipartite subgraphs. That means we can split the users and the books into two distinct sets respectively, such as $users0$ and $books0$, and $users1$ and $books1$ are linked, but no edge relate $users0$ and $books1$, or $users1$ and $books0$. Based on our graph density which is pretty high, we cannot apply this algorithm: we would a unique complete clique. However, we may filter the edges by for instance, keeping only the edges with a rating of 5.

The Simrank method, where a random walker starts at one node, and is allowed to go through the edges is cannot also be applied to our dense graph. The similarity between each node is too time-consuming.

[1]Social networks are generally sparse networks. When the networks are dense, the community detection does not seem to be a wise option. The dense networks possess large number of inter-community edges which yield poor partitioning. For dense networks, discrete data analysis techniques are used.