Improving Discoverability Within Reddit.com Using A Recommender System

Remy Oukaour
Stony Brook University
December 5, 2013

ABSTRACT

We compare a variety of collaborative filtering recommender systems on a public set of voting data for Reddit.com. Our goal is to apply these systems to the novel task of recommending item categories rather than items themselves. This goal is applicable to Reddit for the purpose of making "subreddits" beyond the default set more discoverable. We find descriptive statistics of the data to inform the recommender systems' parameters, discovering that Reddit data frequently follows a power-law distribution. We then evaluate memory-based and model-based recommender systems on both rating prediction and top-N recommendation tasks, and conclude by suggesting how the better-performing systems can be used by Reddit.

1. INTRODUCTION

Reddit.com is a social news website divided into over 275,000 "subreddits" [5]. Users can post links, text, or pictures, comment on the posts, and upvote or downvote them to influence their visibility, but they do so in topic-specific subreddits.

Although Reddit's voting mechanism makes the higher-quality posts in a given subreddit stand out, it can be difficult to find such subreddits in the first place. New users are subscribed to a popular set by default, but these are chosen for their large size and broad appeal, whereas a valuable feature of Reddit is the existence of targeted subreddits for particular interests. To make the less obvious subreddits more discoverable, we evaluate different recommender systems on a subset of Reddit's voting data. The end goal of such a system is to be able to take a user's voting history and compare it with other users' histories to recommend as-yet-unseen subreddits.

In section 2 of this paper, we discuss prior work on recommender systems in general and Reddit-focused systems specifically, and explain how our approach differs from what has already been done. In section 3, we examine statistical properties of the publically available voting data, and process it into a suitable format for cross-referencing users with subreddits, rather than with posts. We discover long-tail distributions in many aspects of the data. In section 4, we describe the recommendation algorithms we use and how we evaluate their performance. In section 5, we discuss our results. We find that even baseline recommender systems which ignore much of the information available to them can significantly outperform a random system, perhaps well enough for real-world use. Finally, in section 6 we conclude by outlining further areas of research and speculating on applications of recommender systems to Reddit.

2. PRIOR WORK

One of the first recommender systems, GroupLens, was created by John Riedl and Paul Resnick in 1994 to filter posts on Usenet. It is the source of the memory-based collaborative filtering algorithm described in section 0 [1]. It was later found that similar item-based algorithms could perform faster and more accurately than user-based ones, given that users generally outnumber items to recommend [2]. Both user- and item-oriented memory-based CF operate directly on the matrix of users versus items, predicting an unknown user–item rating by the ratings other items and users in the same row and column.

Model-based systems, by contrast, attempt to model the factors underlying the distribution of ratings in the user–item table. A useful technique for this purpose is singular vector decomposition, which factors the user–item ratings matrix so as to reduce its dimensionality, and gain as much information as possible from the remaining dimensions.

Emmanuel Benazera, CTO of XPLR Software, demonstrated the company's unsupervised machine learning on this same set of public Reddit data [3]. XPLR used Wikipedia as a corpus for discovering semantic concepts, then clustered 1,800 subreddits according to those concepts. Their recommender is available as a browser extension.

David King, the former Reddit employee who gathered and released the voting dataset, recently implemented a subreddit-based recommender which has been used by Reddit to add a "multireddit" feature [King, personal communication]. Multireddits are collections of related subreddits which can be viewed all at the same time; when viewing one, suggestions appear for other similar subreddits.

Both Benazera's and King's recommender systems are based on subreddit similarity, meaning that they do not provide personalized recommendations. Conversely, each Reddit user's "frontpage" is a personalized list of recommended posts, but this does not help discover new subreddits. A user-based subreddit recommender is currently

lacking, possibly since current recommender systems are focused on recommending items (posts) rather than item categories (subreddits).

3. DATASET

We evaluate the recommendation algorithms on a set of 824,521 affinity scores between 43,976 users and 11,675 subreddits. This is a very sparse dataset, with 0.16% of the possible ratings specified. The affinities range from 0 to 1.

3.1. Acquisition

We downloaded a tab-separated values file, `publicvotes-20101018_votes.dump`, containing 23,091,688 up- and down-votes that Reddit users chose to make available for research. The votes were placed by 43,976 users on 3,436,063 posts in 11,675 subreddits. (This was "almost 17%" of Reddit's total votes in 2011 [4].) We then converted the user–post vote scores into user–subreddit affinity scores.

Given a user u who has voted on n posts in a subreddit s, the affinity of u for s is a Bayesian average of the n votes:

$$A_{us} = \frac{C\bar{A} + n_{\text{up}}}{C + n}$$

Here $\bar{A}$ is the average ratio of upvotes to total votes (approximately 0.85) and C is the median number of votes associating a user to a subreddit (which is 3). We chose to use a Bayesian average because many user–subreddit pairs are associated by only a few votes, which results in raw averages close to the extremes of 0 or 1.

To avoid privacy concerns, the data has been anonymized. Usernames, post titles, and subreddit names have been irreversibly hashed. This means that we are unable to gain particular insights from the data like "users who upvote posts in /r/compsci will probably like /r/tinycode," nor are we able to use features of the users and subreddits in a content-based recommender system. Since this limits us to a collaborative filtering approach, we focus on optimizing these systems to minimize their error metrics based on the known affinities.

3.2. Statistics

We plot quantities against each other—posts and subreddits per user, posts and users per subreddit—and find that, for the most part, they all have long-tail distributions which can be fit by a power law:

$$f(x) = Cx^{-\alpha}$$

$f(x)$ per x	Min.	Max.	Mean	Med.	α
posts/user	1	186,472	525.10	30	−1.057
posts/subreddit	1	3,138,272	1,977.88	5	−1.311
subreddits/user	1	1,041	18.75	8	No fit
users/subreddit	1	36,187	70.62	2	−1.645

Table 1: Statistics for different pairs of quantities.

The counts for the posts voted on per user, posts made per subreddit, and unique users per subreddit all fit a power law distribution well. The exponents are negative, giving them long tails that skew the mean greater than the median.

The counts for subreddits voted in per user, on the other hand, do not fit according to a power law.

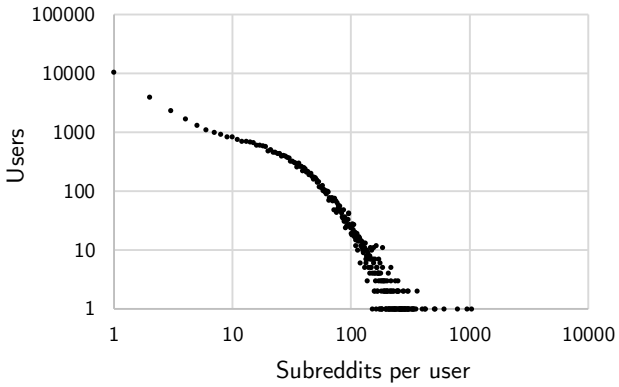

Figure 1: Scatter plot of subreddits per user.

Above more than 20 or so subreddits, the number of users with that many subreddits shrinks more quickly than a power law would expect given the user counts for fewer subreddits. This is due to Reddit's default subreddits system. New users are subscribed to 20 popular subreddits by default, so there are disproportionally many users who participate in subsets of those 20.

We also found that the proportion of upvotes to total votes is consistent: the per-user, per-post, and per-subreddit averages are all 84.24% upvotes. This is close to the average user–subreddit affinity, which is 0.8536.

4. ALGORITHMS

We use two kinds of technique to evaluate the recommendation algorithms' performance. One is how closely they can retrodict known affinities, which we evaluate using the mean absolute error and the root-mean-square error of their residuals.

$$\text{MAE}(p) = \frac{1}{|A|} \sum_{u \in U, s \in S} |p(u,s) - A_{us}|$$

$$\text{RMSE}(p) = \sqrt{\frac{\sum_{u \in U, s \in S} (p(u,s) - A_{us})^2}{|A|}}$$

Here $p(u,s)$ is the retrodicted affinity between user u and subreddit s, and A_{us} is the actual affinity.

The other technique is a top-N recommendations task, which we evaluate using the normalized discounted cumulative gain.

$$\text{nDCG}(p) = \frac{\text{DCG}(p)}{\text{IDCG}(p)}$$

$$\mathrm{DCG}(p) = \sum_{i=1}^{|p|} \frac{2^{A_i} - 1}{\log(i+1)}$$

Here DCG(p) is a measure of the relevance of a permutation p of subreddits to a user u, where p is sorted according to the predicted affinity of u for each subreddit. IDCG(p) is the ideal DCG, calculated using the actual affinities of u. We chose this measure instead of a simpler top-N task where only the top N recommended items are compared with the user's actual top N, because such a model would penalize recommending unknown items more highly than known relevant ones.

For each error metric we perform 10-fold cross-validation on the dataset, using nine-tenths for training and one for testing, and take the average of the metrics over ten tests.

4.1. Baseline

We used a random estimator along with three simple prediction algorithms to establish a baseline error level:

Baseline	$p(u,s)$	MAE	RMSE	nDCG
Random	$\mathcal{U}(0,1)$	0.3967	0.4795	0.9821
Avg. vote	0.8424	0.0989	0.1397	0.9627
User avg. affinity	$\bar{A}_{u\cdot}$	0.0601	0.0961	0.9971
Subreddit avg. affinity	$\bar{A}_{\cdot s}$	0.0908	0.1339	0.9875

Table 2: Baseline error metrics.

Predicting affinities uniformly at random provided a useful upper bound on the error, with an RMSE of 0.4795.

Of the useful but simple algorithms, using the average upvote proportion as a constant prediction performed the worst, with an RMSE of 0.1397. Predicting each user's average affinity for their unknown subreddit affinities performed the best, with an RMSE of 0.0961. Predicting each subreddit's average affinity performed nearly as badly as the single constant prediction.

4.2. Memory-Based CF

We implemented variations on two memory-based collaborative filtering algorithms.

4.2.1. User-Oriented CF (GroupLens)

Used-oriented CF, as designed for GroupLens [1], works by taking a weighted average of other users' standard scores (ratings, centered around the average and normalized by the standard deviation) for the given subreddit, using the users' correlations with the given user as weights.

$$p(a,c) = \bar{A}_{a\cdot} + \frac{\sum_{u \in U} w_{au}(A_{uc} - \bar{A}_{a\cdot})}{\sum_{u \in U} |w_{au}|}$$

Our implementation uses the Pearson correlation coefficient to measure users' similarity.

$$w_{au} = \frac{\sum_{s \in S}(A_{as} - \bar{A}_{a\cdot})(A_{us} - \bar{A}_{u\cdot})}{\sqrt{\sum_{s \in S}(A_{as} - \bar{A}_{a\cdot})^2}\sqrt{\sum_{s \in S}(A_{us} - \bar{A}_{u\cdot})^2}}$$

4.2.2. Item-Oriented CF

Item-oriented CF is the converse of user-oriented CF. It treats subreddits as having an affinity for users, rather than the other way around.

$$p(a,c) = \bar{A}_{\cdot c} + \frac{\sum_{s \in S} w_{cs}(A_{as} - \bar{A}_{\cdot c})}{\sum_{s \in S} |w_{cs}|}$$

It also uses the Pearson correlation coefficient.

$$w_{cs} = \frac{\sum_{u \in U}(A_{uc} - \bar{A}_{\cdot c})(A_{us} - \bar{A}_{\cdot s})}{\sqrt{\sum_{u \in U}(A_{uc} - \bar{A}_{\cdot c})^2}\sqrt{\sum_{u \in U}(A_{us} - \bar{A}_{\cdot s})^2}}$$

4.2.3. Variations

We tested two variations on the CF algorithms above. One is to use cosine similarity instead of Pearson correlation, which for user-oriented CF sums over the subreddits with known affinities for either user, rather than both (and likewise for item-oriented CF). This causes weights to be "dampened" towards zero when there are fewer data points available [6].

We also implemented case amplification as another method of dampening weights. Case amplification applies an exponent to weights:

$$w'_{au} = w_{au} \times |w_{au}|^{\rho - 1}$$

The case amplification power ρ is typically 2.5 [7]. We tested values of ρ from 1 (or no amplification) to 4.

4.3. Model-Based CF

We implemented two model-based collaborative filtering algorithms using singular vector decomposition.

4.3.1. Singular Value Decomposition

Similarity measures like Pearson correlation and cosine similarity are used in memory-based CF because of the impracticality of directly measuring distance in high-dimensional space where data sparsity means that most points will be close in most directions. By reducing the dimensionality, similar techniques can be applied, but with a smaller and more tractable dataset, and a simpler similarity measure (Euclidean distance).

We factor the user–subreddit matrix into three smaller matrices with singular value decomposition [9]:

$$\boldsymbol{M}_{u \times s} = \boldsymbol{U}_{u \times u} \boldsymbol{\Sigma}_{u \times s} \boldsymbol{V}^*_{s \times s}$$

We then replace all but the first k singular values in $\boldsymbol{\Sigma}$ with zeros, approximating the original matrix while reducing the dimensionality. We test values for k from 1 to 500.

Affinities can be predicted by reversing the decomposition, but limiting the reversal to only the row and column relevant to the user–subreddit pair being predicted.

$$p(u,s) = \bar{A}_{\cdot s} + \left(\boldsymbol{U}\sqrt{\boldsymbol{\Sigma}}\right)_u \cdot \left(\sqrt{\boldsymbol{\Sigma}}\boldsymbol{V}\right)_s$$

Here $p(u,s)$ is the predicted affinity between user u and subreddit s, and $\bar{A}_{\cdot s}$ is the average affinity for subreddit s.

The matrices $\boldsymbol{U}\sqrt{\boldsymbol{\Sigma}}$ and $\sqrt{\boldsymbol{\Sigma}}\boldsymbol{V}$ can be precalculated and stored for a small value of k.

4.3.2. PureSVD

The authors of [8] designed a recommender system which uses SVD that is more suited for top-N tasks than for direct rating prediction. Given a user u and a subreddit s, it computes an association score between u and s:

$$\hat{p}(u,s) = \boldsymbol{M}_u \cdot \boldsymbol{V} \cdot \boldsymbol{V}_s^*$$

Here $\hat{p}(u,s)$ does not actually make sense as a predicted affinity between u and s; however, it does maintain the relative ordering of subreddits for a given user.

4.4. Graph-Based CF

We considered modeling the users and subreddits as a graph, instead of a matrix, to see if graph-based algorithms can be effective at predicting ratings. These techniques were abandoned due to their similarity to the model-based algorithms already implemented.

4.4.1. Personalized PageRank

We attempted to model the set of users and subreddits as a bipartite graph, with affinities being weighted undirected edges between them. In this scenario, the personalized PageRank scores from a given source node to any other node could be a good proxy for the similarity or distance between the two nodes. However, PageRank scores are equivalent to entries of the principle eigenvector of the adjacency matrix, and eigenvectors are only defined for square matrices; the user–subreddit matrix is non-square. A similar concept to eigenvectors is defined for non-square matrices: singular vectors. Since we already use these in two systems, we chose not to derive essentially the same algorithms by a different route.

4.5. Hybrid CF

The weighted average of predictions from a variety of models can be more accurate than the prediction of any one model [7]. We did not attempt to combine memory-based and model-based predictions, since any gain in accuracy would be offset by the cost of requiring two separate recommender systems to generate recommendations. However, given the reasonably good performance of simply taking user or subreddit average affinities as predictions, we decided to create a hybrid model from the two quantities.

$$p(u,s) = q\bar{A}_{u\cdot} + (1-q)\bar{A}_{\cdot s}$$

Here q is the weight given to the average affinity of user u, and $(1-q)$ is the weight given to the average affinity of subreddit s. We tested values of q from 0.1 to 0.9.

5. RESULTS

We conducted experiments using programs written in C++, as well as a Python script to perform SVD using the SciPy library. All three error metrics were generally strongly correlated, so the graphs below do not show them all.

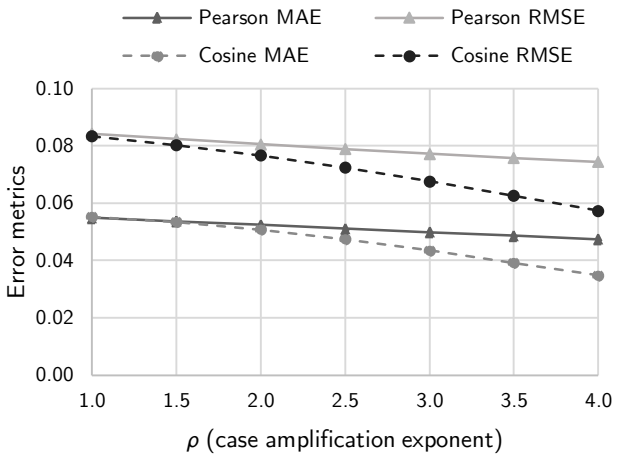

Figure 2: Error metrics for user-based CF.

Scores for both varieties of user-based CF, using Pearson correlation and cosine similarity, improved as ρ was raised. The cosine similarity metric also outperformed Pearson correlation, with the gap widening as ρ increased. The worst-performing variety, using Pearson correlation without case amplification, had an RMSE of 0.0843; the best, using cosine similarity with $\rho = 4$, had an RMSE of 0.0574.

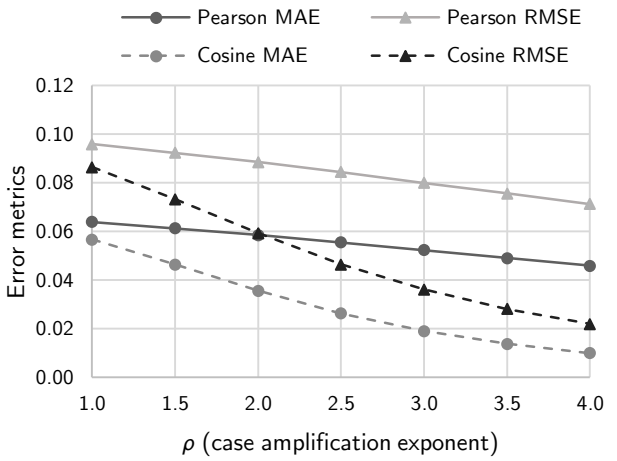

Figure 3: Error metrics for subreddit-based CF.

The same patterns hold for subreddit-based CF as for user-based CF: cosine similarity outperforms Pearson correlation and higher values of ρ improve accuracy. It is more erratic than user-based CF; the worst case had an RMSE of 0.0958, higher than any user-based variation, but the best had an RMSE of 0.0219, lower than any user-based one. This is in keeping with the fact that there are four times

as many users as subreddits, so each subreddit can potentially have affinities with many users and be a reliable indicator for predicting other affinities.

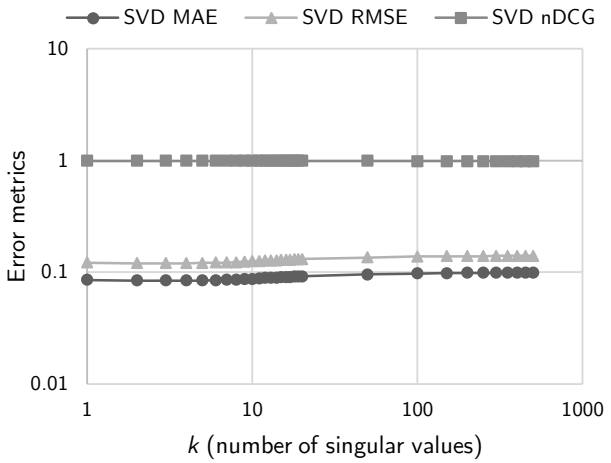

Figure 4: Error metrics for SVD (log-log plot).

The results for the SVD recommender system are confusing. We were surprised that the value of k which resulted in the most accurate predictions was 3 (with an RMSE of 0.1199 and an nDCG of 0.9930, on par with the user- or subreddit-average baseline systems). This seems to contradict the intuition that lesser singular values will still provide an increase in accuracy, albeit a small one. There was in fact very little variation in the error metrics despite testing k from 1 to 500; the greatest RMSE is 0.1395.

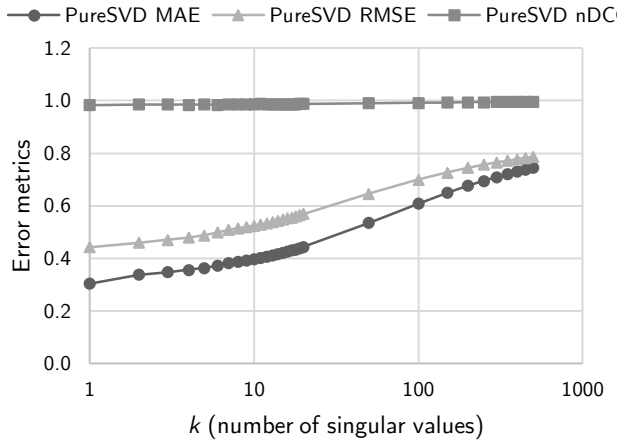

Figure 5: Error metrics for PureSVD (log plot).

PureSVD shows a strong increase in MAE and RMSE metrics as more singular values are taken into account, with the RMSE reaching 0.7867, close to the theoretical worst possible (assuming predicted affinities of 0 or 1, whichever is further from the actual). In fact this is expected, since PureSVD scores are not strictly predicted affinities and can go beyond their range. Like the other SVD system, though, the nDCG values do not vary much at all despite the large range of k. The best accuracy is achieved when $k = 4$, for which the nDCG is 0.9849.

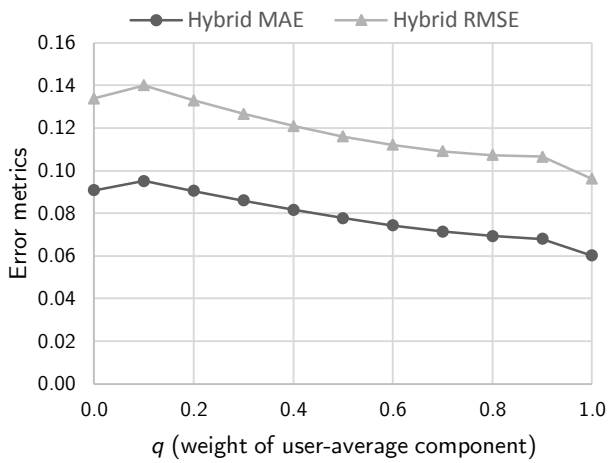

Figure 6: Error metrics for hybrid system.

The results of the hybrid system do not in fact improve on the two systems hybridized to create it. As q (the contribution of the user-average system) varies from 0 to 1, with the subreddit-average system likewise varying from 1 to 0, there is roughly a linear interpolation between the two systems' own error metric results. The RMSE for the hybrid system ranges from 0.1400 to 0.0961.

System	RMSE
Random	0.4795
Average vote	0.1397
User average affinity	0.0961
Subreddit average affinity	0.1339
User-based Pearson CF ($\rho = 1$)	0.0842
User-based Pearson CF ($\rho = 4$)	0.0744
User-based cosine CF ($\rho = 1$)	0.0833
User-based cosine CF ($\rho = 4$)	0.0573
Subreddit-based Pearson CF ($\rho = 1$)	0.0958
Subreddit-based Pearson CF ($\rho = 4$)	0.0714
Subreddit-based cosine CF ($\rho = 1$)	0.0864
Subreddit-based cosine CF ($\rho = 4$)	*0.0219*
SVD ($k = 3$)	0.1199
PureSVD ($k = 4$)	0.4794
User-subreddit hybrid ($q = 0.5$)	0.1160

Table 3: Best RMSE scores for all systems.

Overall subreddit-based collaborative filtering, using cosine similarity and a case amplification power of 4, achieved the most accurate predictions, with an average RMSE of 0.0219 over ten cross-validation trials.

6. CONCLUSIONS

In this paper, we compare the performance of different collaborative filtering recommender systems on a dataset of

anonymized, public Reddit votes. We focus on the under-addressed problem of recommending categories to users instead of single items, achieving this by using the ratings given to categorized items (that is, votes on posts in subreddits) to derive an overall affinity score between the user and the category. This is a general technique which could apply to movie genres on Netflix, product categories on Amazon or eBay, or communities like Twitter or Tumblr where user-created categories emerge from the use of free-form hashtags or labels.

With regard to the particular case of increasing subreddit discoverability, Reddit seems to be aware of this problem, given their recent introduction of recommendations for multireddits. This feature could be further enhanced by adding personalization based on each user's voting history. As this paper shows, even a simple system based on the average votes of users or subreddits can improve on a complete lack of personalization. If computing power is a concern for running a more complex and more accurate recommender system, its use could be limited to users who donate to the site. We hope that Reddit will continue to improve its content-discovery features.

7. REFERENCES

[1] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. 1994. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews." In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work.*

[2] G. Karypis. 2001. "Evaluation of Item-Based Top-N Recommendation Algorithms." In *CIKM '01 Proceedings of the tenth international conference on Information and knowledge management.*

[3] E. Benazera. "A SubReddit recommender with XPLR." (February 12, 2013.) Retrieved November 6, 2013 from https://xplr.com/a-subreddit-recommender-with-xplr/.

[4] TedFromTheFuture. "Attempt #2: Want to help reddit build a recommender? -- A public dump of voting data that our users have donated for research." (October 25, 2011.) Retrieved November 6, 2013 from http://www.reddit.com/r/redditdev/comments/lowwf/attempt_2_want_to_help_reddit_build_a_recommender/.

[5] T. Misera. "metareddit." (November 6, 2013). Retrieved November 6, 2013 from http://metareddit.com/.

[6] M. Ekstrand. "Similarity Functions for User-User Collaborative Filtering." (October 24, 2013.) Retrieved November 6, 2013 from http://www.grouplens.org/similarity-functions-for-user-user-collaborative-filtering/.

[7] X. Su and T. M. Khoshgoftaar. 2009. "A Survey of Collaborative Filtering Techniques." In *Advances in Artificial Intelligence.*

[8] P. Cremonesi, Y. Koren, and R. Turrin. 2010. "Performance of Recommender Algorithms on Top-N Recommendation Tasks." In *RecSys '10 Proceedings of the fourth ACM Conference on Recommender Systems.*

[9] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. 2000. "Application of Dimensionality Reduction in Recommender System — A Case Study." In *ACM WEBKDD Workshop.*