Pinterest Analysis and Recommendations

David Liu, Catherine Lu, Karanveer Mohan

December 10, 2014

1 Introduction

Pinterest is a visual discovery tool for collecting and organizing content on the Web with over 70 million users. Users "pin" images, videos, articles, products, and other objects they find on the Web, and organize them into boards by topic. Other users can repin these and also follow other users or boards.

Each user organizes things differently, and this produces a vast amount of human-curated content. For example, someone looking to decorate their home might pin many images of furniture that fits their taste. These curated collections produce a large number of associations between pins, and we investigate how to leverage these associations to surface personalized content to users.

Little work has been done on the Pinterest network before due to lack of availability of data. We first performed an analysis on a representative sample of the Pinterest network. After analyzing the network, we created recommendation systems, suggesting pins that users would be likely to repin or like based on their previous interactions on Pinterest. We created recommendation systems using four approaches: a baseline recommendation system using the power law distribution of the images; a content-based filtering algorithm; and two collaborative filtering algorithms, one based on one-mode projection of a bipartite graph, and the second using a label propagation approach.

2 Related Prior Work

In the past few decades, there has been much work done on creating and improving online recommendation systems. A recommendation system can be generally described as a way to predict the relationship between users and items (e.g. user ratings for products on Amazon). Broadly, there are two major categories of approaches: collaborative filtering and content-based filtering.

Collaborative filtering methods work using the previous relationships between users and items; they infer user preferences through observing user behavior (e.g. purchase history, browsing history) [1]. Since the introduction of the Netflix Prize in 2006, these methods have been actively developed and improved.

A collaborative filtering technique using label propagation is described by Baluja et. al [2]. This paper describes a personalized YouTube video suggestion algorithm called *Adsorption*. Most YouTube videos have very little context-specific information associated with them; at most, a few tags and a description are given. *Adsorption* is able to handle labeled and unlabeled objects when there is a rich graph structure underneath. In this case, the graph structure is a YouTube user-video view graph, which has undirected edges between users and videos if the user watched the video. *Adsorption* believes video v is relevant to user u if there are multiple, short paths between u and v that avoid going through high-degree nodes. That is, u and v are relevant if multiple users have watched both, and the users are not ones who watch a large number of videos (i.e. may have much broader interests than u for which we are recommending). The *Adsorption* algorithm iterates through each $v \in V$ and calculates a probability distribution over the label set, L_v. L_v is calculated as the weighted sum of the neighbors of v, then normalized to have unit L_1 norm. This repeats until convergence.

Zhou [3] et al. proposes a model of one-mode projection to compress bipartite graphs. They first create a weighted bipartite graph between users and items and then perform a one-mode projection onto one of its sets. They modify traditional one-mode projection by adding a weighting method which can be directly applied in extracting the hidden information of networks. To calculate the weights, a bipartite graph with two sets X and Y is considered. It is assumed that a resource at X is distributed evenly among all nodes in Y to which it is connected, so that a resource amount in Y can be written in terms of resource amounts at X. The resource then flows back to Y, with the weights distributed evenly among connected nodes in X. Then, the resource at each X can be written in terms of nodes at other X nodes. This gives weights for a directed graph.

Zhou et al. then proposes a recommendation algorithm as a direct result of the weighting. After weighting the one-mode projection onto X, consider putting a single unit of resource on each node. For any user, all of his uncollected objects are sorted in order of descending final weights, and those objects with highest final weight values are recommended. This approach was evaluated only on a data set with 943 users and 1682 objects; it is unclear how well this algorithm would perform in regards to speed and accuracy when the number of objects is much larger than the number of users.

Content-based filtering techniques use characteristics of the users and items (e.g. match items with certain characteristics based on users' preferences). One example is PRES (*P*ersonalized *Re*commender *S*ystem), a content-based filtering system for suggesting small articles about home improvements. Terms are extracted and cleaned from documents, and a document is represented using the vector space model and specifically tf-idf as the weights for each term. User profiles are also represented as documents in the same way. Meteren et. al then use cosine measure to determine similarity. However, their data set only consisted of 3 fictious users, so it has not been shown to work on real or larger scale data.

3 Data

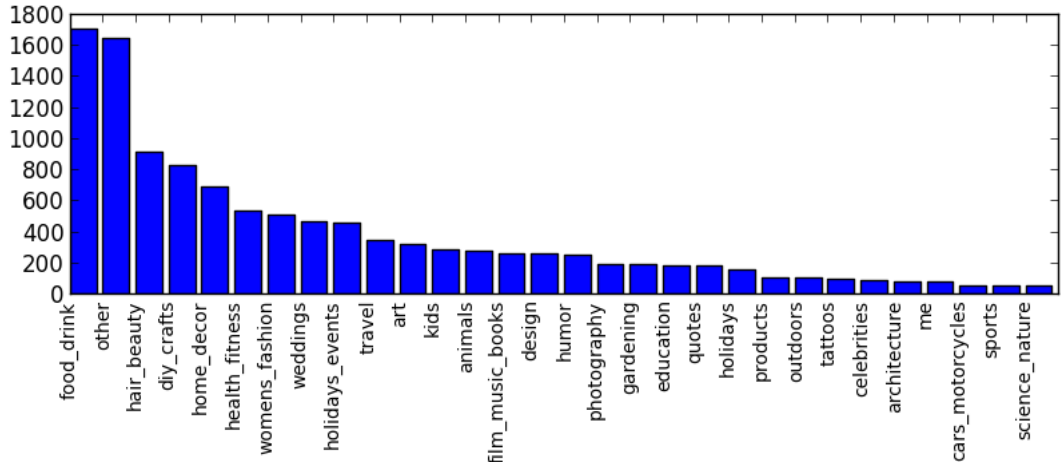
Our data consists of a sample of Pinterest's users along with all of their boards and pins. This sample consists of about 3K users, 27K boards, 1M unique pins and 2.2M pins total. 25K boards and 950K pins are labeled with interests associated with the board or pin. Further, there is at least 1 pin term for 945K pins. Pin terms are extracted in a Pinterest-internal workflow which uses TF-IDF-based aggregation of pin descriptions, board titles, and a dictionary of entities, identifying a specific topic for each pin, such as "The Hunger Games," and an associated weight between 0.02 and 13.48. A higher weight means more confidence that the term is relevant to the pin. The total size of this dataset, including additional metadata, is about 4GB uncompressed. This data is not publicly available but is made available to us by Pinterest.

The data was sampled by starting with 3K randomly sampled users, and subsequently all of their boards and pins were added to the data set. Additional users who had pinned these pins were also added to the data set.

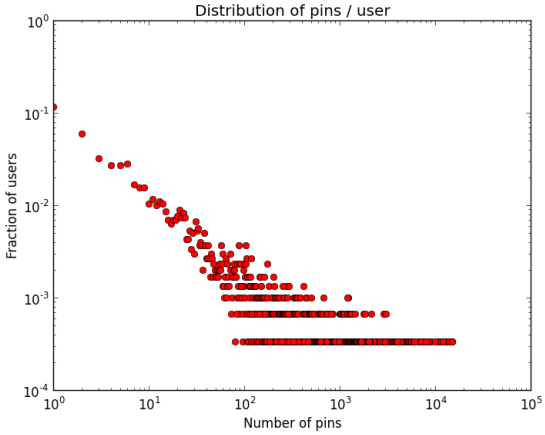
4 Data Characteristics & Analysis

2101 (70%) of users were female, 420 (14%) were male, and 483 (16%) were unspecified. Users came from 116 countries with the vast majority (1995 users or 66%) being from the US, followed by 138 (4.6%) users in Canada and 125 (4.2%) users in the United Kingdom. Within our data set, only 35 countries had 5 or more users and 38 countries had only one user representation.

The Pinterest website shows 37 board categories, such as Gifts, Videos, and Animals. Many of the boards in our data set did not have a board category. Among the 11.6K boards that did have valid categories, we graphed the 30 most common board categories. Food & Drink (1710 boards, 15%) was the most popular category, followed by Other (1646 boards, 14%) and Hair & Beauty (910 boards, 7.9%).



The pin behavior among users followed a power-law. The most active Pinterest user within the data set had 15K pins, but 26% of users had 5 or fewer pins. The graph below shows the user-pins count on a log-log plot.



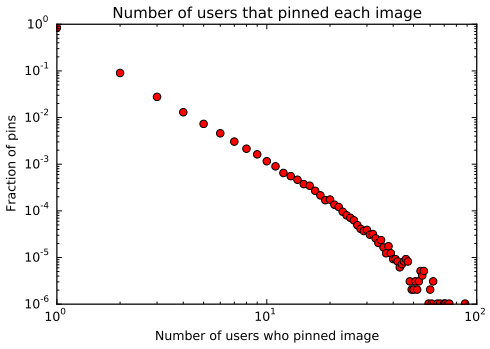
We also looked at re-pinning behavior. Overall, 92% of pins were re-pinned. This is a promising statistic, since it suggests that users will be more receptive to Pinterest recommendations than if most of their pins were generated from outside of Pinterest. Thus, our recommendation system may have real influence in encouraging users to create more pins.

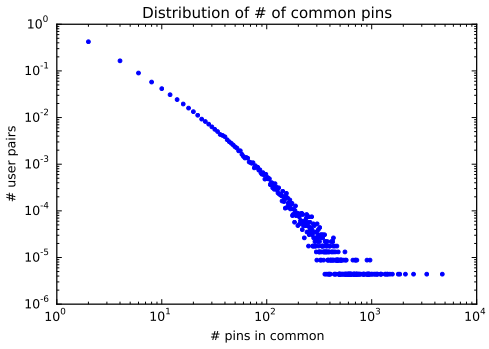
4.1 Network Analysis

The data set we analyzed consisted of images with a wide range in the number of users who had pinned those images. We found that the distribution followed a power law fairly well. This is likely due to a preferential attachment-like process in the Pinterest system; images which have been pinned a large number of times 1) have more chances to be seen by users on the site, and 2) are likely to be of higher quality, so users are even more likely to repin them. These factors would both contribute to the power law distribution observed (below left).

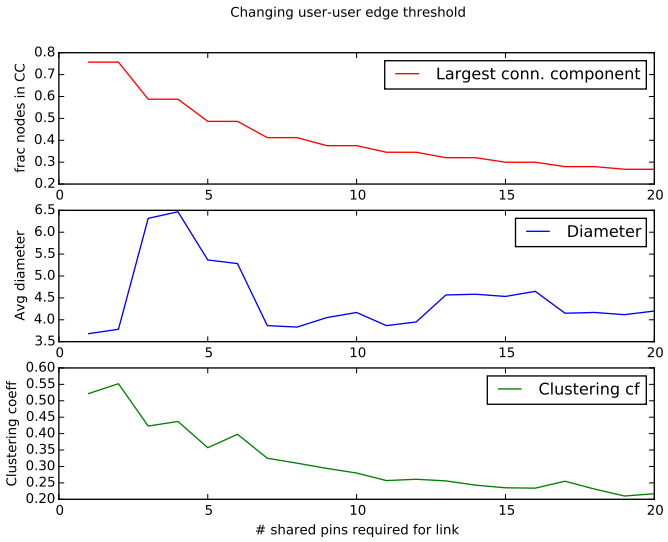
We focused on analyzing the graph of users, where users were defined as being connected if they have pinned the same image, either by directly repinning an image from another user or by pinning another copy of the image. Ultimately, we would like to recommend pins to users, so this analysis will give us more information that may help to determine which users are similar to the target user. The distance in this graph should reflect the similarity of the users' interests.

We began by defining the graph of users as a weighted graph, where the number of images shared between two users gives its weight. Below right is the distribution of edge weights.

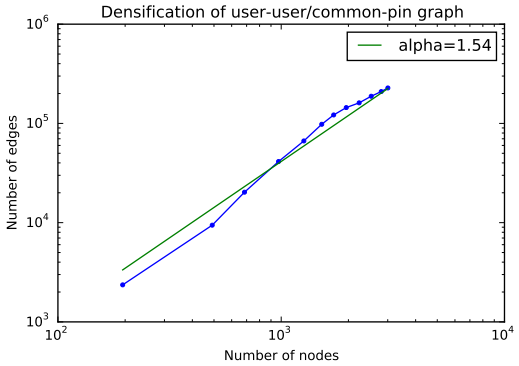




For further analysis, we thresholded the edges, retaining only those with weight greater than k. The below graph indicates several properties of the graph with varying k. The top chart shows that the size of the largest connected component decreases as k increases and fewer edges are included. The diameter of the graph, estimated from a subsample of random nodes, increases sharply as k increases. Combined with the low average diameter of 3.6 at $k = 1$, this suggests that the graph consists of many nodes which are linked together by only one or two common pins. Similarly, the clustering coefficient is quite high when linking all users with just 1 or 2 pins in common.



Finally, we looked at an evolution of the network over time. Since both users and pins have creation timestamps, we examined the growth of the graph over time. The observed graph of users to edges below indicates that the graph indeed follows the densification power law, with $|E| \approx |N|^{1.54}$.



5 Recommendation Systems and Results

5.1 Baseline Recommendation System

From the previous section, we learned that the number of users that have pinned a certain image obeys a power-law distribution. Our baseline recommendation system calculates the probability that an image is pinned by a user by assuming it is proportional to the current number of users that have pinned that image.

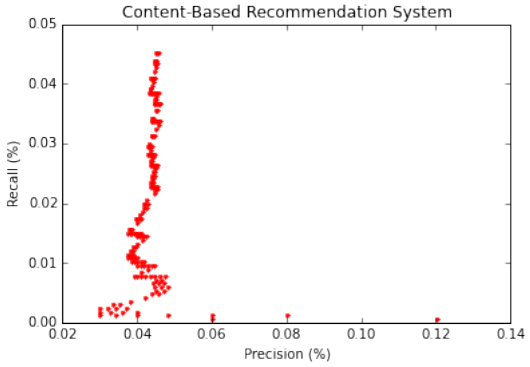
We only evaluated the baseline recommendation system on users that had 300 or more pins thus far, so that we would have enough data to provide recommendations. There were 691 such users. For each of these users, we randomly selected 200 of their pins for testing and used the remaining pins for training.

The recommendation system picks the k top pins for the test set for each user, with k ranging from 1 to 200. Then, letting R_k be the set of the k top recommendations, and T be the set of true images actually pinned by the user, we use the following metrics.

$$\text{Precision} = \frac{|R_k \cap T|}{k} \qquad \text{Recall} = \frac{|R_k \cap T|}{|T|} \qquad \text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{||\text{Precision}|| ||\text{Recall}||}$$

Precision-recall is a common evaluation metric for information retrieval applications [2]. For this data, we assumed that if a pin has not been repinned by the user, it would not be repinned if shown to the user. This is true with high probability because the overwhelming majority of pins that are seen on the site are not actioned on by the user: users see hundreds of pins per session, but only pin a few each time. Baluja et. al [2] made the same assumption for YouTube users.

Our results are below. We found that this baseline system performs quite well, considering that random selection would have a 0.02% chance of selecting a relevant pin for the user. The expected precision for a given user is thus 0.02%. We can see that the baseline algorithm performs much better than random (its precision and recall are slightly more than 2.5× greater). Note that the precision-recall curve doesn't follow the normal sloping line, since the recommendations are not ranked in any particular order. So, the shape of the curve we got is expected.



The baseline recommendation system is crude in that it only uses how popular a pin is, without any information about the pin or users. Thus, the recommendation system is not personalized per user, and it makes the assumption that the pin's current popularity is entirely predictive of the pin's future popularity.

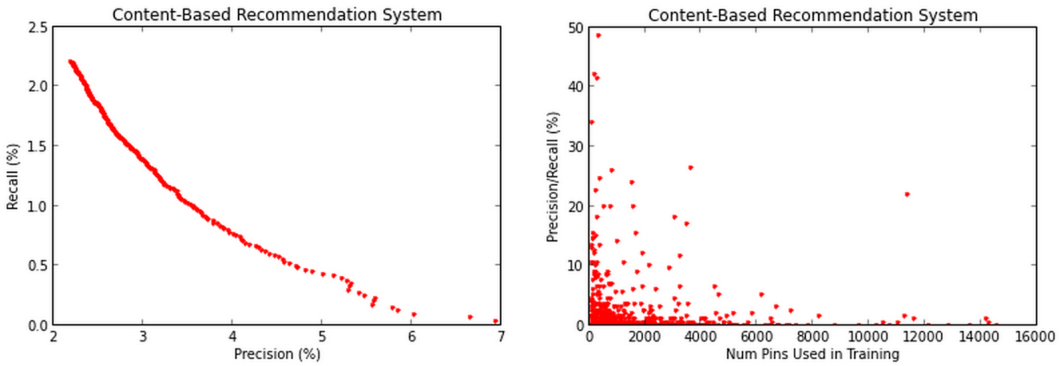
5.2 Content-Based Filtering Recommendation System

We implemented a content-based filtering recommendation system similar to PRES [4]. Our approach was to use the pin terms (see Section 3 for details) as the content.

As before, we only evaluated users that had 300 or more pins thus far, so that we would have enough data to provide recommendations. For each of these users, we randomly selected 200 of the pins for testing and used the remaining pins for training. The training period consisted of creating a user profile for the person based on the pin terms available. A user profile is a very sparse feature vector where each entry is the sum of the term weights among all training pins divided by the total number of training pins. More formally, for a pin's feature vector F, the ith entry f_i corresponds to the weight of the ith pin term. The user profile is thus created as $\frac{1}{N}\sum_{j=1}^{N} F_j$ where F_j corresponds to the jth pin's feature vector out of N in the training set. We then used cosine-similarity of the user profile to all pins with terms and ranked them, where cosine-similarity between two feature vectors F_1 and F_2 is $\frac{F_1 \cdot F_2}{||F_1|| ||F_2||}$.

The precision-recall curve is below left. As we can see, our simple recommendation system does well, especially considering that our recommendation system must choose relevant pins among the 950K pins with terms. We calculate precision/recall at the end of the 200 pins, so that precision and recall are the same. We had a maximum average F1 score of 2.2, which was achieved after recommending 200 pins. This is more than a 40× improvement over the baseline.

We also analyzed how the number of training examples in the set affects the precision/recall (below right). There is no clear correlation between the number of pins in the training set and the precision/recall. This is unfortunately not what we would expect. One possibility is that for users who have many more pins, they are more likely to have pins that span many board categories and interest areas. Thus, it is harder to choose relevant recommendations. Another possibility is that users with more pins are more likely to pin/re-pin the images that they're exposed to. Thus, our initial assumption that users rarely pin images is false, at least for these more active pinners. Figuring out how to take advantage of the users with a lot of training data will be a challenge.

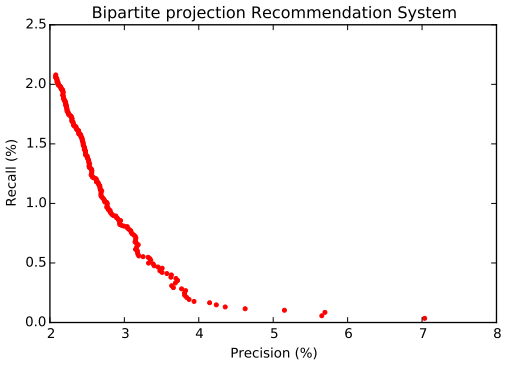


5.3 One-Mode Projection Recommendation System

This is a graph-based approach where we can consider the users as nodes in a set X and pins in a set Y. Edges go from users to pins they have pinned resulting in a bipartite graph. We then project this bipartite graph onto set Y. This is done in the following manner: If a user X_i pinned pins Y_j and Y_k, an edge is added between Y_j and Y_k with a weight one. If another user has also pinned the exact two pins, the weight is increased by one.

Similar to before, we only consider users with 300 or more pins and create the bipartite graph with 200 of their pins randomly held out as the test set. After we project the graph onto set Y, we proceed as follows. For each user, we look at all pins they have not already pinned. For each unseen pin, we create a score. The score of an unseen pin is the sum of the weights of the edges from each of the user's pins to that unseen pin. We recommend the pin with the highest score calculated in this way. Intuitively, we want to recommend the pin that is most strongly connected to all of the user's existing pins.

Unfortunately, if we consider all users with more than 300 pins, we end up with more than 900,000 pins. Since the number of edges in the one-mode projection grows quadratically in the number of pins, this algorithm ends up being very slow. Zhou et al. had only considered a graph with 1682 objects and we are 500× bigger. Therefore, we used a reduced data set size, only considering users with more than 300, but less than 1000 pins. This decreased the number of users from 691 to about 400 users. To further speed it up, we only considered each pin's top 10,000 edges by weight. Despite these pruning heuristics to speed up the algorithm, one-mode projection performs quite well, achieving an F1 score of slightly more than 2.0.



5.4 Adsorption

Adsorption is a collaborative filtering-based graph algorithm simlar to label propagation. The goal of adsorption is to infer the "labels" of nodes based on the labels of similar nodes. In a recommender system, the labels correspond to items (pins) and the nodes correspond to users. The specific algorithm we used is explained in Baluja et al. [2]

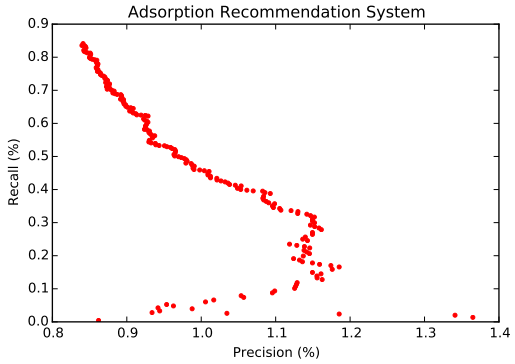
Whereas Baluja et al. used a bipartite graph of users and items, and then assigned labels to both types of items, we chose to use the projection of this graph onto users for ease of implementation. This formulation is simpler because there is only one type of node (users), but it is much more computationally expensive to produce the graph. Users were linked to one another if they had pins in common, with the

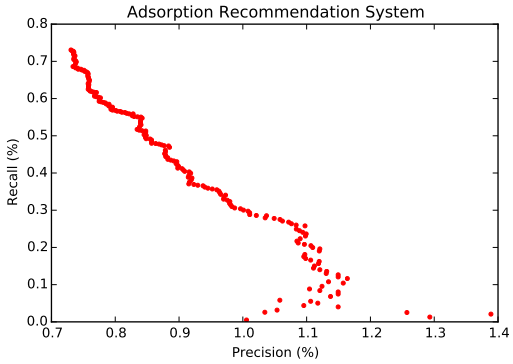
weight of the edge representing how many images they had pinned in common. This produces up to $O(|U|^2)$ edges instead of $O(|U| \cdot |P|)$, but it is still tractable for our sampled data set.

Each user u has a label vector $l^{(u)}$, initialized such that $l_p^{(u)} = \frac{1}{|P_u|}$ for each $p \in P_u$, the user's set of pins. On each iteration of adsorption, the labels are propagated by assigning each node a new label vector $\propto \sum_{v \in U} w(u,v) \cdot l^{(u)}$, normalized to have L_1 norm equal to 1. That is, the new label of a node is the average of its neighbors' label vectors, weighted by the weights of the edges.

On each step of the propagation, the label vector is then truncated so that only the top m entries are non-zero. This truncation of the vector is essential to maintaining good performance of the algorithm: it limits the number of labels stored to $O(m * |U|)$ instead of $O(|U| * |P|)$. We ran the propagation steps repeatedly and picked the precision-recall from the iteration that achieved the best F1 score.

m trades off accuracy of the adsorption algorithm for speed, since label information is discarded. Below we show precision-recall curves for two runs of the adsorption algorithm, first with $m = 5000$ and then with $m = 500$. $m = 500$ performs only slightly worse, suggesting that using $m = 500$ is a good trade-off to make.





5.5 Comparing Recommendation Systems

The content-based system produced the most accurate recommendations, as defined by maximum $F1$ score achieved. The full ranking is as follows:

Method	F_1 score at $k = 200$ recommendations
Content-based	**2.2%**
One-Mode Projection	2.0%
Adsorption, $m = 5000$	0.85%
Adsorption, $m = 500$	0.72%
Baseline (popularity)	0.05%
Baseline (random)	0.02%

The content-based recommendation system, for our given data set, slightly out-performs one-mode projection. Further, content-based out-performs adsorption by more than 2.5×. Howver, this makes sense based on the data we have. The sparseness of our repin graph makes it very difficult for collaborative filtering methods to work well: no image has more than 100 repins in our data set. Content-based filtering uses terms associated with the pins, allowing it to connect pins even when there is not a path of mutual repins between users with the same interest. Since there is rich data available in the pin content beyond the repin graph, the content-based filtering method is able achieve much higher accuracy. As a concrete example, suppose user 1 has pinned pins with IDs a, b and c that have the term "Game of Thrones" associated with it. If user 2 has pinned a pin with ID d also associated with the term "Game of Thrones," a content-based system will be able to infer the relationship to pins a, b and c, and a purely adsorption-based system would not.

In terms of F1 score, our content-based algorithm does more than 40× better than our baseline popularity-based recommender and 2× better than adsorption, although it is only slightly better than one-mode projection.

We also want to compare the memory usage and speed among the various recommendation systems. For any recommendation system on a large social network, scalability and performance are of the utmost importance. In this investigation, the implementations are not highly optimized. However, the relative rankings still hold. In order of best to worst performance:

1. The baseline recommendation system is the cheapest to compute, since it randomly chooses a pin according to a global probability distribution.

2. The adsorption algorithm's speed is tuneable by the truncation constant m, trading off accuracy for runtime. We found that we can easily achieve good results while keeping the algorithm efficient.

3. While content-based filtering gave the best results, it is slow compared to adsorption.

4. One-mode projection ends up being the slowest of all algorithms, although its accuracy is comparable to content-based filtering.

6 Discussion

We found that the Pinterest pin and user networks are very similar to many real-life networks described in class. User activity and pin popularity both follow power law distributions.

We also explored four different algorithms for recommending pins to users. Our baseline implementation for recommending pins based on popularity was 2.5× better than random predictions. Content-based filtering and one-mode projection performed 40× better than baseline, reaching an F_1 score as high as 2.2, whereas adsorption had accuracies 15-20× higher than the baseline. We determined that this was likely due to the sparsity of the user-to-pin graph; since most pins only have very few users pinning them, it is difficult to make recommendations based on those pins. Content-based similarity allows us to take full advantage of all the metadata associated with a pin, rather than just its relationships in the user-to-pin graph.

There are many possible directions to improve the accuracy of the recommendation systems. One way to increase the accuracy of the recommendation system is to train a hybrid model using both graph-based features and content-based features. Many real-world systems use a hybrid approach for recommendations, since they often out-perform either approach alone [5]. For instance, the hybrid recommendation system may use the rankings achieved from the content-based filtering system combined with the rankings achieved from the baseline system. Another way, for collaborative filtering techniques, is to increase the amount of data we have available so that the sparsity of the user-to-pin graph is reduced. For a user currently in our data set, there may be many more users who have pinned the same image that are not in the data set; thus, we have more sparsity in the user-to-pin graph from our data set than would be the case in the actual graph. For the content-based recommendation system, additional features could be added as signals for training.

There are also possible directions to explore to improve the performance of the recommenation systems. Some of these algorithms, like adsorption, present themselves much more easily to be implemented using a distributed framework like map-reduce. In general, performance optimization is very specific to the algorithm in question. These algorithms can be expressed or approximated with varying degrees of efficiency in map-reduce, and other distributed computing models may be appropriate. Another way is to implement the algorithm in a high performance language.

It will also be worth exploring an online recommender system that updates the recommended pins as time progresses and doesn't recommend a pin once the user has seen it.

We could also consider doing graph-based recommendations on other entity types, for instance recommending boards for the user to follow as opposed to images to re-pin. We could also find interests that are related to one another, or find pins that belong to an interest given the boards that it belongs to. Or, pins in common could be used to find other users that are very similar to a particular user.

7 Contributions

All group members contributed equally to the final project. Specific task distribution:

- Literature Review: Catherine, Karanveer
- Data Sampling and Export: David
- Data Analysis: Catherine
- Network Analysis: David
- Content-Based Recommender: Catherine
- One-mode projection: Karanveer
- Baseline Recommender: Karanveer

- Adsorption Recommender: David, Karanveer
- Write-up: David, Karanveer, Catherine
- Poster: Catherine, David

References

[1] Yehuda Koren. 2008. *Factorization meets the neighborhood: a multifaceted collaborative filtering model*. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08). ACM, New York, NY, USA, 426-434.

[2] Shumeet Baluja, Rohan Seth, D. Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph. In Proceedings of the 17th international conference on World Wide Web (WWW '08). ACM, New York, NY, USA, 895-904.

[3] Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. *Bipartite network projection and personal recommendation*. In Physical Review E 76, 046115 (2007).

[4] Robin van Meteren and Maartan van Someren. *Using content-based filtering for recommendation*. ECML/MLNET Workshop on Machine Learning and the New Information Age, pages 47-56.

[5] Robin Burke. 2007. *Hybrid web recommender systems*. In The adaptive web, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Lecture Notes In Computer Science, Vol. 4321. Springer-Verlag, Berlin, Heidelberg 377-408.