Recommendit

A Recommendation Service for Reddit Communities

Suphanut Jamonnak, Jonathan Kilgallin, Chien-Chung Chan, En Cheng
Department of Computer Science
The University of Akron
Akron, OH

Abstract—Reddit.com is a popular website consisting of many online forums, or "subreddits", each centered on a particular topic. Many of these subreddits represent, prima facie, closely related topics. However, two communities that might appear to be similar may in fact have little overlap in their user-base or content, while apparently dissimilar subreddits may actually represent topics that appeal to such similar audiences that their users do substantially overlap. At present, there lacks effective methods of identifying these relations. In this paper, we propose and implement an automated method of identifying such relations using association rule mining. Given the output from association rule mining algorithm, we develop a Python-based web application named *Recommendit* that a user can query and retrieve subreddits closely associated with a given subreddit.

Keywords—Data mining; Association rule mining; Reddit.com; Web applications; Recommendation services

I. INTRODUCTION

Reddit is a social network and content aggregation site launched in 2005. Users create posts containing text, photos, or links, to a particular "subreddit" - a forum on the site centered around a single topic. As of April 2015, Reddit had over eight thousand subreddits, ranging across a wide variety of popular topics such as "jokes", "art", "world news", "gaming", and "recipes", as well as more esoteric subjects like "haskell", "doctorwho", and "cyber laws". Users may view posts recently submitted to a subreddit, and can also contribute to a crowd-based ranking system by voting posts up or down to indicate approval or disapproval. A user may also comment on a post, reply to a comment, and vote comments up or down as well. The interactive nature of the site can best be illustrated by visiting the site directly – for example, at "www.reddit.com/r/python" [1] for the forum about the Python programming language – though note that some actions require account creation and login.

The Reddit homepage consists of a feed of popular and recent posts across a set of different subreddits. The nature of the set of included subreddits depends on whether a user is logged in or not. A user browsing anonymously (i.e. without logging in) will see posts from among a select group of "default" subreddits, which are hand-picked by the site administrators according to the subreddit's perceived quality and applicability to a broad audience. If a user is logged in, however, the front page will show primarily content from subreddits to which the user has chosen to subscribe. A user can subscribe or unsubscribe from any subreddit from either the subreddit itself or from a list of subreddits available through a personal settings menu. A particularly popular and recent post from a subreddit the user has not subscribed to may also occasionally appear on their front page, but generally, the relevance, quality, and quantity of posts that appear in this feed depend on the user's selection of an appropriate set of subreddits. As the feed of posts on the front page is so integral to the site, and as the site entertains hundreds of millions of visitors per month, the task of selecting a set of subreddits is important for users' engagement on the site.

In this paper, we present an automated mechanism for a user to discover subreddits of potential interests by examining associations between a pair of subreddits based on the overlap in the user-base of the two. We first crawl existing archives of posts from Reddit site; and for each user, list the set of subreddits that he posted comments. Then, we apply association rule mining algorithm to associate subreddits. Based on the results from association rule mining, we develop an easy-to-use web-based application named *Recommendit* that a user can query and retrieve subreddits closely associated with a given subreddit.

II. RELATED WORK

The process of discovering subreddits generally requires a user to actively search and browse through a list of subreddits, or else to come across links to new subreddits more organically through other posts on the site or elsewhere. Such links may regularly be found within comments to a post in a related subreddit. For example, a user from the United Kingdom requesting help from the "/r/legaladvice" forum may likely receive a comment directing him to "/r/legaladviceuk". A subreddit's main page may list a limited set of ostensibly related subreddits, as determined by the moderators of that subreddit. For example, toward the bottom of the sidebar on the r/Python page, a list is included as shown in Figure 1 [1]. With only this system, a user could go months or years on the site remaining unaware of the existence of a subreddit pertinent to the user. Furthermore, in any case, all of these discovery methods rely on input from other users, which may at any time be incomplete or out of date.

Some third-party sites aim to facilitate this discovery process. For example, the site "subreddits.org/" and other third-party sites group subreddits by category; again, the site development team among other shortcomings largely builds these by hand. At least a few projects do aim to programmatically build recommendation engines though. In one case, Sundaresan et al build weighted graphs to identify communities on the site [2]. A github project - also called "Recommendit" by "ben444422", with no updates in 2 years,

appears to have begun on a similar path to the one we intend, but did not seem to get very far [3].

Related subreddits

- /r/pythoncoding (strict moderation policy for 'programming only' articles)
- /r/flask (web microframework)
- /r/django (web framework for perfectionists with deadlines)
- /r/pygame (a set of modules designed for writing games)
- /r/IPython (interactive environment)
- /r/inventwithpython (for the books written by /u/AlSweigart)
- /r/pystats (python in statistical analysis and machine learning)
- /r/coolgithubprojects (filtered on Python projects)
- /r/pyladies (women developers who love python)
- /r/git and /r/mercurial - don't forget to put your code in a repo!

Fig. 1. Subreddits related to /r/python

Reddit itself, through its API, offers a relatively similar mechanism for recommending subreddits. While the sidebar on a subreddit may give a handpicked list of similar subreddits based on the subreddit moderators' opinion, the Reddit API provides a method for recommending subreddits based on common appearance in users' custom-made "multireddits" – that is, sets of subreddits that a user may define and view posts from concurrently. This method – "get_subreddit_recommendations", takes as input a subreddit and returns a list of related subreddits, just as ours does [4]. However, with the API method, the recommendations are based on what users *view* together, while ours is based on activity actually occurring on the subreddits. We believe that this approach, in that it reflects the overlap in users active on both subreddits of a pair, can more accurately predict the degree of common interest in the two topics.

III. METHODOLOGY

To facilitate the process of discovering subreddits pertinent to a user, we propose an association-rule based recommendation engine, associating two subreddits when a large number of individual users post to both. The degree to which the user-base overlaps provides insight into the degree to which interest in the topics themselves overlap.

The general process of association rule mining consists of identifying "transactions", each containing multiple "items", and selecting those items that appear together in sufficiently high proportion. In our case, an "item" is a subreddit and a transaction is a set of subreddits that a given user frequently posts to. By mining Reddit for many users' post history, we create rules to identify and associate subreddits that have several joint users by using the "Apriori" association rule-mining algorithm, with appropriate support and confidence thresholds (as discussed in section V). From the rules produced, we develop a web application that allows a user to input a subreddit of interest, and returns related subreddits which may be of interest to the user as well.

To implement this approach to identifying similar subreddits, we perform the following five steps, further illustrated in Fig. 2:

1. For each of the top 2500 subreddits by subscriber count, as obtained from redditlist.com [5], we scrape Reddit to obtain the set of users who have made a popular post to the subreddit this year. This is done with a Python program using Reddit's API [6] and is stored in a Comma-Separated Value file (CSV) for each subreddit. This dataset is augmented with the top 1000 posts of all time as of August 2013, downloaded from [7]. See "Data Collection" below for more information on this part.

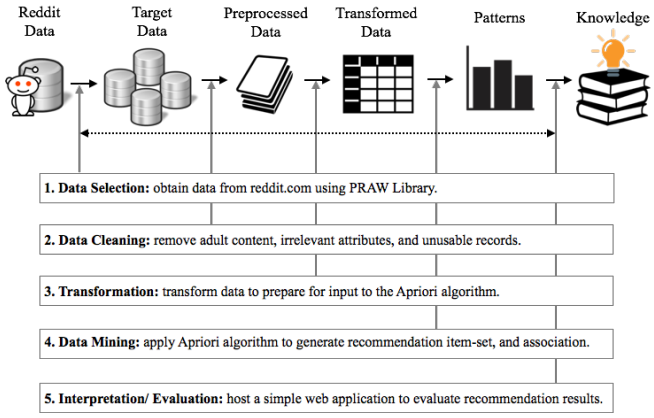


Fig. 2. Recommendit Methodology

2. We then strip the irrelevant attributes and unusable records from the data collected in step 1, and remove posts containing adult content or missing critical data.
3. Third, we group records by author in order to form a list of transactions as described in our formulation above, in preparation for provision as input to the Apriori algorithm. This is done by loading the CSV files into SQLite and using SQL queries to transform the data.
4. We next pass the list of transactions to an implementation of the Apriori algorithm written in Python [8], in order to find the list of association rules.
5. Using the association rules discovered, we host a simple web application, also written in Python and using the CherryPy library [9], that allows a user to query for the set of subreddits associated with a given input subreddit.

These steps, taken together, allow users to find subreddits of interest. Due to the growing and shifting nature of Reddit communities, steps 1 to 4 should be repeated periodically - perhaps on the order of once a month. This is an informal estimation, but refreshing less frequently may cause data to become excessively stale, while more often may waste effort without significantly changing the results.

IV. Data Collection And Processing

A. Data acquisition

The data we require consists of the author and subreddit for as many posts as we can collect, store, and process. There are many ways to obtain such data - from Reddit itself we can scrape the HTML site or use their provided API. In this case, we have used a Python library called "Python Reddit API Wrapper", or PRAW [10]. PRAW is a Python package that allows for simple access to Reddit's API. It aims to be easy to use and is also designed to respect all of Reddit's API rules. Another approach, which we also make use of, is to rely on data previously collected by others and hosted elsewhere. Fortunately, an existing github project provides an older, but still usable, copy of exactly the data we are interested in - post information from the top 2500 subreddits by subscriber count [7].

We created a crawling application implemented in Python using PRAW called "subrtime.py". This application takes as input the name of a subreddit, along with starting and ending timestamps, then will crawl the subreddit's posts within the given timestamp range and return data regarding the included posts and their attributes. After collecting 1000 posts each from these 2500 subreddits, we are left with 2.5 million posts.

B. Data preprocessing

In this step, we clean the data to preserve only the posts we are able to use. Specifically, in order to create the recommendation system, Recommendit requires the author name in order to compute a transaction set for the Apriori algorithm. Thus, we have to remove posts with "none" or "NULL" values for the author. Moreover, we are also removing posts marked "Not Safe For Work", or NSFW. These are posts with adult or explicit material, which in general the site allows but we wish to disregard. We have removed the missing-author and NSFW posts by writing another Python script "transubreddit.py" to filter the data in the SQL library. This decreased the number of records from 2.5 million to 2.4 million; i.e. by about 4% of our original data set.

C. Data transformation

In order to apply the Apriori algorithm, these records need to be grouped by author into sets of subreddits the author posts to. This is accomplished with SQLite and SQL queries to group the records. The nature of the transformation is illustrated in Fig. 3.

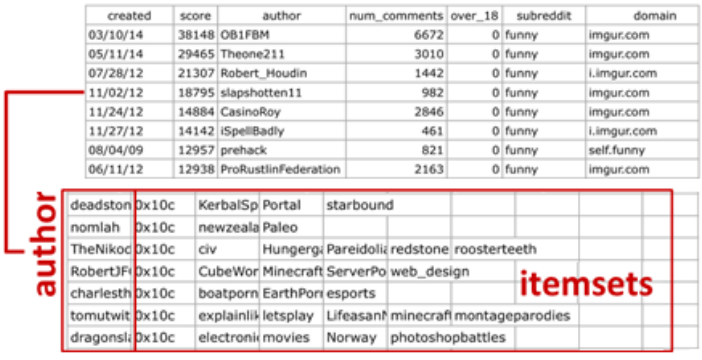


Fig. 3. Grouping records into itemsets by author

V. Apriori Application

Discovering interesting relations between variables in large databases is a popular and well-researched problem in data mining. These techniques are very popular in several fields such as marketing strategy, data analytics, machine learning and so on. For instance, in marketing strategy, association rule mining may be used for discovering patterns of purchase behavior in large-scale transaction data recorded by Point-Of-Sale (POS) systems in a supermarket.

For example, "{milk, bread}⇒{butter}", found in sales data of a supermarket, can indicate the premise that, if a customer buys bread and milk together, they may be also likely to buy butter. Such information can be used and adapted for several marketing activities such as promotional pricing, product placements, and so on.

Apriori is a classic algorithm for solving and finding association rules, introduced by Agrawal and Srikant in 1994 [11]. This technique is popular for this type of data mining problem, and although it is not the only algorithm for solving this problem, it is sufficient for our case. We address only the necessary concepts for our application here, but further information on the algorithm can be found in Agrawal [11] or in many texts or other resources on data mining.

An association rule is defined through the following terms:

- Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of m binary attributes called "items".
- Let $T = \{T_1, T_2, \ldots, T_n\}$ be a set of transactions, where $T_i \subseteq I$.
- A rule is an implication of the form $X \Rightarrow Y$, where $X, Y \subseteq I$ and $X \cap Y = \emptyset$.

To illustrate this concept, we use a small hypothetical example from Reddit. The set of subreddit items is I = {*Python, Compsci, Django, Java*} and a small set of transactions is coded with 0 for absence and 1 for presence. Consider Table 1:

TABLE I. Itemset Database And Structure

Transaction ID	*Python*	*Compsci*	*Java*	*Django*
Author1	0	1	0	1
Author2	1	1	0	1
Author3	1	1	0	1
Author4	0	0	1	1
Author5	1	1	0	1
Author6	1	1	1	0

We can predict that a user interested in *Python* and *Compsci* is interested in *Django* as well. In larger cases of association rule mining, an item set may need to appear together many times before it can be considered significant. Our dataset contains millions of transactions, and so constraints on the associations made are needed. The best-known constraints are minimum thresholds on support and confidence, as defined below:

- *Support* - The proportion of transactions in the data set which contain the item-set.

supp(X) = number of transactions which contain the item-set X / total number of transactions

- *Confidence* – The fraction of transactions containing the items from itemsets X and Y, over the items from X alone.

$$conf(X \Rightarrow Y) = supp(X \cup Y) / supp(X)$$

In the example item-set database above, the item-set {*Python, Compsci, Django*} has a support of 3/6 = 0.50 since it occurs in 50% of all transactions. For the rule {*Python, Compsci*} => {*Django*} we have the following confidence: supp({Python, Compsci, Django}) / supp({Python, Compsci}) = 0.50 / 0.66 = 0.75. This means that for 75% of the transactions containing "Python" and "Compsci" the rule is correct.

Apriori uses a bottom-up search to count candidate itemsets efficiently. Starting with sets of a single item, (i.e. L_1=I) it generates "candidate itemsets" of length K from itemsets of length K-1. Then, since any candidate set cannot meet the support and confidence thresholds unless ALL of its subsets of length K-1 do as well, it prunes the candidates of length K whose (K-1)-length subsets do not all appear in L_{K-1}, to finally achieve L_K. The algorithm terminates when it reaches some round J where $L_J = L_{J-1}$. This makes it effective for exploring transactions with a large number of items and transactions. In this project, we make use of an existing implementation of the Apriori algorithm in Python, which takes the transaction list along with support and confidence thresholds, and outputs the set of subreddits associated with each subreddit in the dataset [10].

//TODO: support and confidence thresholds.

In Apriori algorithm [8], the default thresholds of support and confidence are set between 0.1 – 0.2 (support) and 0.5 – 0.7 (confidence). After couple time of testing, and executing Apriori algorithm, we found out that the number of recommendation results were changed drastically depending on the decreasing of support value. And also changed with the increasing elements of the transaction pattern As a result, the most optimal thresholds for support and confidence is equal to support = 0.05, and confidence = 0.60, in order to getting the best recommendation results with Approximately 200,000 users transaction.

VI. Recommendation Service

The output of the Apriori algorithm now gives us the data we need to host *Recommendit* as a Python-based web application. We have developed a simple version of such a service, which is hosted at "kilgallin.com" as of Sept 2015. This version allows a user to enter the name of a subreddit in a webform, and returns a formatted list of subreddits with links directly to the subreddit so that a user can quickly examine the forum and determine if it is actually of interest. It is suitable to be used as a web service or integrated into a richer application. It is important to note, however, that the underlying data will quickly become stale as the Reddit communities evolve, and repeated mining would be necessary to maintain such a service.

VII. Results

With the web application hosted, we can now easily examine the results of our mining process and evaluate their usefulness to the Reddit community at large. Looking at the subreddits related to "mycology" (i.e. the study of mushrooms and fungi), for example, we see the following list:

- /r/whatsthisplant
- /r/gardening
- /r/botanicalporn
- /r/entomology
- /r/biology
- /r/spiders
- /r/botany
- /r/permaculture

These results certainly capture multiple types of association with a mycology forum. Some entries, such as "/r/whatsthisplant", "/r/entomology", and "/r/spiders", have in common the feature of containing many posts containing photos of their respective category of natural plants and animals. Some, such as "/r/gardening" and "/r/permaculture", may appeal more to enthusiasts of growing mushrooms for culinary use. Finally, "/r/biology" and "/r/botany" both capture broader categories that subsume the field of mycology. Compare this to the "related subreddits" list actually appearing on the mycology subreddit:

- /r/Slimemolds
- /r/seedstock
- /r/Mycoporn
- /r/MushroomPorn
- /r/Gardening
- /r/PhysicGarden
- /r/InvasiveSpecies
- /r/MushroomGrowers

This list consists of subreddits more directly addressing the study of fungi, with only "gardening" appearing on both. One may question why these would not show up in our web service. The reason for this is that many of these are smaller subreddits, which do not make the top-2500 cutoff, which is certainly a limitation of our results, as discussed below. Conversely, though, the related subreddits listed omit several subreddits of potential interest to a mycology enthusiast, such as botany. These lists thus serve to complement each other effectively.

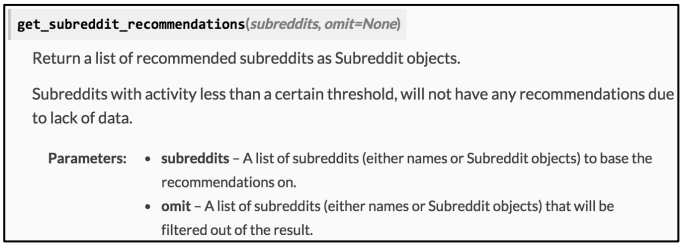
//TODO: more results, compare to get_subreddit_recommendations

A. *Evaluation*

We evaluate the precision and accuracy of Recommendit using this formula:

$$\text{P-Value} = |A| \cap |B| \ / \min(|A|, |B|)$$

Where, A and B are the sets of recommended subreddits between two groups. (1) Recommendit group and (2) for the existing PRAW library group, in addition; we now collecting the recommendation results from all of 4,000 subreddit listed, and ranked by subscribers corrected from redditlist.com [12].



`get_subreddit_recommendations`(*subreddits, omit=None*)

Return a list of recommended subreddits as Subreddit objects.

Subreddits with activity less than a certain threshold, will not have any recommendations due to lack of data.

Parameters:
- **subreddits** – A list of subreddits (either names or Subreddit objects) to base the recommendations on.
- **omit** – A list of subreddits (either names or Subreddit objects) that will be filtered out of the result.

Fig. 4. Existing PRAW subreddit recommendation functions

We implemented precision calculator, in order to comparing, and apply the P-Value between those two groups. Below is the example of usage by inputting *"/r/naslsoccer" (North American soccer)* as the subreddit input.

```
-------------------------------------------------
Recommendit & Praw comparision table
-------------------------------------------------
input subreddit: /r/naslsoccer
+-----------------+-------------+
|   RECOMMENDIT   |    PRAW     |
+-----------------+-------------+
|     football    | FantasyMLS  |
|       NWSL      |     NWSL    |
|  PremierLeague  |  KitSwap    |
|   indianapolis  |     MLS     |
|       MLS       |  ussoccer   |
| minnesotaunited |      -      |
|       rbny      |      -      |
|     OCLions     |      -      |
|     ussoccer    |      -      |
+-----------------+-------------+
-------------------------------------------------
intersection = 3 subreddit
P-Value = 60.0%
-------------------------------------------------
```

Fig. 5. Comparing precision value between Recommendit and Praw

From figure above, its appear that the set of recommended results of /r/naslsoccer are 9 from Recommendit and 5 from PRAW library. And the number of the intersection subreddit is equal to 3. As a result, 3 / 5 is 0.60, which equal to 60.0% in percentages.

After running in all iteration for ~4,000 subreddits as input. We now evaluate all of the subreddits input, which have the precision value fall between: 60.00% to 100.00%. However, the subreddits that have the precision value fall between 0.00% to 59.99%, can be considered as an additional recommendation result from the existing library.

B. *Knowledge Discovery Comparision*

After testing on all of the iteration, we can conclude that PRAW and Recommendit share the similarity of recommendation results as expected. Here are the 3 example lists of subreddits with 100% in P-Value of recommendation.

- /r/indianauniversity
- /r/ohiostatefootball
- /r/lawschool

1) *Recommendation Results*

Here are the examples of recommendation table, which P-Value equal to 100 percentages in recommendation results.

- *Equality Comparison* – according to the result listed in the table below (TABLE II.). Its shown us that Recommendit and PRAW can share a similarity in term of the recommendation results.

TABLE II. INPUT: /R/INDIANAUNIVERSITY

	Recommendit	PRAW
Input subreddit: /r/indianauniversity	/r/bloomington /r/indianapolis /r/indiana	/r/bloomington /r/indianapolis /r/indiana

- *Inequality Comparison* – according to the result below (TABLE III.) and (TABLE IV.), Its shown that Recommendit and PRAW also have the different recommendation capability depending on the input of subreddit.

TABLE III. INPUT: /R/OHIOSTATEFOOTBALL

	Recommendit	PRAW
Input subreddit: /r/ohiostatefootball	/r/OSU /r/MichiganWolverines /r/Columbus /r/Reds /r/WahoosTipi /r/clevelandcavs /r/cfbmemes /r/Ohio /r/Browns /r/mlb	/r/OSU

TABLE IV. INPUT: /R/LAWSCHOOL

	Recommendit	PRAW
Input subreddit: /r/lawschool	/r/law	/r/law /r/lawschooladmissions /r/LSAT /r/LawFirm /r/VetTech /r/legal

VIII. Conclusion & Future Work

We have shown that association rule mining approach can be useful and effective for identifying relations between subreddits, and that this can be used to facilitate the process of discovering additional Reddit communities of potential interest to a user. While there is ample room for future development of this technique, the results we obtained in this paper are indicative of a promising automated method for such associations.

While our process appears to be sound for the data we collect, there are some limitations that could be overcome in future work. First, by only looking at the 2500 largest subreddits, we miss out on the ability to suggest new or small subreddits. Since these are the subreddits that are typically the hardest to discover on one's own, this is a potentially valuable application of our program - perhaps even one of the most valuable from a discovery viewpoint, and one that could help smaller communities to grow effectively. However, since the 2500th subreddit has only 0.01% the user count of the top few subreddits, the next 6000 subreddits will, cumulatively, only correspond to the number of users in the top 4-6 subreddits. The overhead to scrape and store the data for these subreddits is an important factor we have to consider when we expand the scope of the project in the future. Second, our web app architecture as written is limited in its ability to serve a large amount of traffic. Since Reddit has several million unique daily visitors and hundreds of millions of unique monthly visitors, we need provide a scalable application, which can serve millions of concurrent sessions in the future.

References

[1] Reddit. (2008). *Python* [Online]. Available: http://www.reddit.com/r/python

[2] V. Sundaresan et al. (2014, Dec 9). *Subreddit Recommendations within Reddit Communities* [Online]. Available:. http://web.stanford.edu/class/cs224w/projects/cs224w-16-final.pdf

[3] ben444422. (2013, May 12). *Recommendit* [Online]. Available: https://github.com/ben444422/Recommendit/

[4] Shulurbee. (2013, Aug 1). *New beta feature: subreddit suggestions* [Online]. Available: https://redd.it/1jisr2

[5] Redditlist. (2010, June 19). *Reddit list* [Online], Available: http://www.redditlist.com

[6] Reddit. (2012, Jan 1). *Reddit API documentation* [Online[. Available: https://www.reddit.com/dev/api

[7] Umbrae. (2013, Aug 27). *Reddit Top 2.5 Million* [Online]. Available: https://github.com/umbrae/reddit-top-2.5-million

[8] Asaini. (2015, Jul 1). *Python Implementation of Apriori Algorithm for finding Frequent sets and Association Rules* [Online]. Available: https://github.com/asaini/Apriori

[9] The CherryPy Team. (2015). *CherryPy: A Minimalist Python Web Framework* [Online]. Available: http://cherrypy.org/

[10] B. Boe. (2015, Sept 4). *PRAW: The Python Reddit API Wrapper* [Online]. Available: https://github.com/praw-dev/praw

[11] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases" in *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, 1994, pp. 487-499.

[12] Rank of subreddits by subscriber [Online]. Available: http://www.redditlist.com