Interaction Graph-Based Community Recommendation on Reddit Users and Subreddits

Scott Sutherland
UC San Diego
sasuther@ucsd.edu

Ryan Don
UC San Diego
rdon@ucsd.edu

Felicia Chan
UC San Diego
f4chan@ucsd.edu

Pravar Bhandari
UC San Diego
psbhanda@ucsd.edu

Abstract

Subreddit Recommendation is a powerful tool that can enhance user experiences and improve user churn and engagement for Reddit. Reddit currently employs algorithms for subreddit recommendation; however, we seek to improve upon this recommendation system by selecting a graph-based machine learning algorithm to suggest subreddits to users through community detection along with user interactions. We utilize the tools of TigerGraph to examine various ways of representing these interactions between Reddit users as features to guide our analysis.

1 Introduction

Social media has become the most important way for people to access information, connect with friends, and expand businesses. Around 70 percent of all Americans use social media to connect with each other and share information [4]. The social media platform Reddit is a massive collection of forums where people can share news and content. Reddit is made up of more than a million communities known as subreddits; each subreddit consists of a different topic. Users in these subreddits can make posts and comments to interact with other users, essentially forming communities with specific interests. As of 2022, Reddit has 50 million daily active Reddit users worldwide [6]. Reddit users are often recommended subreddits based on those they have visited or interacted with. Like many other social media platforms, Reddit's comment representation is a tree structure where separate replies to a comment are branches. We can think of conversation in one of these trees as a comment chain, that is, a set of comments where each one (apart from the first comment) is a reply to one preceding it. We will investigate the typical size, shape and diversity of users in each of these chains. From a business point of view, the impact of this task would be in user acquisition and user retention. For current users, this would help expose them to subreddits they may be interested in, decreasing the likelihood of churn. For newer users without a few established subreddit communities they are a part of, immediately exposing them to similar communities may be beneficial in making them recurring users rather than one-time users. From a graph perspective, this problem is worth investigating as it can evaluate the relevancy of user interactions in a graph-like comment interaction structure as a basis for recommendation systems. We use a graph-based machine learning algorithm that assists in recommending subreddits to users based on their interactions with other users in subreddits through comment chains.

Previous work has been done in the space of subreddit recommendation. In one attempt, a subreddit-to-subreddit interest network was built, where two subreddits are connected if a large portion of one subreddit's members are also active in the other [7]. The model they use is K-means clustering, with another choosing to use the K-Nearest Neighbors algorithm [**?**]. These examples along with others are not necessarily graph-based. Where graph community detection tasks of subreddits are used, the graphs are represented in relatively simple ways, primarily incorporating user and subreddit subscription relationships and using that data to make recommendations using "non-graphy" criteria such as similarity scores like Jaccard [1]. By contrast, we look to use additional information which can be represented as graphs such as interactions between different users within subreddits in an attempt to leverage that interaction information as an indicator of user's affinity for different communities/subreddits. We aim to use collaborative filtering encoded within our algorithm with user interactions to then accurately recommend subreddits to users.

In order to create an algorithm ourselves, we use a dataset provided and maintained by Jason Baumgartner which can be downloaded via their website pushshift.io at its data directory subdomain. There, data for Reddit users, subreddits, posts and comments among other things are hosted in monthly time increments going back to Reddit's

inception in 2005. In order to ensure we didn't face any hardware limitations, we opted to use all comments prior to 2011 (2005 - Dec. 2010). However, because the data across all time periods is formatted in the same way, our work could easily be scaled up simply by downloading and using the more recent files. From it, we build a graph with 3 different types of nodes and 4 types of edges. Our 3 nodes are: User nodes, Comment nodes, and Subreddit nodes. Our first edge, interacted_with, connects users who interacted with other users. Second, the edges posted, connect a user with the comment they posted. Third, the replied_to connects comments that are part of a singular comment chain; in other words, when a user replies to another user's comment, we view this as an edge between the two comments. Finally, the belongs_edge connects a comment with the subreddit it was posted under.
Using the data from the source above, we create several datasets as listed below:

Dataset Name	Contents
Users-Users	Connects users who interacted with one another
Subreddits	All Subreddits in the dataset
Subreddits-Comments	All comments and the subreddit they were made in
Users	All users in the dataset
Users-Comments	Comments and users who made the comments
Comments	All comments in the dataset
Comments-Comments	Comments that are connected in a comment chain

All of the edges in this dataset are undirected, meaning that the relationship between any edge (v_i, v_j) in the graph is mutual. In total, our graph contains $\sim 172,000$ total vertices and $\sim 1,200,000$ edges. In a bit more detail, the graph contains $2,929$ unique subreddit vertices, and $\sim 170,000$ user vertices along with $\sim 865,000$ "interacted_with" edges and $\sim 345,000$ "commented_in" edges.

In particular, this dataset allows us to make this evaluation on a network like the one that can be derived from Reddit. That is, one which encodes information about the interaction of users with other users within pre-defined communities. However, as many social media platforms use very similar representations for the links between users, this data can hopefully allow us to make more broad claims about social graphs and their usefulness in making predictions about users preferred communities.

2 Methodologies

2.1 Data Preparation

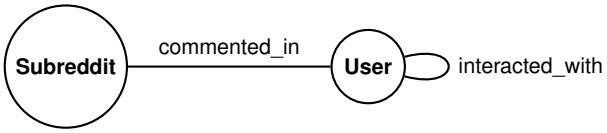
In order to fully leverage the graph structure of our data, we use TigerGraph, a graph analytics platform which provides a myriad of tools and resources for graph-based data science and machine learning, making it a valuable solution for us.

The data we downloaded from pushshift, each comment is given in the same format as the following example:


```
{
    "author": "RaptorLog",
    "author_flair_css_class": "",
    "author_flair_text": "ro ru ",
    "body": "Oh, I have been going crazy...",
    "can_gild": true,
    "controversiality": 0,
    "created_utc": 1506816002,
    "distinguished": null,
    "edited": false,
    "gilded": 0,
    "id":"dnqik36",
    "is_submitter": true,
    "link_id": "t3_73g6fg",
    "parent_id": "t1_dnqf2cj",
    "permalink": "/r/duolingo/comments/73g6fg/...",
    "retrieved_on": 1509189607,
    "score": 32,
    "stickied": false,
    "subreddit": "duolingo",
    "subreddit_id": "t5_2t6ze"
}
```


There is a lot of cool data here, but we are most interested in the user who made the comment, the subreddit it was posted to and the parent comment if any. Due to the fact that any comment on Reddit is either a child of a post of another comment, we can use this parent relationship to generate the comment chain graph structure we will set up in TigerGraph. We use this relationship to generate our comment to comment edges in our graph where the vertices are the parent and child comments respectively. We then use the author field to create edges between user vertices identified by those author names and the comment id's as well as subreddit to comment edges using the subreddit name as an identifier for the subreddit vertices. Once we have the correct files specifying those vertices and edges, we simply use TigerGraph's tools for uploading and mapping the data to generate our graph.

After loading our data to TigerGraph, we develop a schema that looks like so:



This schema consists of two vertex types (Subreddit, User) and 2 edge types (interacted_with, commented_in).

2.2 Data Analysis

Once we prepare and load our data, we do some Exploratory Data Analysis to better understand the data we are working with. First, we look into some more general statistics like the

top 10 largest subreddits with the most unique users:

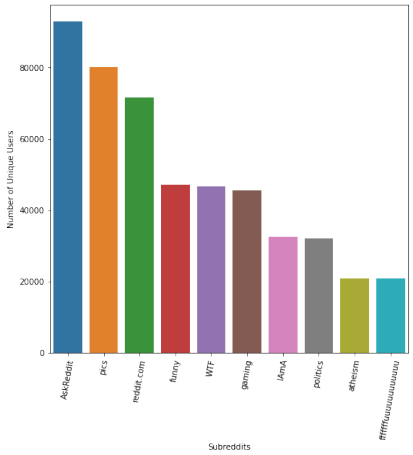


Figure 1: Top 10 subreddits with Unique Users

Looking at Figure 1, subreddits that seem more broad and common like 'gaming' and 'pics' are in the top 10 subreddits. Since they are broad, it is more likely that users will join these and be interested in these subreddits. In Figure 2, we look into the distribution of the number of comments made by users:

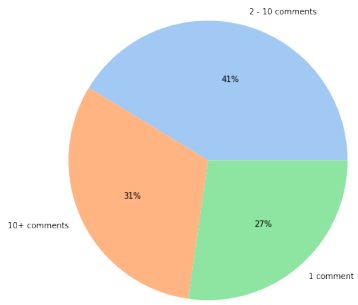


Figure 2: Distribution of number of comments made by users

We see that 27% of users have made 1 comment, 41% of users have made between 2 and 10 comments, and 31% of users have made more than 10 comments.

Next, we look into the karma of users. Karma is a score of a user/comment that is determined by the sum of all upvotes minus the sum of all downvotes. First we look into the Average Karma for a Comment by Subreddit:

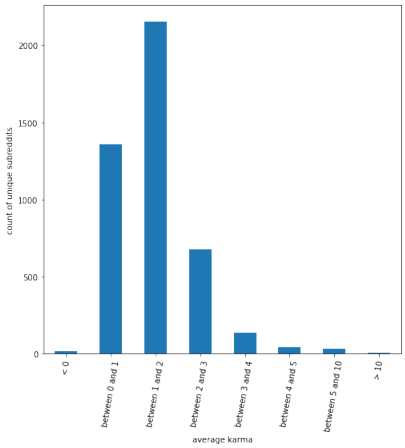


Figure 3: Average Karma for a Comment by Subreddit

In Figure 3, we can see that most of these subreddits have an average karma for comments in the subreddit between 0 and 3. Next, we look into the average karma for a comment by user:

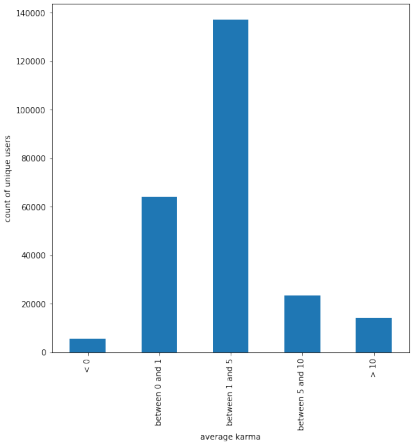


Figure 4: Average Karma for a Comment by User

In Figure 4, we see that once again, a large majority of users have an average karma between 0 and 5 for their comments. Finally, we also analyze the distribution of users by karma:

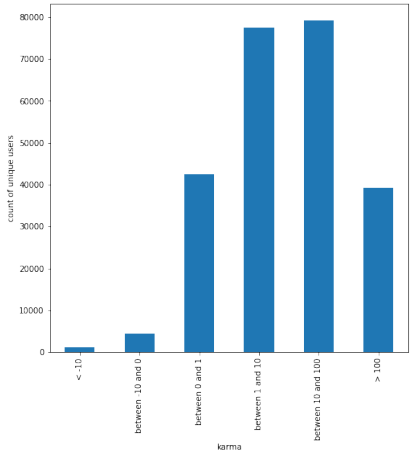


Figure 5: Distribution of Users by Karma

Figure 5 shows that a majority of users have between 1 and 100 karma associated with their accounts, while few users have negative karma.

After this, we aim to look at the Volume of Comments By day. From our dataset, we find that the majority of comments are made in the month of December. Looking at Figure 6:

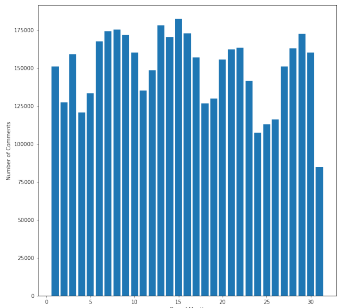


Figure 6: Volume of Comments by Day (December)

This doesn't give us much insight into large trends, but we do see that for the most part, there are several days where the volume of comments seems to be relatively similar to one another, meaning that there does not seem to be a special relationship between day of month and number of comments.

Finally, we delve a bit deeper into comment chains, as that is where the biggest focus of our model is. We find that we have 942,558 comment chains in our data. Figure 7 shows the distribution of comment chain lengths.

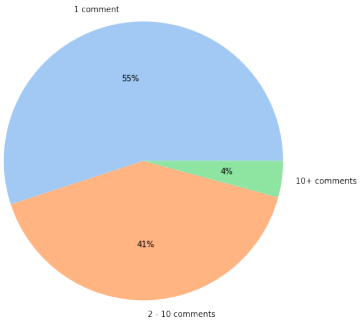


Figure 7: Distribution of Comment Chain Lengths

From Figure 7, we can see that the majority of comments do not actually belong to a comment chain. This means that the comment was posted and no-one replied. These data are likely not very useful in generating meaningful metrics for recommendation based on user comment interactions.

However, 45% of the comment data are part of chains with 41% being 'small' chains of only 2 - 10 comments and the other 4% being long chains with over 10 comments. We can examine the distributions of both of these groups:

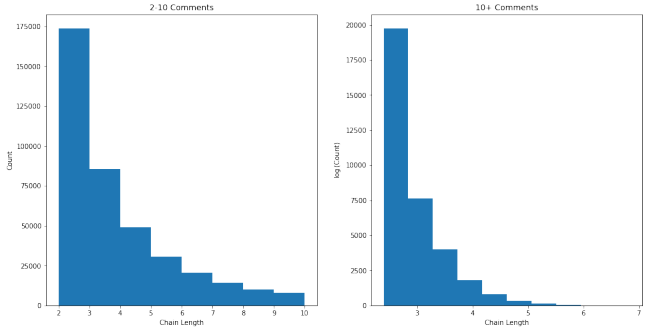


Figure 8: Distribution of Comment Chain Lengths (2-10 and 10+)

As we might expect, we generally have less chains of longer lengths. In short, there are mostly smaller chains in the data but the distribution is heavily right-skewed due to some very long chains.

Now we look at some data such as the number of users in each chain and the number of times users tend to interact in a chain. Figure 9 details the number of unique users per chain:

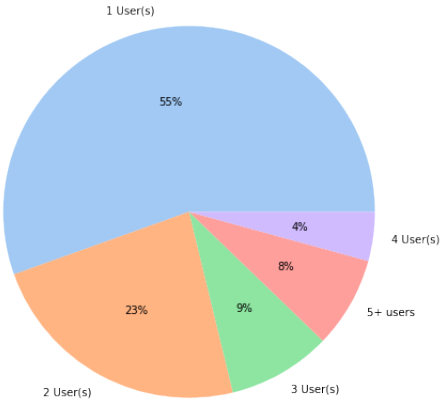


Figure 9: Distribution of Unique Users in Chains

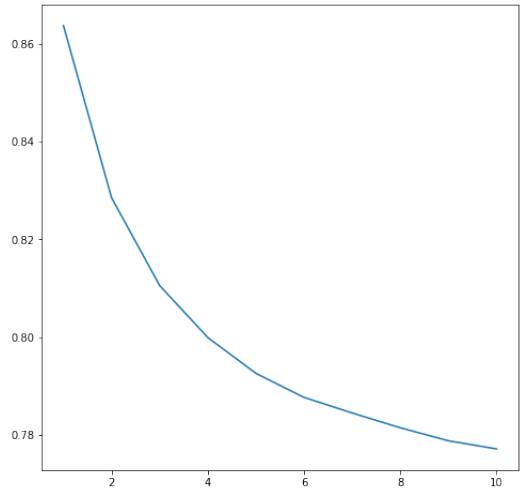


Figure 10: Average Ratio of Unique Users in Chains

Next, we look at the ratio between the number of unique users in a chain and the total number of comments in that chain. This tells us how unique the set of users participating in a chain is. Smaller ratios means there is less diversity. We find that the chains are generally quite diverse as, on average, the number of unique users is very close to the total number of comments. However, because we are including chains of length one which the ration is guaranteed to be 1.0 for, this value is quite skewed. We remove those and try again:

Minimum Number in chain	count	mean	std	min
1	428186.0	0.864	0.202	0.016
2	203798.0	0.829	0.202	0.016
3	117313.0	0.811	0.197	0.016
4	76377.0	0.800	0.191	0.016
5	54273.0	0.793	0.186	0.016
6	40719.0	0.788	0.181	0.074
7	32033.0	0.784	0.176	0.085
8	25871.0	0.781	0.173	0.085
9	21444.0	0.779	0.17	0.085
10	18123.0	0.777	0.166	0.085

This is more indicative of the data we wanted to see. Now, as we move towards larger and larger chains, the average ratio of unique users seems to plataeu at 0.78 as indicated by Figure 10.

After completing our EDA, we aim to develop baseline models for community detection so as to compare these to our final model. We develop baseline models: K-Nearest Neighbors, a recommender system using Jaccard similarity, and a simple popularity prediction model. These baselines are not necessarily graph-based, but they help us to set and measure the baseline to then build upon for our final model.

1. The K-Nearest Neighbors algorithm we develop uses cosine similarity as the metric, and chooses 20 neighbors. Cosine similarity is a measure of similarity between two data points in a plane, and in a KNN is used to determine the distance between two points. This can be found mathematically as follows:

$$cos(\theta) = \frac{A \cdot B}{||A||||B||} \tag{1}$$

where A and B are two vectors of attributes.

2. The recommender system we develop uses Jaccard similarity to recommend subreddits to users. The Jaccard index is used to gauge the similarity and diversity of sets, and can be determined mathematically as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2}$$

3. A final baseline we tried was a simple popularity predictor model, which recommends users the most popular subreddits that they are not already subscribed to.

We then build simple models in TigerGraph to calculate several different metrics based on our graph network:

- Closeness for Users: a measure of the average farness (inverse distance) to all other user nodes. Nodes with a high closeness score have the shortest distance to all other nodes. The closeness of a node x is determined by:

$$C(x) = \frac{N-1}{\sum_y d(y,x)} \tag{3}$$

where N is the number of nodes in the graph, and $d(y,x)$ is the length of the shortest path between user vertices x and y.

- Betweenness for Users: This is a measure of the percentage of shortest paths to other users that must go through a specific user node. A user node with high betweenness is likely to be aware of what is going on in multiple circles. The equation for betweenness of a given node u is:

$$B(u) = \sum_{u \neq v \neq w} \frac{\sigma_{v,w}(u)}{\sigma_(v,w)} \tag{4}$$

where $\sigma_{v,w}$ is the total number of shortest paths from node v to node w, and $\sigma_{v,w}(u)$ is the total number of shortest paths from node v to node w that pass through u.

- Eigenvector for Users: eigenvector centrality is a measure of the influence of a node in a network. A high score means that a node is connected to many nodes who have high scores. For a given graph $G := (V,E)$ with $|V|$ vertices, where $A = (a_{v,t})$ is the adjacency matrix (i.e. $(a_{v,t})$ = 1 if the vertex v is linked to vertex t and 0 otherwise.). We can find the eigenvector centrality using:

$$Ax = \lambda x \tag{5}$$

- Degree Centrality for Users: The number of edges a user node has. The higher the degree, the more central this user is.

- Degree Centrality for Comments: The number of edges a comment node has. The higher the degree, the more central this comment is.

- PageRank for Comments: PageRank counts the number and quality of links to a comment to determine how important that comment is, assuming that that comment is more likely to receive more connections to other comments.

- PageRank for Users: PageRank counts the number and quality of links to a comment to determine how important that comment is, assuming that that comment is more likely to receive more connections to other comments.

We want to use a combination of these metrics in our final algorithm to best model user interactions. We then use TigerGraph to build the Louvain algorithm and the Label propagation algorithm, which are two community detection algorithms.

- Louvain Algorithm:

  Louvain is a greedy community detection algorithm that focuses on optimizing modularity, which measures the relative density of edges inside communities with respect to edges outside communities. By optimizing modularity, Louvain results in the best possible grouping of nodes of a given network. In Louvain, small communities are first found by optimizing modularity locally on all nodes, then each small community is grouped into one node and the first step is repeated [5]. The value modularity, Q is defined as:

$$Q = \frac{1}{2m} \sum_{ij} [A_{ij} - \frac{k_i k_j}{2m}] \delta(c_i, c_j) \tag{6}$$

  where A_{ij} represents the edge weight between nodes i and j, k_i and k_j are the sum of the weights of the edges attached to the nodes i and j, respectively. m is the sum of all the edge weights in the graph, c_i and c_j are the communities of the nodes, and δ is the Kronecker delta function ($\delta(x,y) = 1$ if $x = y$, 0 otherwise) [3]

- Label Propagation Algorithm:

  The Label Propagation Algorithm (LPA) is a fast algorithm for finding communities in a graph. The LPA works by propagating labels throughout the network and forming communities based on this process. The intuition behind this algorithm is that a single label can quickly become dominant in a densely connected group of nodes, but will have trouble crossing a sparsely connected region. Labels will get trapped inside a densely connected group of nodes, and those nodes that end up with the same label when the algorithms finish can be considered part of the same community [2]. LPA works as follows:

  - Every node is initialized with a unique community label, an identifier.
  - These labels propagate through the network.
  - At every iteration of propagation, each node updates its label to the one that the maximum number of its neighbors belongs to. Ties are broken arbitrarily but deterministically.
  - LPA reaches convergence when each node has the majority label of its neighbors
  - LPA stops if either convergence, or the user-defined maximum number of iterations is achieved.

Our final model is a Network Statistics Model. An embedding is created for every user in the graph which consists of a combination of graph based features and standard machine learning features. For the graph based features, the network statistics model calculates a PageRank, Degree, Louvain, and Label Propagation score for each user node. These four scores represents how influential a user is, how active a user is based on number of interactions, what community a user belongs to based on Louvain, and what community a user belongs to based on LPA respectively. To build the standard machine learning features, we first create a corpus of text for every user using all of the comments that user has posted. We then find the top 25 most influential words from that corpus by applying TFIDF and picking the top 25 highest scoring words. Then we use a pre-trained word2vec model which converts each of the 25 keywords into an embedding of length 50. This then gives each user an embedding of size 25 X 50, representing the keywords that the user is interested in. Combining the graph based features with the standard features results in each user having an embedding of size 1254. Using this size 1254 embedding, K-Nearest Neighbors is utilized to calculate the two most similar users to the input user. We then create a pool of possible subreddits to recommend. This pool consists of subreddits that the two neighboring users are active in that the input user isn't already interacting in. In order to determine what subreddits to recommend from this pool, we use a similar approach to our standard machine learning feature pipeline. For every subreddit node, we create a corpus which consists of text extracted from 25 randomly picked comments made in that subreddit. We follow the same process as above by picking 25 keywords for each subreddit using TFIDF scores. Same as above, we apply a pre-trained word2vec model to the keywords to create 25 X 50 size embeddings for each subreddit. We then train another K-Nearest Neighbors model on the pool of possible subreddits that we created earlier. From there, we iterate through the subreddits the user has already interacted in and use K-Nearest Neighbors to find the five most similar subreddits from the pool. These similar subreddits will be the recommendations that we output to the user. If the number of recommendations is greater than the total number of unique subreddits in the possible recommendation pool, the network statistics model uses the Popular Recommender to fill in those missing recommendations.

3 Results

We run into a very common problem that many people face when trying to evaluate unsupervised recommender system models: how can we successfully evaluate our recommender system if there are no truth labels to compare to?

After researching several different methods, we decide to use Precision@k as our evaluation metric. Precision@k is the proportion of the recommended items in the top-k set of recommendations that are relevant.

$$\textbf{Precision@k} = \frac{\textbf{\# of recommended items @k that are relevant}}{\textbf{\# of recommended items @k}} \quad (7)$$

We calculate these values by pulling data from 1 year in the future from the training data and examining all the Subreddits the training users interact in that they did not interact in before. Precision quantifies how well the recommendations we make match those true interactions. Our results are below:

Algorithm	Graph-Based?	@1	@3	@5	@10	@25
Popularity Recommender	No	0.1000	0.0944	0.0729	0.0475	0.0251
Jaccard Similarity	No	0.0250	0.0201	0.0195	0.0204	0.0205
Cosine Similarity KNN	No	0.0604	0.0423	0.0372	0.0387	0.0402
Network Statistics Recommender	Yes	0.0000	0.0207	0.0338	0.0712	0.0755

Table 1: Precision@k results for all models (scores are averages across users)

Table 1 shows our results at different values of k; specifically, when @k is 1, 3, 5, 10, and 25.

When @k is 10 and 25, our final model outperforms standard recommendation techniques. However, our model fails to surpass standard models when @k is 1, 3, or 5. We believe that this under-performance may be due to the bias of users only interacting in the most popular subreddits at the time and not exploring new and upcoming subreddits based on their personal interests. As we mentioned earlier, our model was trained on data from 2010, when Reddit was relatively new and most user interactions were happening in the most popular subreddits. This is why we see such a successful precision@k for our popularity recommender that just recommends the most popular subreddits in the data set that a user is not already part of.

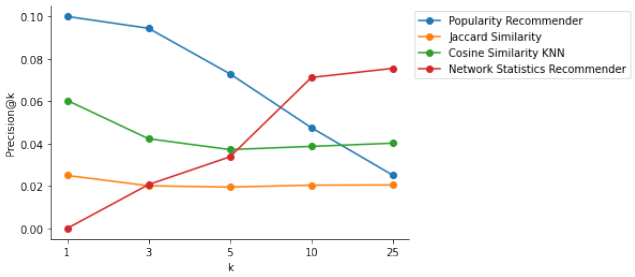


Figure 11: Precision@k for different k values on all models.

4 Conclusion

Our project showcases the potential of graph-based recommendations, which are a relatively new concept in comparison

to decades-old methods like k-nearest neighbors. However, the ability for graphs to handle complex relationships, as well as increased efficiency in data storage and computation for graphs creates a huge advantage, and the increased metrics from our final model definitely demonstrate the effectiveness of interaction graph-based recommendations for Reddit.

Future applications include graph-based and interaction-based recommendation on other social media networks such as Twitter or Facebook, where potential communities or topics can be recommended instead. For expansions done to this project in particular, we could use a more recent dataset in order to overcome the bias of popular subreddits. We hypothesize that our model will have improved results when trained on new data. We used a smaller subset of data from earlier years as otherwise the graph sizes would be too large. We could explore further algorithms within TigerGraph and keep tuning the hyperparameters for the algorithms to ensure the optimal precision@k.

References

[1] Cs224w: Home.

[2] Label propagation - neo4j graph data science.

[3] (pdf) fast unfolding of communities in large networks (2008): Vincent d. blondel: 11078 citations, Oct 2008.

[4] Social media fact sheet, Oct 2022.

[5] GRAPH, K. Louvain community detection, Jun 2022.

[6] LIN, Y. 10 reddit statistics you should know in 2023, Nov 2022.

[7] RIZIO, D. Building a reddit recommendation system, May 2021.