Recommenddit

A Recommendation Service for Reddit Communities

Suphanut Jamonnak, Jonathan Kilgallin, Chien-Chung Chan, En Cheng
(sj70@zips.uakron.edu), (jdk72@zips.uakron.edu), (chanc@uakron.edu), (echeng@uakron.edu)
Department of Computer Science
College of Arts and Sciences
University of Akron
Akron, OH 44325-4003

Abstract—Reddit.com is a popular website consisting of many online forums, or "subreddits", each centered on a particular topic, many of which are closely related. 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 substantially overlapping audiences. At present, there is a lack of effective methods for identifying these relations. In this paper, we propose and implement an automated method of identifying such relations using an association rule mining algorithm. Given the output from this algorithm, we develop a web-based application named *Recommenddit* that a user can query to 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 users to discover subreddits of potential interest. We accomplish this by examining associations between two subreddits based on the overlap in the user-base of the two subreddits. We first crawl existing posts from the Reddit website and from other archives. We list the subreddits each user posts to. Then, we apply an association rule mining algorithm to associate subreddits. Based on the results from association rule mining, we provide an easy-to-use web-based application named *Recommenddit* that a user can query to retrieve subreddits closely associated with a given subreddit.

II. Related work

The process of discovering subreddits generally requires a user to browse through a list of subreddits, or else to come across links to new subreddits elsewhere on the site or internet. For example, a user from the United Kingdom requesting help from the "/r/legaladvice" forum may 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.

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.64

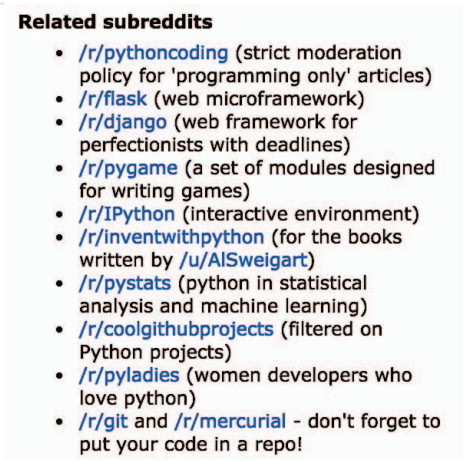

Figure 1. Subreddits related to /r/python

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 teams largely build 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 called *Recommendit* [3] appears to have begun on a similar path to the one we develop, but has had no updates in 2 years.

Reddit itself, through its API, offers a 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. The *get_subreddit_recommendations* method [4] takes a subreddit as input and returns a list of related subreddits. However, with the API method, the recommendations are based on what users view together, while ours is based on content actually hosted on the subreddits. We believe that our approach, in that it reflects the overlap in the user-bases of two subreddits, 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 identify similar subreddits, we perform the following five steps, further illustrated in Fig. 2:

1. For each of the top 4000 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 section IV.A for more information on this part.
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. This is explained in section IV.B.
3. Third, we group records by author in order to form a list of transactions as described in our formulation above, in preparation for passing to the Apriori algorithm. This is done by loading the CSV files into SQLite and using SQL queries to transform the data, and is elaborated in section IV.C.
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. See section V for details on this step.
5. Using the association rules discovered, we host a simple web application, also written in Python and using the CherryPy library [9]. It allows a user to query for the set of subreddits associated with a given input subreddit. This is discussed in section VI.

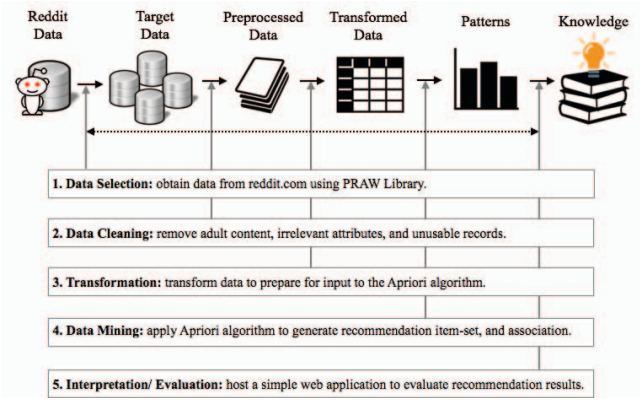

Figure 2. Recommenddit Methodology

These steps, taken together, allow users to find subreddits of interest. Due to the growing and shifting nature of Reddit communities, all steps 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 we 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 usable data - 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 around 1000 posts from each of 4103 subreddits in the augmented results, we are left with 4 million posts.

B. Data preprocessing

In this step, we clean the data to preserve only the posts we are able to use. Strictly speaking, Recommenddit 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 also remove posts marked "Not Safe For Work" (NSFW) or "over_18". These are posts with adult or explicit material, which in general the site allows but we wish to disregard. We removed the missing-author and NSFW posts by writing a Python script called transubreddit.py to filter the data. The preprocessing step decreased the number of records from 4 million to 3.7 million; i.e. by about 7.5% of our original data set.

C. Data transformation

To apply the Apriori algorithm, all post records need to be grouped by authors. In other words, we need group the records to a set of subreddits that each author posts to. We use SQL queries to group the records and develop a Python script called Transaction_Builder.py to accomplish the data transformation task. The script automatically process post records, selects author, and generates a set of distinct subreddits for each author. After one week of execution, 3.7 million records were grouped to ~200,000 authors with a set of subreddits. Fig. 3 shows an example of data transformation process.

Created	*Score*	*Author*	*Num_comment*	*Subreddit*
12/30/14	8379	Author1	1194	InternetIsBeautiful
05/04/15	7318	Author1	1763	crusadersquest
08/18/15	6505	Author2	554	startups
05/19/15	5773	Author2	906	Meditation
07/16/15	5068	Author2	1463	InternetIsBeautiful
04/08/15	5037	Author3	2543	worldnews
06/07/15	5012	Author3	297	rollerblading
08/26/15	5018	Author3	612	PostHardcore
01/13/15	4999	Author3	1258	InternetIsBeautiful
07/24/15	5006	Author4	307	TrueReddit
05/11/15	5126	Author4	783	Frisson
11/23/15	6773	Author4	241	Heavymind
06/19/15	4802	Author4	1587	apple

Author	*Subreddits*			
Author1	crusadersquest	InternetIsBeautiful		
Author2	startups	Meditation	InternetIsBeautiful	
Author3	worldnews	rollerblading	PostHardcore	InternetIsBeautiful
Author4	TrueReddit	Frisson	Heavymind	apple

Figure 3. An example of grouping records by authors

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 indicates 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, in TABLE I. we use a simple example from our data. The set of subreddits are {*Python, Compsci, Django, Java*}.

TABLE I. EXAMPLE TRANSACTION AND STRUCTURE

Author	Subreddits		
Author1	Compsci	Django	
Author2	Python	Compsci	Django
Author3	Python	Compsci	Django
Author4	Java	Django	
Author5	Python	Compsci	Django
Author6	Python	Compsci	Java

We can predict that a user interested in *Python* and *Compsci* is interested in *Django* as well. In larger cases of association rule mining, a set of items 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 input transaction as a subset.

$$support(X) = \frac{|\{T_i \in T | X \subseteq T_i\}|}{|T|}$$

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

$$confidence(X \Rightarrow Y) = \frac{support(X \cup Y)}{support(X)}$$

In the example database above, the 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 sets" of size K from sets of size 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 [8], which takes the transaction list along with support and confidence thresholds, and outputs the set of subreddits associated with each subreddit in the dataset.

In the Apriori implementation [8], the default threshold of support is set between 0.1 – 0.2, and 0.5 – 0.7 for confidence. After some test executions of the Apriori algorithm on our dataset, we found that the number of recommendation results increased drastically when decreasing the support value. We prefer to be liberal with our recommendations, so we selected thresholds for support and confidence as: support = 0.05 and confidence = 0.60, in order to get appropriate recommendation results with approximately 200,000 users' transactions.

VI. Recommendation Service

The output of the Apriori algorithm now gives us the data we need to host *Recommenddit* 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 web-form, and returns a formatted list of related 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

A. Precision

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. We evaluate the precision of Recommenddit using this formula:

$$Precision = \frac{|A| \cap |B|}{min(|A|, |B|)}$$

Here, A and B are the sets of subreddits recommended by two different methods. We use the Recommenddit results for group A and the existing Reddit API method "get_subreddit_recommendations" for group B [4]. We also perform a comparison to the moderator-provided list of related subreddits listed on the sidebar of the input subreddit.

We implemented a precision calculator in order to perform the comparison and apply the precision formula between those two groups. Fig. 4 is an example of usage by inputting *"/r/bloomington"* as the subreddit.

```
-------------------------------------------------
Recommendit & Praw comparision table
-------------------------------------------------
input subreddit: /r/bloomington
+-------------------+-------------------+
|    RECOMMENDIT    |        PRAW       |
+-------------------+-------------------+
|    indianapolis   | IndianaUniversity |
|      Indiana      |     fortwayne     |
| IndianaUniversity |    indianapolis   |
|         -         |      Indiana      |
+-------------------+-------------------+
-------------------------------------------------
intersection = 3 subreddit
P-Value = 100.0%
-------------------------------------------------
```

Figure 4. Comparing precision value between Recommenddit and Praw

From Fig. 4, the number of recommended results of /r/bloomington is 3 from Recommenddit and 4 from the Reddit API. The size of the intersection is 3, so we have a precision of 3 / 3 = 1.00, or 100.0%. Since the precision formula is undefined if one set is empty, we consider only subreddits for

which both services produce at least one recommendation. From the original 4,000 subreddits, only 2,908 pass this requirement, for which the precision results are summarized and illustrated in TABLE II and Fig. 5

TABLE II. PRECISION PERCENTAGE AND NUMBER OF SUBREDDIT

Precision	Number of subreddits
0% - 9.9%	1768
10% - 19.9%	242
20% - 29.9%	265
30% - 39.9%	204
40% - 49.9%	78
50% - 59.9%	177
60% - 69.9%	55
70% - 79.9%	15
80% - 89.9%	10
90% - 99.9%	0
100%	49

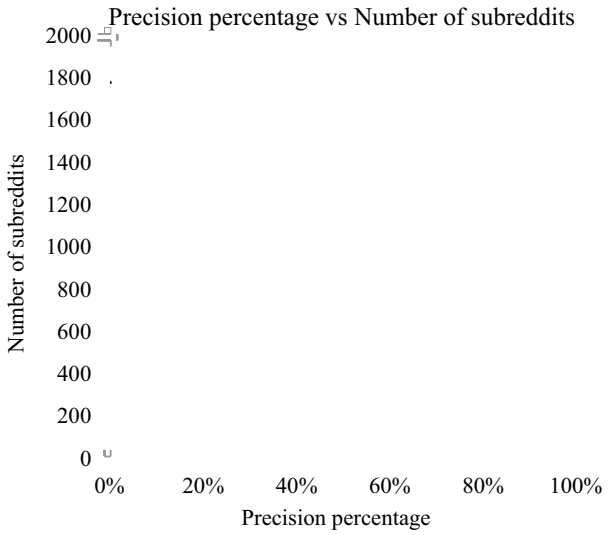

Figure 5. Number of subreddits by precision percentages

This shows that while, for many subreddits, the results are comparable, we see that for most others, there is no or little overlap in the two lists. However, we see this as an indication that our results may effectively augment the results from the Reddit API. In subsection B we provide 3 example subreddits with 100% precision, each with a different reason for such a result, along with 2 examples having 0%. For these examples, we perform a 3-way comparison between the results from Recommenddit, those from the Reddit API, and the moderator-selected results appearing on the sidebar for the subreddit page. The subreddits we consider are:

- /r/indianauniversity
- /r/Unity3D
- /r/lawschool
- /r/lectures
- /r/Homestead

B. *Knowledge Discovery Comparision*

Here we look at these examples in more depth, showing the similarities and differences between the two sets of results.

- *Equality and 100% precision* – Based on the result listed in TABLE III, we can see that Recommenddit, PRAW, and the subreddit sidebar sometimes share the same set of multiple recommendation results.

TABLE III. INPUT: /R/INDIANAUNIVERSITY

Sidebar	Recommenddit	API
/r/bloomington /r/indianapolis /r/indiana	/r/bloomington /r/indianapolis /r/indiana	/r/bloomington /r/indianapolis /r/indiana

- *Inequality and 100% precision* – As shown in TABLE IV and TABLE V, sometimes one list will be a proper subset of another. Looking at TABLE IV, we mostly see subreddits related to games development or to content related to the Unity3D engine. In TABLE V, we see only one result from Recommenddit, but several law-related subreddits from the API, along with /r/VetTech, which is not as clearly related. In TABLE IV, when comparing to the moderator's selections on the sidebar, we achieve a precision of 50% compared to the API's 100%, yet the list we provide clearly provides more substantive overlap. In TABLE V the reverse is true – where we have a precision of 100%, the API has 50% yet still more useful results.

TABLE IV. INPUT: /R/UNITY3D

Sidebar	Recommenddit	API
/r/Unity2D /r/Oculus /r/GameDev /r/Gamedesign /r/Indiegames /r/Unity_tutorials /r/LearnProgramming /r/Blender /r/Devblogs /r/Playmygame /r/UnityAssets	/r/Unity2D /r/gamedev /r/low_poly /r/gamedesign /r/shittyprogramming /r/iOSProgramming /r/oculus /r/IndieGaming /r/indiegames	/r/Unity2D

TABLE V. INPUT: /R/LAWSCHOOL

Sidebar	Recommenddit	API
r/law r/lsathelp r/lsat r/lawschoolscam r/legaled /r/CABarExam /r/lawschooladmissions	/r/law	/r/law /r/lawschoo ladmissions /r/LSAT /r/LawFirm /r/VetTech /r/legal

- *No overlap and 0% precision* – TABLE VI and TABLE VII show more surprising results; in TABLE VI we have 0% precision between Recommenddit and the API, and no related subreddits listed in the sidebar to which we can compare. Our results do largely consist of topics about which one may wish to view lectures, while the API provides just one source of lecture content. In TABLE VII, we do have results from all three methods, but still there is 0% overlap between the API and the sidebar, and only 11% between Recommenddit and the Sidebar, though these last two lists do appear to have more in common – the Reddit API does not give particularly meaningful results here. This goes to illustrate that our method can provide distinct results that augment existing methods for finding subreddits of interest, or even provide recommendations where no existing method produces useful recommendations.

TABLE VI. INPUT: /R/LECTURES

Sidebar	Recommenddit	API
	/r/Economics /r/philosophy /r/physics /r/socialism /r/worldpolitics /r/compsci /r/EndlessWar /r/collapse /r/humor /r/math /r/education /r/Foodforthought /r/wikipedia /r/TrueReddit /r/Anarchism /r/Anticonsumption /r/skeptic /r/Documentaries	/r/TED

TABLE VII. INPUT: /R/HOMESTEAD

Sidebar	Recommenddit	API
/r/permies /r/frugal /r/diy /r/food2 /r/beekeeping /r/green /r/renewableEnergy /r/livestock /r/collapse /r/seedstock /r/financial independence /r/GuerrillaGardening /r/canning /r/mason jars /r/ecoevents /r/poultry /r/cottage_industry	/r/Beekeeping /r/mycology /r/simpleliving /r/Anticonsumption /r/collapse /r/RenewableEnergy /r/SelfSufficiency /r/gardening /r/energy	/r/homeless /r/Dreadlocks

VIII. CONCLUSION & FUTURE WORK

We have shown that association rule mining can be a useful and effective method 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 identifying such associations.

While our process appears sound for our data, there are some limitations that could be overcome in future work. First, by only looking at the largest subreddits, we miss out on the ability to suggest new or small subreddits. Since these are typically the hardest subreddits to discover on one's own, this could be a valuable application of our program, and one that could help smaller communities to grow more quickly. However, since the 2500th subreddit has only 0.01% the user count of one of the top few subreddits, the bottom half of the subreddits by subscriber count will correspond, at most, 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 project to include them. Additional site functionality such as sorting results is possible as well. However, 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, we need to provide a scalable application which can serve millions of concurrent sessions. These limitations are surmountable and this approach therefore appears quite promising for identifying relationships between communities on Reddit and other similar sites.

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.