Subrecommendit: Recommendation Systems on a Large-Scale Bipartite Graph

Yunhe (John) Wang, Alexis Goh Weiying, David Xue
{yunhe,gweiying,dxue}@stanford.edu

Abstract— **With the massive amount of content on social media platforms today, personalized recommendation systems are crucial for identifying relevant content to continuously engage users with. In this paper, we compare various graphical approaches, both classic and recent, for large-scale bipartite graphs by testing their performance on subreddit recommendations for users on Reddit. We also investigate community detection as a potential tool for recommendation. We show that through taking into account user-specific preferences, Collaborative Filtering and Mixed Similarity Diffusion performed the best on standard recommendation metrics, and the Random Walk approach ran the fastest while performing better than recommending the top subreddits. Our community detection approach reveals both intuitive and non-intuitive relationships between subreddits in communities up to a certain size, shows stable communities of subreddits across time, and offers direction for future recommendation systems.**

I. Introduction

Reddit, often called "the front page of the Internet", is an online community where users share and discuss their topics of interest. These entries are organized by areas of interest called 'subreddits", which serve as subcommunities within the overall Reddit community. A user may post new content to individual subreddits (termed "submissions"), and may also participate in the community by upvoting, downvoting, and commenting on other users' submissions and comments.

In this paper, we implement various graphical recommendation approaches and compare their performance on generating subreddit recommendations. Recommendation algorithms on large-scale bipartite graphs is a highly relevant problem as personalized recommendations are crucial for user engagement on social media platforms. Recommending relevant subreddits is highly challenging considering the volume and frequency of content posted on Reddit - in November 2018 alone users posted over 14 million submissions and 119 million comments [4].

To tackle this question, we construct the user-subreddit bipartite graph on Reddit data. Undirected edges between user and subreddit nodes represent a user commenting on a subreddit. Edges can be unweighted, or weighted by the number of comments a user makes on the subreddit. We use Reddit data generated over five months from January to May 2018 and a heldout dataset for June 2018.

We investigate three different approaches for recommendations on large-scale bipartite graphs:

1) Collaborative Filtering
2) Resource Diffusions
3) Random Walk

As the above algorithms have never previously been applied to the user-subreddit graph, we contribute performance findings. We show that by taking into account user-specific preferences, Collaborative Filtering and Mixed Similarity Diffusion perform the best on 3 standard recommendation metrics, and the Random Walk approach ran the fastest while still performing noticeably better than our baseline of recommending the most popular subreddits.

Further, we investigate the recommendation task from the perspective of *community detection*. Intuitively, community structure on a projected unipartite subreddit graph can give us insight into "clusters" of similar subreddits and form the basis for subreddit recommendations. We generate the folded one-mode subreddit graph, where edges between subreddits represent that a user that commented on both subreddits, and use the state-of-the art Leiden Algorithm [18], an improvement over the Louvain Algorithm, for detecting communities of subreddits. We apply an extension of modularity to address the *resolution-limit* problem, showing that community detection reveals related subreddits at different size scales of communities. We hypothesize and validate that clusters of subreddits remain stable over time, i.e. new edges between subreddits should appear in the same communitiy clusters. This suggests that communities can offer valuable information for community-based recommendation systems and offers direction for future research.

II. Related Work

There are several areas of investigation on the user-subreddit bipartite network. Below we review the literature on recommendation systems for bipartite graphs.

A. Collaborative Filtering

Collaborative filtering techniques are common within the recommendation system space. For example, York et al. [19] employed such techniques to recommend products on Amazon, and Resnick et al. [20] on News. We base our algorithm on Deshpande et al.'s [21] item-item collaborative filtering technique, which they demonstrate to be effective on 8 real datasets.

B. Resource Diffusion

Resource diffusion is a popular field of recommendation algorithms for bipartite graph networks, first studied by Zhou et al in 2007 [5].

Consider item nodes, m and n which are not directly connected. Resource diffusion describes the two-step process where item m sends resources to n through their common users. In the first step, item nodes distribute resources amongst its users equally based on the items' degrees. In the second step, item nodes recover resources from the users based on the users' degrees. This process of resource diffusion allows resources to be distributed from items each user has collected (subreddits that they have commented on), to items that share common users with them (subreddits that they may be keen on).

In its simplest form, recommendations are only generated with implicit feedback where edges between users and items are unweighted. Wang et al. proposes a method to utilize information from explicit feedback, the weight of the edges, in the mass diffusion process [1]. The method, known as Mixed Similarity Diffusion, captures richer information from the bipartite graphs as it accounts for users' ratings on items when diffusing resources. They demonstrate competitive results against other recommendation techniques on the MovieLens dataset.

In this paper, we investigate the performance of both the original Mass Diffusion and the Mixed Similarity Diffusion algorithms on generating recommendations for the Reddit bipartite graph.

C. Random Walk

Another approach to graphical recommendation systems involves random walks with restarts. In this approach inspired by the PageRank algorithm [9], we simulate a user who begins at a random node in a starting set of nodes S, and at each step randomly traverses to a node adjacent to their current node. In addition, at each step, the user may teleport to a random node in S instead of moving to an adjacent node (a "restart"). This way, nodes closer to the starting set S are visited more often.

Pixie [15] uses such an algorithm to recommend new content (termed "pins") to users of Pinterest. In order to do so, Pixie simulates multiple random walks on a bipartite graph of pins and "boards" (collections of pins), where the starting set S is a set of pins that a user has interacted with. On each walk, Pixie collects counts on how many times a pin is visited, and aggregates the counts at the end in order to recommend new pins. The authors demonstrate that through biasing the walk using user preferences and various other optimizations, Pixie achieves higher user engagement than previous Pinterest recommendation systems while being capable of recommending pins in real-time.

In this paper, we will extend the random walk recommendation system to the Reddit dataset, and compare it against other recommendation systems.

D. Community Detection on Bipartite Graphs

Community detection is a well studied problem for unipartite graphs. Since it was proposed in 2008, the greedy Louvain algorithm [16] has been found to be one of the fastest and best performing algorithms. However, the treatment of the problem on bipartite networks has been sparse. Because edges connect vertices of two different types, the classical definition of communities does *not* directly apply.

Most bipartite community detection efforts have extended modularity [12], the classical community quality metric, to bipartite networks. In 2007, Barber [6] developed a modularity-based algorithm called Bipartite Recursively Induced Modules (BRIM). BRIM is an iterative algorithm that employs a refined modularity matrix to accommodate for the bipartite structure. In 2009, Liu and Murata [7] proposed a hybrid algorithm called LPBRIM that uses the Label Propagation heuristic to search for a best community configuration, and subsequently uses BRIM to refine the results. A pitfall of most BRIM-based approaches, as acknowledged by Barber, is that it only handles unweighted and undirected bipartite networks. Like unipartite modularity, maximizing bipartite modularity is an NP-hard problem [11]. Therefore, there is no guarantee to achieve the best possible modularity which makes it difficult to create or find an algorithm that performs well on any network.

Projection-based approaches, where a bipartite network is projected to a unipartite network, have historically been used in recommendation systems. A key idea is the emphasis on one of the two node sets called the *primary set*. These sets can be switched for different applications. The primary strength of projection approach are that they allow us to investigate bipartite networks using powerful one mode algorithms. Empirically, Guimera et al. [10] have found no difference in the node communities detected in P whether they resulted from modularity maximization after projection, or projection after bipartite modularity maximization. However, some papers have found sometimes the project resulted in loss of the bipartite structural information [5], [14].

In 2018, Traag et al. [18] proposed the Leiden algorithm which they found to be faster than the Louvain algorithm while yielding communities with proven guarantees to be connected. Furthermore, this work has incorporated recent work to extend the traditional quality function of modularity to address the *resolution limit*. Modularity optimization algorithms are subject to a *resolution limit* in that the maximum modularity community-wise partition can fail to resolve communities, causing smaller communities to be clustered into larger communities.

In this paper, we investigate the Leiden algorithm [18] for community detection on the folded subreddit graph.

III. Data

Reddit post and comment data is publicly available [4]. Each submission has information such as subreddit name, user, submission content, and more. Each comment contains attributes on subreddit name, text, upvote score, user, and date. Each user contains information such as account creation time, comment ids, last activity, and more. We examined a subset of subreddits and users over the first six months of 2018 from January to June.

During this entire 6 month period, 9,731,646 users commented on 162,242 subreddits. The number of unique comment edges was 68,138,004 and on average each user com-

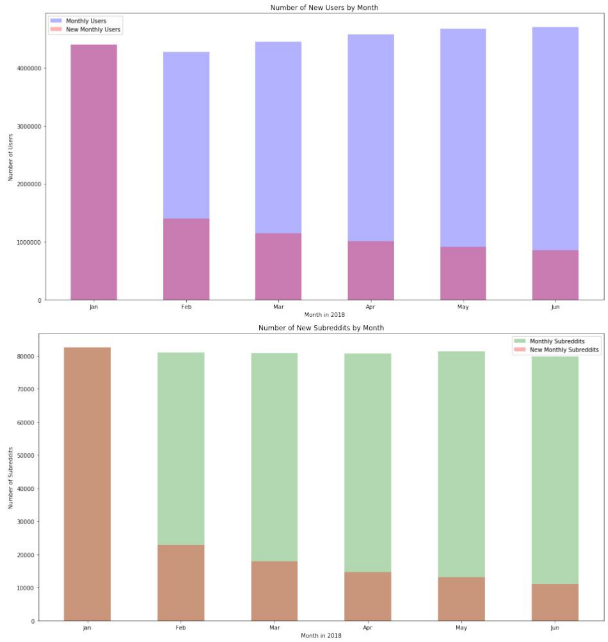

Fig. 1. New users (top) and subreddits (bottom) out of total monthly users and subreddits from January to June 2018. The proportion of previously unseen users and subreddits (i.e. new nodes in the graph) begins to level off to a low value by May and June. This suggests that the graph node structure begins to stabilize after a few months.

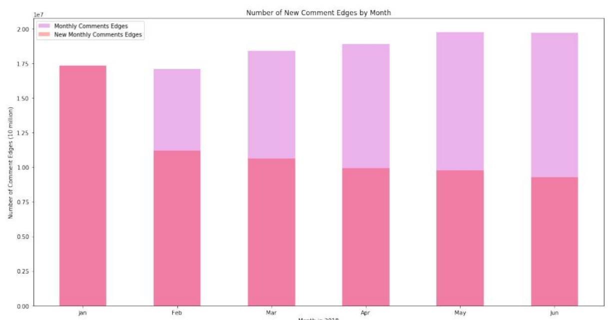

Fig. 2. New edges between users and subreddits out of total monthly edges from January to June 2018. The proportion of new edges in the graph remains fairly high even by May or June.

mented on 6.63 unique subreddits and made 49.4 comments. Figure 1 illustrates how the graph nodes structure (i.e. users and subreddits) stabilize by May or June while Figure 2 shows how the number of new graph edges (i.e. comments on new subreddits by a user) remains fairly robust into later months.

A. Preprocessing

We evaluate how well various recommendation systems can predict user subreddit behavior by feeding the systems “Historical Behavior” and seeing how well they predict “New Behavior”. Historical Behavior refers to the number of times each user commented on each subreddit from from January 2018 to May 2018, and New Behavior refers to which subreddits a user commented on in June 2018 that they did not comment on between January and May 2018.

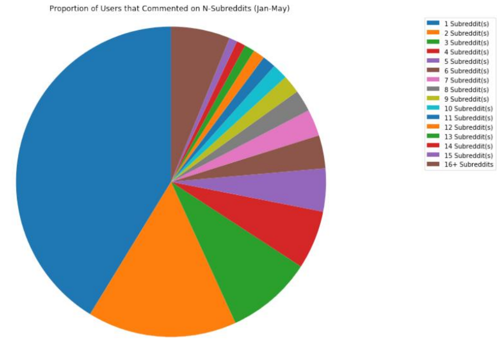

Fig. 3. Proportion of users who commented on N-subreddits from January to June 2018. 39.45 percent of users only commented once ($N \leq 1$) during this time period, 54.34 percent commented on one or two ($N \leq 2$), and 62.93 percent of users commented on three or fewer ($N \leq 3$).

In order to model Historical Behavior, we build a user-subreddit bipartite graph in which an edge is drawn from each user to a subreddit the user commented on, weighted by the number of comments the user made to the subreddit. This results in a graph with 8,876,403 users and 151,144 subreddits, which is computationally intractable given our available resources. As Figure 3 demonstrates, a majority of users commented on just one or two subreddits over this time period. Users who commented on one subreddit do not connect subreddits in the graph and thus do not contribute to our graph based recommendation systems, and users who commented on two contribute very little. At the same time, making recommendations for these users with very little information is known as the *cold start problem* and is beyond the scope of our project, so we filter them out.

In addition, 151,144 subreddits is intractable given our resources. As Figure 5 demonstrates, the vast majority of subreddits have very few unique users commenting on them - about 90,000 were commented on by at most 9 users from Jan to May 2018. If we were to filter these subreddits out, however, we’d be unable to recommend these subreddits to new users. Figure 6 demonstrates the impact of such a filter - all the subreddits commented on by at most 9 users from Jan to May 2018 cumulatively gained about 50,000 new users in June 2018 - this means for these 50,000 users we would be unable to recommend one of the correct subreddits. This is insignificant, however, since if we were to add up the new users gained by all subreddits, we’d obtain 7 million. The intuition is that we have little data on these new or unpopular subreddits. For these reasons, filtering out these subreddits is efficient yet sacrifices minimal accuracy. After applying both the user and subreddit filters, we obtain a graph with 4,052,716 users and 54,204 subreddits. The node degree distribution of our filtered graph is is shown in Figure 4.

For the month of June, we found that 2,812,982 of the users from Jan to May made at least 1 comment. This is far too many users to feasibly evaluate on, and as the average user made comments in 2.42 new subreddits, also includes users with insufficient data to effectively evaluate on. In addition, there are users who made comments in over a

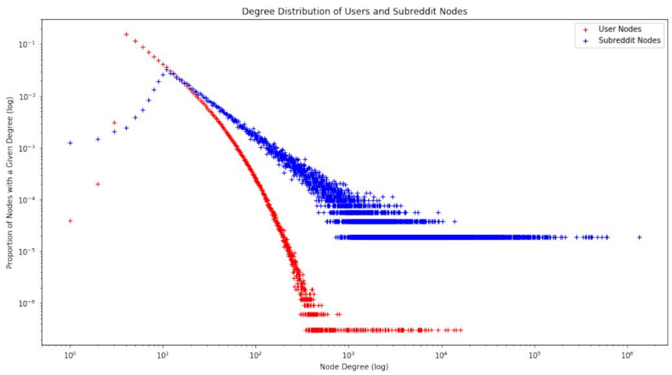

Fig. 4. The final node degree distribution for users (blue) and subreddits (red) after filtering out users and subreddits. Note the left-side has trailing values due to our thresholding choices.

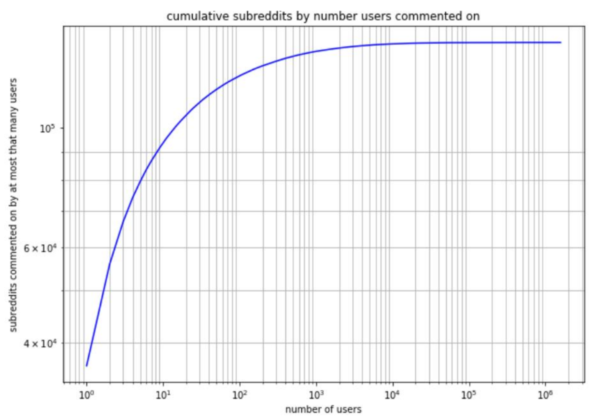

Fig. 5. Cumulative subreddits by the number of users who commented on them. Data is from Jan to May 2018 (Historical Behavior). Let (u,s) be a point on the curve in the graph. This point represents that if we were to count up all the subreddits commented on by at most u users, we'd have s subreddits.

thousand new subreddits, such as "CommonMisspellingBot" - these are likely to be bots.

In order to have meaningful evaluations, we generated our test set by randomly sampling 100 selected users out of the 118,620 users who commented on between 10 to 100 subreddits.

IV. Methods

We provide a brief theoretical outline of each approach for recommendations.

A. Baseline: Popularity

For our baseline, we rank all the subreddits by number of users. For each user u in our test set, we recommend the top n subreddits with the most users, excluding the ones u has already commented on from January to May.

B. Item-Item Collaborative Filtering

We use a method based on Deshpande et al.'s [21] item-item collaborative filtering technique. Let s_1 and s_2 be subreddits, and define a similarity metric $S(s,s_2)$ that is greater if s_1 and s_2 are considered more similar to eachother. While Deshpande et al used Cosine Similarity and Conditional Probability-Based Similarity, we use Jaccard Similarity, which is independent of edge weights. This is given by:

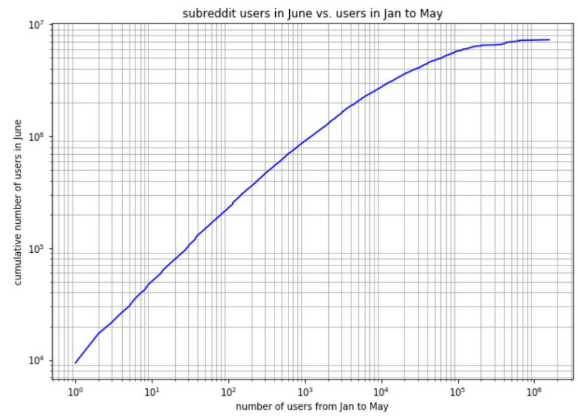

Fig. 6. Comparing user comments in June vs. user comments in the same subreddits from Jan to May. Let (u,v) be a point on the curve in the graph. This point represents that if we take all the subreddits that were commented on by at most u users from Jan to May 2018, and sum up the gain in new users who commented on the same Reddits in June 2018, we'd get v.

$$S(s_1,s_2) = \frac{|\text{unique users who commented in } s_1 \bigcap s_2|}{|\text{unique users who commented in } s_1 \bigcup s_2|}$$

Next, let $N(s_1;k)$ be the k-nearest neighbor subreddits to subreddit s_1 as defined by similarity metric $S(s_1,s_2)$, and then given a query user u and set of subreddits S_u that u commented on in the Input Graph, we score a subreddit s using the following:

$$\texttt{Score}(s) = \sum_{s_1 \in S_u} \sum_{s_2 \in N(s_1;k)} S(s_1,s_2)$$

Finally, we recommend the top n subreddits by highest score.

C. Resource Diffusion: Original Mass Diffusion

In this section, we use Greek letters for subreddits and Latin letters for users for ease of readability in line with [1]. For user i, subreddit α, the adjacency matrix $A_{i\alpha}$ is given by:

$$A_{i\alpha} = \begin{cases} 0, & \text{if user } i \text{ comments on reddit } \alpha \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

and the degrees for user i and subreddit α are k_i and k_α respectively.

The two-step process of Mass Diffusion is as follows:

Step 1: For target user i, we distribute resources from subreddits that i has participated in to other users j based on the subreddit degrees:

$$f'_{ij} = \sum_{\alpha=1}^{n} \frac{A_{i\alpha} A_{j\alpha}}{k_\alpha}$$

Step 2: For target user i, their resources on item β is recovered by:

$$f'_{i\beta} = \sum_{j=1}^{m} \frac{A_{j\beta}}{k_j} f'_{ij}$$

The recommendation list for target user i is obtained by ranking the final resource vector; the subreddits that have recovered the most resources are the recommended subreddits.

D. Resource Diffusion: Mixed Similarity Diffusion

Based on the Mixed Similarity Diffusion introduced by Wong. [1], we extend mass diffusion by utilizing the number of comments made by users on subreddits as our explicit feedback. In the first step, the resource distribution to users is weighted by the similarity between the target user i and other users j. We utilize the cosine similarity, where the similarity between user i and j is given by:

$$Cos(i,j) = \frac{\sum_{\alpha=1}^{n'} R_{i\alpha} Rj\alpha}{\sqrt{\sum_{\alpha=1}^{n'} R_{i\alpha}^2}\sqrt{\sum_{\alpha=1}^{n'} R_{j\alpha}^2}}$$

where $R_{i\alpha}$ is the number of comments user i makes on subreddit α.

The two-step process of Mixed Similarity Diffusion is then as follows:

Step 1: For target user i, we weight the initial distribution of resources from subreddits that i has participated in to other users j by their cosine similarities:

$$f'_{ij} = \sum_{\alpha=1}^{n} \frac{A_{i\alpha} A_{j\alpha} Cos(i,j)}{\sum_{k=1}^{m} A_{k_\alpha} Cos(i,k)}$$

Step 2: For target user i, their resources on item β is recovered by:

$$f'_{i\beta} = \sum_{j=1}^{m} \frac{A_{j\beta}}{k_\beta^{\lambda} k_j^{1-\lambda}} f'_{ij}$$

where λ is introduced as an additional hyperparameter between 0 and 1 to weigh for the relative importance of the the users' degree and the subreddits' degree in this step.

Subreddits are ranked by amount of resources recovered.

E. Random Walk

We implement a basic version of the Random Walk Recommendation System as shown in Algorithm 1. In brief, given a user u, the `Scores` function returns a vector of scores, one for each subreddit, and we recommend the top n subreddits by highest score that u has not already commented on in Jan through May 2018. In order to calculate the scores, we iterate through all subreddit neighbors s of u, and perform random walks with restarts for N_s total steps, with the subreddit s being the only node in the starting set. The length of each random walk is sampled from the geometric distribution with parameter α, a distribution inspired by PageRank[9] in which a user traversing the graph will have probability α of teleporting at each node. During the random walks for subreddit neighbor s, we record the number of times we visit each subreddit in the vector $scores_s$, which is subsequently aggregated into the vector $scores$.

We use Multi-Hit Boosting as introduced in Pixie[15] in order to aggregate subreddit neighbor s score vector $scores_s$ into the final score vector $scores$; this weighs the scores so that subreddits visited multiple times from different subreddit neighbors are weighted higher than ones visited multiple times from the same subreddit neighbor. Pixie[15] also uses various other techniques, including scaling N_s based on the degree of subreddit neighbor s and biasing the random walk using additional user preferences. We found the former to be ineffective on our graph, while the latter difficult due to lack of more data on Reddit user preferences in our graph. The basic random walk serves as a good baseline for the potential of the algorithm, and we comment on advantages and extensions in the Results section.

Algorithm 1 Random Walk Algorithm

procedure SCORES(User u, Graph G, Real α, Int N)
  $scores \leftarrow \vec{0}$
  $N_s \leftarrow N/|\texttt{Neighbors}(u,G)|$
  for all $s \in \texttt{Neighbors}(u,G)$ **do**
    $scores_s \leftarrow \texttt{RandomWalk}(s,G,\alpha,N_s)$
    $scores \leftarrow scores + \sqrt{scores_s}$
  return $scores^2$
procedure RANDOMWALK(Subreddit s, Graph G, Real α, Int N_s)
  $totalSteps \leftarrow 0$
  $scores_s \leftarrow \vec{0}$
  while $totalSteps < N_s$ **do**
    $curSubred \leftarrow s$
    $walkLength \leftarrow \texttt{GeometricDist}(\alpha)$
    for i **from** 1 to $walkLength$ **do**
      $curUser \leftarrow \texttt{RndNeighbor}(curSubred, G)$
      $curSubred \leftarrow \texttt{RndNeighbor}(curUser, G)$
      $scores_s[curSubred] \leftarrow scores_s[curSubred] + 1$
    $totalSteps \leftarrow totalSteps + walkLength$
  return $scores_s$

F. Community Detection on the Folded Subreddit Graph with the Leiden Algorithm

The Leiden algorithm [12] for community detection is similar to the Louvain algorithm in many respects. The Leiden algorithm consists of three phases: (1) local moving of nodes, (2) refinement of the partition and (3) aggregation of the network based on the refined partition, using the non-refined partition to create an initial partition for the aggregate network. We outline two of the key stages similar to the Louvain algorithm while offering key refinements: *optimization* and *aggregation*.

Phase 1: Local Node Optimization: We start by initializing a queue with all nodes in the network. The nodes are added to the queue in a random order. We then remove the first node from the front of the queue and we determine whether the quality function can be increased by moving this node from its current community to a different one. If we move the node to a different community, we add to the rear of the queue all neighbours of the node that do not belong to the node's new community and that are not yet in the queue. We keep removing nodes from the front of the queue, possibly moving these nodes to a different community. This continues until the queue is empty. After all nodes have been visited once, Leiden visits only nodes whose neighbourhood has changed, whereas Louvain keeps visiting all nodes in the network. The pseudocode is shown in Algorithm 2.

Algorithm 2 Leiden Phase 1: Local Node Optimization

procedure MOVENODESFAST(Graph G, Partition $\mathcal{P}$)
    $Q \leftarrow \texttt{QUEUE}(V(G))$
    while $Q \neq \emptyset$ **do** ▷ Continue until no more nodes.
        $v \leftarrow Q$.remove()
        $C' \leftarrow \arg\max_{C \in \mathcal{P} \cup \emptyset} \Delta\mathcal{H}_{\mathcal{P}}(v \to C)$
        if $\Delta\mathcal{H}_{\mathcal{P}}(v \to C') > 0$ **then**
            $v \to C'$
            $N \leftarrow \{u | (u,v) \in E(G), u \notin C'\}$
            Q.add$(N - Q)$
    return $\mathcal{P}$

Algorithm 3 Leiden Phase 2: Aggregation (Refined)

procedure AGGREGATEGRAPH(Graph G, Partition $\mathcal{P}$)
    $\mathcal{P}_{ref} \leftarrow \texttt{RefinePartition}(G, \mathcal{P})$
    $V \leftarrow \mathcal{P}_{ref}$
    $E \leftarrow \{(C,D) | (u,v) \in E(G), u \in C \in \mathcal{P}_{ref}, v \in D \in \mathcal{P}_{ref}\}$
    return $\texttt{Graph}(V,E)$
procedure REFINEPARTITION(Graph G, Partition $\mathcal{P}$)
    $\mathcal{P}_{ref} \leftarrow \texttt{SingletonPartition}(G)$
    for $C \in \mathcal{P}$ **do**
        $\mathcal{P}_{ref} \leftarrow \texttt{MergeNodesSubset}(G, \mathcal{P}_{ref}, C)$
    return $\mathcal{P}_{ref}$

Phase 2: Aggregation with Refinement: Aggregation is almost identical to the Louvain algorithm with a key difference. In the refinement phase, nodes are not necessarily greedily merged with the community that yields the largest increase in the quality function. Instead, a node may be merged with any community for which the quality function increases. The pseudocode is shown in Algorithm 3.

Lastly, note the Louvain and Leidag algorithms can be optimized for any quality function. The vanilla modularity quality function [12] is

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(\sigma_i, \sigma_j)$$

where A is the adjacency matrix, k_i is the (weighted) degree of node i, m is the total number of edges (or total edge weight), $sigma_i$ denotes the community of node i and $\delta(\sigma_i, \sigma_j) = 1$ if $\sigma_i = \sigma_j$ and 0 otherwise.

Another quality function is Reichardt and Bornholdt's Potts (RBP) model [22] which introduces a γ linear resolution parameter term to modularity

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \gamma \frac{k_i k_j}{2m} \right) \delta(\sigma_i, \sigma_j)$$

Note that this is identical to vanilla modularity when $\gamma = 1$.

V. METRICS FOR RECOMMENDATION

To present a comprehensive evaluation of the recommendation systems investigated, we utilize a number of well-known recommendation metrics. Given a user u, we define "recommended subreddits" as the ranked list of subreddits our algorithm recommends for user u, and "relevant subreddits" as the target list of subreddits, that is, the subreddits that u commented on in the month of June for the first time.

A. Precision@n

Precision@n is an important evaluation metric for ranking predictions in recommendation systems. Given a list of n recommended subreddits, *Precision@n* is the fraction of relevant subreddits, formally:

$$Precision@n = \frac{|\{\text{relevant subreddit}\} \cap \{\text{retrieved subreddit}\}|}{|\{\text{retrieved subreddit}\}|}$$

We will be using $n = 10$ to evaluate our recommendation systems.

B. Mean Reciprocal Rank

While *Precision@n* gives a good measure of how many recommended subreddits are relevant, it does not take into consideration the order in which we rank the recommended subreddits. *MRR*, or Mean Reciprocal Rank, addresses this by utilizing the reciprocal of the rank of the first relevant subreddit in the recommendation list, averaged across all users, defined as:

$$MRR = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\text{rank}_u}$$

where U is the set of users and rank_u is the rank of the first relevant recommendation for user u.

C. Mean Average Precision

MAP, or Mean Average Precision, also takes into consideration the order in which we rank the recommended subreddits. However, while *MRR* only considers the rank of the first relevant result per user, *MAP* considers the rank of all the relevant results within the list of recommended subreddits. The use case determines which metric is more useful.

MAP is calculated by averaging the *Precision@n* values for all relevant subreddits in the recommendation list per user, and then averaging this value over all users, defined as:

$$MAP = \frac{1}{|U|} \sum_{u \in |U|} \frac{1}{|S_u|} \sum_{s \in S_u} Precision@\text{rank}_{su}$$

where S_u is the set of relevant subreddits for user u and rank_{su} is the rank of subreddit s in the recommendation list for user u.

VI. Results

A. Quantitative Results

We generate a recommendation list for each user in our test set of 100 users, using each of our algorithms, and evaluate them using the 3 metrics, *Precision*@10, *MRR* and *MAP*. The results are tabulated in Table I.

The investigated recommendation systems all performed better than baseline on all 3 metrics. Intuitively, this is because all other systems take into account user-specific preferences, while the popularity baseline only takes into account global subreddit preferences. In addition, Collaborative Filtering performed best on *Precision*@10, and Mixed Similarity Diffusion performed best for *MRR* and *MAP*.

Collaborative Filtering performed well on all 3 metrics. By representing subreddits as a vector of users, and taking into account pairwise similarity over all subreddits, the user-personalized recommendations from Collaborative Filtering had the highest $Precision@10 = 0.141$. However, it is also the most computationally expensive algorithm presented here. Finding the nearest neighbors of a subreddit involves comparing against all other subreddits, and we'd have to perform this action for all subreddits a user commented on, for each user. This can take on the order of hours to days if we had to recommend subreddits for many users. We can potentially cache this information, but of course as users post new comments, the nearest neighbors may change as well. That said, there are also nearest neighbor approximation techniques that can significantly speed up this approach, such as Locality Sensitive Hashing.

By taking into account edge weights, Mixed Similarity Diffusion showed significant improvements over the original Mass Diffusion algorithm. Interestingly, Mixed Similarity Diffusion performed best when the hyperparameter $\lambda = 0$, and worst when $\lambda = 0.9$. Intuitively, this means that in the second step of the diffusion, recovering resources based on the degrees of similar users rather than the degrees of subreddits generated more relevant recommendations. Mixed Similarity Diffusion also performed best on both $MRR = 0.385$ (the first relevant result is ranked 2 to 3 on the recommendation list on average) and $MAP = 0.0968$, implying that it was not only able to retrieve, but also best rank the personalized recommendations.

Another significant advantage of the diffusion algorithms is the runtime performance, as recommendations can be generated modularly for each target user, without having to perform calculations on the entire graph. The runtime to generate recommendations for each user took an average of 15 seconds for Mass Diffusion and 25 seconds for Mixed Similarity Diffusion, where the additional time was incurred to compute the cosine similarity between target user and connected users.

The basic Random Walk showed average performance relative to the other recommendation algorithms. High degree subreddits connect a very large portion of the graph (70,000-1,000,000 users). This means that, while a starting set S of subreddits tuned towards the user incorporates some degree of individual user preference, a Random Walk without biasing the edges is still likely to visit subreddits with high degrees more. In addition, it is very likely for these very high degree subreddits connect a diverse set of users, and thus Random Walk will likely traverse to a user unrelated to the original user from a high degree subreddit. We reason that it is because of this that scaling N_s (the number of steps per subreddit neighbor s) with the degree of subreddit s, a technique used by Pixie, did not perform well for our graph.

In addition, this means that global subreddit preference has a heavier than desired influence on the Random Walk, and we reason that biasing the Random Walk using additional user preferences is crucial for good performance, as Eksombatchai et al.[15] indicated in their work on Pixie. Unfortunately, our graph does not contain more information about user preferences besides the number of comments (which did not perform well), but biasing the algorithm would be interesting future investigation. This is especially true considering that this algorithm performed the fastest of all the algorithms we tried (besides the popularity baseline), averaging at about 0.5 seconds per user.

B. Qualitative Results

In order to better understand the performance of various recommendation systems, we examine the top 10 recommendations for user "Jxxxxxxx" (full userId withheld to protect privacy). Results are tabulated in Table II. This user commented on the following Pokemon themed subreddits in the months of January to May (which we use to generate recommendations):

`CasualPokemonTrades, PokemonPlaza,`
`Pokemongiveaway, pokemontrades, pokemon`

Then, the user continued on to comment in 5 new Pokemon themed ones in the month of June (which we use as our test set of relevant subreddits):

`relaxedpokemontrades, Pokemonexchange,`
`pokemonrng, PokeMoonSun, SVExchange`

Our popularity baseline fared very poorly on this user, recommending none of the relevant subreddits, intuitively because this user has very specific tastes. Collaborative Filtering recommended the most relevant subreddits (correctly recommending 4 of 5 relevant subreddits), and none of the most popular subreddits, indicating a high degree of user-specific preference. This is followed by Mass Diffusion and Mixed Similarity Diffusion, which recommended 3 relevant subreddits, and 2-3 of the most popular subreddits. Finally, Random Walk recommended 2 relevant subreddits, and 5 of the most popular subreddits, indicating that indeed an unbiased Random Walk is too heavily weighted towards popular subreddits.

These results illustrate that the algorithms make a trade-off in how much to incorporate global subreddit preferences into user-specific recommendations. For user "Jxxxxxxx", global subreddit preferences are useless, but this does not mean they are unimportant in recommendation systems. After all,

TABLE I

EVALUATION OF VARIOUS RECOMMENDATION SYSTEMS

Method	Precision@10	MRR	MAP
Baseline: Popularity	0.118	0.3264	0.0717
Collaborative Filtering ($k = \infty$)	**0.141**	0.3381	0.0931
Mass Diffusion	0.125	0.367	0.0909
Mixed Similarity Diffusion ($\lambda = 0.5$)	0.130	0.368	0.0891
Mixed Similarity Diffusion ($\lambda = 0.9$)	0.126	0.375	0.0861
Mixed Similarity Diffusion ($\lambda = 0$)	0.137	**0.385**	**0.0968**
Random Walk ($N = 100000$, $\alpha = 0.5$)	0.126	0.344	0.0824

TABLE II

RECOMMENDATIONS FOR USER 'JXXXXXXX' (RELEVANT RECOMMENDATIONS ARE BOLDED)

Method	Recommended Subreddits
Popularity	AskReddit, funny, pics, gaming, todayilearned, gifs, worldnews, videos, aww, news
Collaborative Filtering ($k = \infty$)	**SVExchange**,PokemonCreate,**PokeMoonSun**,ShinyPokemon,**Pokemonexchange**, friendsafari, BankBallExchange, stunfisk, battleagency, **relaxedpokemontrades**
Mass Diffusion	AskReddit, **SVExchange**, **PokeMoonSun**, gaming, ShinyPokemon, friendsafari, PokemonCreate, NintendoSwitch, **Pokemonexchange**, funny
Mixed Similarity Diffusion ($\lambda = 0$)	**PokeMoonSun**, AskReddit, PokemonCreate, ShinyPokemon, **SVExchange**, gaming, friendsafari, **Pokemonexchange**, battleagency, stunfisk
Random Walk ($N = 100000$, $\alpha = 0.5$)	AskReddit, gaming, funny, pics, **SVExchange**, **PokeMoonSun**, ShinyPokemon, todayilearned, friendsafari, NintendoSwitch

the popularity baseline fared decently on our quantitative metrics, showing that users can be attracted to popular subreddits regardless of individual preferences.

C. Community Detection for Recommendation

Our folded subreddit graph contains 52,050 subreddit nodes and 384,663,141 weighted edges (weighted by the number of users that commented on *both* subreddits) in the five months from January to May. The size of our graph makes it imperative for our community detection algorithms to run efficiently. We evaluate the Leiden community detection algorithm on the folded subreddit graph and tabulate the quantitative results in Table III.

TABLE III

EVALUATION OF QUALITY FUNCTIONS FOR LEIDEN COMMUNITY DETECTION OPTIMIZATION

Quality Function	# Communities	Modularity
Modularity (Vanilla)	7	0.0156
RB Potts ($\gamma = 4.0$)	3275	0.0184
RB Potts ($\gamma = 8.0$)	7256	**0.0264**

The modularity value from using vanilla modularity with Leiden optimization was 0.0156. However, only 7 communities were detected of sizes 36680, 13145, 2195, 11, 10, 5, and 4 where the largest community was 70 percent of subreddits. Clearly we see the *resolution limit* issue, causing smaller communities to be clustered into larger communities. In previous work, Leskovec et al. [13] found that above the size scale of roughly 100 nodes the network community profile plot gradually increases, and thus there is a nearly inverse relationship between community size and community quality.

Using Reichardt and Bornholdt's Potts (RBP) quality function, with $\gamma = 8.0$, we had an improved modularity score of 0.0264 and 7256 detected communities. Notably, the vanilla modularity score improves even though we are optimizing a modified quality function. The largest community was of 2495 subreddits. Figure 7 shows the distribution of community sizes using the RBP metric. We limited our exploration of the resolution γ values due to computational limitations. To find an optimal range of values for γ, however, one could construct a resolution profile by bisectioning values of γ.

Next, we qualitatively examined communities generated by the RBP model in Table IV. While smaller communities (< 100 like Community 3, 4, 5, or 6) can contain strong recommendations they limit the number of possible recommendations. Only the other hand, subreddits in larger communities (> 200 like Community 1) are less clearly relevant and may contain merged sub-communities. However, these large communities may offer still less 'direct' recommendations. Interestingly, Community 6 contains many divorce and infidelity reddits but also contains indirect subreddits like 'FindMeFood' ('A sub to help tourists or passersby find places in the local area to eat good food with the help of reddit.') and 'Memoir' ('A place to share stories of your life events or create stories for fictional characters'). This suggests that some small communities may also be able to offer indirect recommendations.

Lastly, we examine how well the community structure we found from January to May translates to the month of June. We construct a new projected subreddit graph from comments made in the month of June. Recall that in a projected subreddit graph, an edge between two subreddits represent users have commented on both subreddits. Out of the 107,362,861 edges between subreddits in the month of June, we only consider edges with both subreddit nodes seen in the previous months. This leaves 98,966,519 edges. We recompute the modularity using the communities found by the Leiden algorithm from January to May and find a positive modularity of 0.0165 for our June subreddit graph. This

TABLE IV

EXAMPLE COMMUNITIES OF SUBREDDITS DETECTED BY LEIDEN ALGORITHM

	Comm. Size	**Randomly Sampled ($n = 15$) Subreddits**	**Analysis**
Community 1	683	FifaMobileBuySell, GTA_Vinewood, reptilians, cmake, inspiration, FetishCouples, PokemonGoIndia, culture, The_Italia, legalporno, truecapitalistradio, ETL, Frightfurs, Montages, GoldCoastTitans	No clear relationship (due to large comm. size).
Community 2	161	metalproduction, AbletonLive9, AdvancedProduction, IsolatedVocals, gamecomposers, SouthJerseyMusic, PittsburghLocalMusic, musicbusiness, FL_Studio, handpan, wavesaudiophiles, chopping, presonus, MusicFeedback4All, MusicGear	Music themed community.
Community 3	87	wii, sm64hacks, MedalHeroes, metalslug, punchout, Gameboy, dawes, retrogaming, gamecollectors, GoldenEye, AtariJaguar, nokia3310, SEGA, gamerooms, snes	Game console themed community.
Community 4	64	javascript_jobs, Web_Development, reactjs, FullStack, loljs, Heroku, learnphp, angular, ECMAScript, webaccess, laravel, elementor, Angular2, website_design_info, learnjavascript	Web development themed community.
Community 5	35	Cyclopswasright, DC_Cinematic, Fables, comicbookscirclejerk, arkhamgames, DCEUleaks, MichelleWolf, BMS, papergirls, TrueComicBooks, divergent, MUBookClub, DCcomics, batman, Superboy	Comic book themed community.
Community 6	22	AsOneAfterInfidelity, OnlineAffairs, Divorce, SingleParents, Islamic, DivorceHelp, SingleDads, Marriage, FindMeFood, adultery, deadbedroom, Custody, SurvivingMyInfidelity, naughtyfromneglect, Affairs	Divorce and infidelity themed community.

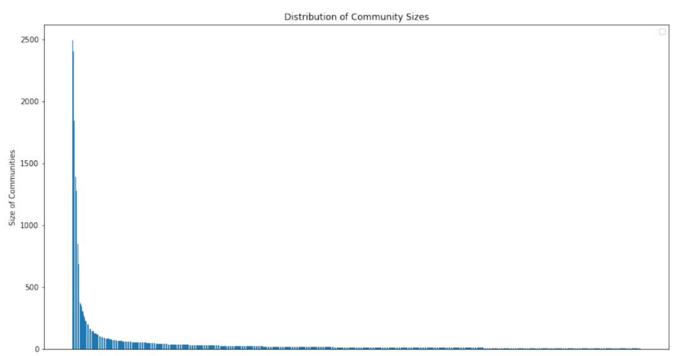

Fig. 7. The distribution of community sizes for Leiden algorithm using RBP ($\gamma = 8.0$). Large community (> 200) become uninterpretable while small communities (< 10) are Interestingly, the cumulative distribution of community sizes (not shown) appear to follow the power law, similar to [24]. It is unclear why such a distribution should arise (sociology of user interactions or dynamics of Leiden algorithm) but it is proposed as an area of future research.

suggests that using communities of subreddits identified from January to May remain strongly connected into the month of June. Users continue to comment on subreddits within the same communities, including new subreddits that they haven't commented on before. This indicates that community detection offers a basis for recommendation.

VII. CONCLUSION

In summary, our key contributions for this paper are:

1) We implement and evaluate several graphical recommendation systems on a user-subreddit bipartite graph in order to recommend new subreddits for users to comment on. These systems include both classical approaches and very recent approaches, most of which have never been evaluated before on this graph.
2) We demonstrate that while all recommendation systems outperform our popularity baseline, each recommendation system makes trade-offs in recommendation quality and runtime. In particular, while Collaborative Filtering and Mixed Similarity Diffusion perform the best, the Random Walk approach ran the fastest. In addition, each system makes trade-offs in how much to consider globally preferred subreddits in making recommendations.
3) We investigate the Leiden algorithm for community detection on our folded subreddit graph, incorporating the Reichardt Bornholdt Potts (RBP) quality function to address the *resolution-limit* problem of modularity. We show that this approach yields meaningful communities of subreddits at up to a certain size. We also show preliminary results that indicate these communities can produce relevant recommendations.

Our results indicate that implementing a personalized recommendation system on Reddit may improve new subreddit discovery at the expense of incurred computational time. In terms of future work, more work can be done in biasing the Random Walk towards user-specific subreddit preferences. For example, Reddit metadata such as user upvotes and downvotes are valuable sources of information about user preferences. The next step for community detection based algorithms is finding ways to directly incorporate community clusters into recommendation systems. In particular, subreddits belonging communities up to a certain size can offer meaningful and interpretable recommendations. In addition, we would like to investigate the performance of various content-based recommendation algorithms, and perhaps incorporate them into our graphical approaches.

VIII. LINK TO GITHUB

The code for this project is publicly available at:

https://github.com/yhjw88/subreddit-recommender

REFERENCES

[1] X. Wang, Y. Liu, G. Zhang, Y. Zhang, H. Chen and J. Lu, "Mixed Similarity Diffusion for Recommendation on Bipartite Networks," in IEEE Access, vol. 5, pp. 21029-21038, 2017. doi: 10.1109/ACCESS.2017.2753818

[2] S. Jamonnak, J. Kilgallin, C. Chan and E. Cheng, ”Recommend-dit: A Recommendation Service for Reddit Communities,” 2015 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, 2015, pp. 374-379. doi: 10.1109/CSCI.2015.64
[3] Nguyen, H., Richards, R., Chan, CC. et al. J Intell Inf Syst (2016) 47: 247. https://doi.org/10.1007/s10844-016-0410-y
[4] Baumgartner, Jason. https://pushshift.io/
[5] Zhou T, Ren J, Medo M, Zhang Y-C (2007) Bipartite network projection and personal recommendation. Phys Rev E 76:046115.
[6] M. J. Barber, Modularity and community detection in bipartite networks, Physical Review E, vol. 76, no. 6, p. 066102, 2007.
[7] X. Liu and T. Murata, Community detection in large-scale bipartite networks, in Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01, ser. WI-IAT 09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 5057. [Online]. Available: http://dx.doi.org/10.1109/WI-IAT.2009.15
[8] Larremore, D. B., Clauset, A., & Jacobs, A. Z. (2014). Efficiently inferring community structure in bipartite networks. Physical Review E, 90(1), 012805.
[9] Page, Lawrence, et al. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab, 1999.
[10] Hu, Y., Chen, H., Zhang, P., Li, M., Di, Z., Fan, Y.: Comparative definition of community and corresponding identifying algorithm. Phys. Rev. E 78(2), 026121 (2008)
[11] Atsushi Miyauchi and Noriyoshi Sukegawa. Maximizing Barbers bipartite modularity is also hard. Optimization Letters, 9(5):897913, 2015.
[12] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. Physical Review E, 69(2):026113, 2004.
[13] Leskovec, J., Lang, K. J., Dasgupta, A., & Mahoney, M. W. (2008, April). Statistical properties of community structure in large social and information networks. In Proceedings of the 17th international conference on World Wide Web (pp. 695-704). ACM.
[14] Yaozu Cui and Xingyuan Wang. Detecting one-mode communities in bipartite networks by bipartite clustering triangular. Physica A: Statistical Mechanics and its Applications, 457:307315, 2016.
[15] Eksombatchai, Chantat, et al. ”Pixie: A system for recommending 3+ billion items to 200+ million users in real-time.” Proceedings of the 2018 World Wide Web Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2018.
[16] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, J. Stat. Mech. Theory Exp. 10008, 6 (2008).
[17] L. Waltman and N. J. van Eck, J. Am. Soc. Inf. Sci. Technol. 63, 2378 (2012).
[18] Traag, V., Waltman, L., & van Eck, N. J. (2018). From Louvain to Leiden: guaranteeing well-connected communities. arXiv preprint arXiv:1810.08473.
[19] Linden, Greg, Brent Smith, and Jeremy York. ”Amazon. com recommendations: Item-to-item collaborative filtering.” IEEE Internet computing 1 (2003): 76-80.
[20] Resnick, Paul, et al. ”GroupLens: an open architecture for collaborative filtering of netnews.” Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, 1994.
[21] Deshpande, Mukund, and George Karypis. ”Item-based top-n recommendation algorithms.” ACM Transactions on Information Systems (TOIS) 22.1 (2004): 143-177.
[22] Reichardt, J., & Bornholdt, S. (2006). Statistical mechanics of community detection. Physical Review E, 74(1), 016110. 10.1103/PhysRevE.74.016110
[23] Traag, V. A., Van Dooren, P., & Nesterov, Y. (2011). Narrow scope for resolution-limit-free community detection. Physical Review E, 84(1), 016114. 10.1103/PhysRevE.84.016114
[24] Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. Physical review E, 70(6), 066111.