# Social Networks: Finding Highly Similar Users and Their Inherent Patterns

Chris Tanner[*]
Department of Computer Science
Los Angeles, CA 90095
christanner@ucla.edu

Irina Litvin[†]
Department of Computer Science
Los Angeles, CA 90095
litvin@cs.ucla.edu

Amruta Joshi[‡]
Department of Computer Science
Los Angeles, CA 90095
amrutaj@gmail.com

## ABSTRACT

Social networks provide a framework for connecting users, often by allowing one to find his pre-existing friends. Some social networks even allow users to find others based on a particular interest or tag. However, users would ideally like the option of finding others who share *many* of their interests, thus allowing one to find highly-similar, like-minded individuals whom we call *clones*. This functionality would not only be a strong improvement to current social networks, but it could yield interesting research work regarding graph theory and sociology.

In this paper we explore a means for finding "*clones*" within a large, sparse social graph, and we present our findings that concern patterns of shared interests. Additionally, we explore how this correlates with connectivity and degrees of separation. With our introductory work, we hope to encourage future work regarding developing efficient, accurate algorithms for finding clones. In addition, we aim to inspire others to further our data mining efforts toward understanding the relationship between connectivity and having shared interests.

## Categories and Subject Descriptors

H.4 [**Data Mining**]: Social Networks

## General Terms

Data Mining, Social Networks, Clone Finding, Facebook

## Keywords

Social networking, Data mining, Facebook, Clones

[*]Responsibilities included general research direction, writing code, and authoring documents

[†]Responsibilities included general research direction, managing program design decisions, and coordinating priorities

[‡]Contributions included implementing Min-Hash and LSH, and providing research advice

## 1. INTRODUCTION

Social Networking web sites consume a large percentage of today's internet traffic, and are amongst the most frequented web sites in the world: As of October 19, 2007, Myspace.com ranks at #6 and Facebook.com at #7 [1][2]. In fact, these network sites have between 60 and 200 million users[4][5]. Naturally, this global phenomenon has invoked much research to be conducted. Typically, researchers treat an entire social network as simply being a large, undirected graph. This naturally invites much graphy-theory-related research to be conducted, allowing one to find patterns, cliques, and other interesting features. Specifically, there have been efforts to understand the global structure and connectivity [3][12], relating the findings to the six-degrees of separation, small world phenomenon that was first introduced by Milgram in 1967 [6][7]. Recently, many analysis approaches have been proposed [8][9]. Yet, we focus on a new problem: finding and understanding correlations between highly similar users.

There have been research efforts to study users' interests, but these were for the sake of analyzing various measurements and predicting taste performances [11][14][13]. Moreover, we believe we are the first to relate similarity with connectivity. And, unlike most related research efforts, we concentrate on a *pure* social network–Facebook.com–in the sense that our medium of choice is one whose entire premise is based on developing social networks. This contrasts with many other *loose* social networks such as photo or video sharing sites (i.e. Flickr, PicasaWeb, YouTube, Google Video) whose primarily focus is on providing featured content. These loose networks allow for social-network-like communities to form, but this is more of a complemented, secondary feature and is not the premise of the site. In order to provide realistic results, we focus on the closed, small-world dataset of the entire regional network of Los Angeles, California. The specifics of our dataset, along with our reasoning for choosing our dataset, are furthered mentioned in Section 3.

In our approach, we first represented all users and their interests as a large, sparse matrix. We applied Min-Hashing and Locality-Sensitive Hashing to find users who are potentially very similar. We call these users *candidates*, and we describe this work in Section 4. In Section 4.4, we discuss our method for calculating the exact similarity shared between these candidate users, for not all candidates are necessarily clones. Section 5 mentions our work concerning connectivity. Specifically, we aimed for calculating the av-

erage degree of separation between clone pairs. Our results and data analysis are discussed in Section 6. The myriad of potential research for future is listed in Section 7, and we summarize our work in Section 8.

## 2. BACKGROUND

If you have ever used a social networking site, you can probably skip this section. Yet, to be self-contained, we provide the following overview:

Although social networking sites may differ greatly from one another, as seen by the aforementioned contrast between *loose and* pure social networks, much of their underlying concept remains constant: each active user represents a unique entity that represents oneself. Each user has a customizable *profile*, whereby the user may provide a picture of himself and list characteristics to represent who he is in the real world. Social networking sites typically provide a general template of these characteristics, allowing users to enter their favorite activities, interests, music, movies, and general information about themselves. Optionally, users may choose to leave any of these fields empty. In addition, users may add their own fields to their profile. In fact, recently, there has been an enormous burst in the development of "applications" for users' profiles. These applications are essentially displays of enriched content, which often make use of and/or allow input from others. Along these lines, profiles are also becoming increasingly rich in content, for users may typically upload pictures, videos, and other media that includes them or interests them.

Now that a user has a profile, he may want to set privacy restrictions so that not everyone can view his information. For example, if the user lists his home address, phone number, or embarrassing/incriminating pictures, he obviously may want to protect such from the public eye. Social networks allow for communities-like sub-networks to form. These networks are generally founded to represent members of a shared university, geographical region, company for employment, or interest. With this, users may conveniently set their privacy controls based on each of the networks to which they belong.

Having discussed a user's profile and the existence of subnetworks, we lead to the key element of social networking: friends. Users may often search or browse for other users. Upon finding other users, generally one would want to view the found user(s) profiles. This may or may not be possible, depending on the aforementioned privacy controls. However, a user would not want to accidentally limit a long-lost friend from viewing his profile, preventing them from rediscovering each other again. So, usually users have somewhat relaxed privacy controls–specific enough not to allow *everyone*, but general enough to allow people who belong to the same networks. We mention this because it plays a key role in devising ways and knowing what is possible for us to crawl. Nevertheless, users may find each other, and *friendships* may form: user A making a request to be friends with user B. User B may choose to accept or decline the friendship request. Note that friendship is always bi-directional: if user A is listed as one of user B's friends, then user B will be listed as one of user A's friends.

With our mention of users and friendship, it should be clear that we can represent a social network via a set of graphs:

- Let vertices $V_1$ and $V_2$ represent two users. $V_1$ and $V_2$ are directly connected via edge $E$ if they are friends with each other
- Construct the connections $\forall\ V_i \in V$, where V = all users and $V_i$ represents a unique user

In fact, our work concerning connectivity makes use of this graph format. More important though is our representation of users and their interests. We will later formally define our notation, but let it be stated that we can form a matrix where the columns are users within our social network and rows are all of the interests that are listed by the users. We form this structure not only for intuitive design, but because there exists efficient algorithms for finding very similar columns. Naturally, this would allow us to find users who are highly similar to another user. As mentioned, this functionality does not currently exist within social networking web sites, and we would like to encourage this feature to be added.

## 3. DATASET

Ideally, we would wish to have data for the entire network of Facebook's 60 million users. As mentioned, due to privacy limitations (some people only allow their friends to view their profiles), computation resource limitations (60 millions user profiles would consume roughly 6 Terrabytes of data), and crawling prohibition (Facebook bans accounts that crawl), this was not possible. So, how do we choose to sample? Research has shown that regional networks–specifically Los Angeles–of Facebook possess the characteristic of being a closed, small-world representation of the all-encompassing network [3]. Thus, we "limit" our dataset to the entire regional network of Los Angeles, California.

There were approximately **320,000 users** within the Los Angeles, California network when we began our research. Yet, *many* of these have the aforementioned privacy settings such that we could not view all users' pages. Nevertheless, we obtained **176,786 unique user profiles,** and we respected privacy issues by removing all identifiable information: every user's real name was converted to a unique ID #. Moreover, for each user, we only maintained: a listing of their friends' IDs, and all of their listed interests. Note that Facebook allows each user to list his favorite items within (6) distinct categories: *activities, interests, music, books, T.V., movies.* Naturally, not all users supplied information for every category. Additionally, the nature of the data lends itself vulnerable for noisy input and words with identical meanings but different syntactical representations. So, to help combat this issue, we removed all listed items that only occurred once. We assert that this will not remove legit information because it is unrealistic that a person is so unique that he likes a particular item that none of the other 176,000 users like.

## 4. FINDING HIGHLY SIMILAR USERS

Since we have over 176,000 crawled Facebook user profiles, the number of possible clones in our dataset is approximately

$\frac{176{,}000^2}{2}$. Computing similarity values for each of these pairs is practically infeasible given our limited resources. Hence, we first detect candidate pairs for clones and later compute the actual similarity. To generate candidate pairs we employ Min-Hashing and Locality Sensitive Hashing (LSH). We provide a brief overview of these algorithms.

## 4.1 Min-Hashing and Locality-Sensitive Hashing

We represent users and their interests as an incidence matrix. We represent users as columns and interests as rows. $A$ is an incidence matrix. $A[i, j] = 1.0$ indicates that user $j$ lists interest $i$ in his Facebook profile. Likewise, $A[i, j] = 0.0$ indicates a lack of interest. Thus, each user is now represented by a column vector where value 1 in the column indicates user's interest. Our objective is to find similar users based on their common interests. (i.e., we want to find similar columns in matrix $A$ based on the co-occurrence of 1's in the rows).

We define similarity among users as the Jaccard co-efficient between its two column vectors. Thus, similarity between users $i$ and $j$ is defined as

$$S(u_i, u_j) = |C_i \cap C_j| \,/\, |C_i \cup C_j|$$

where $C_i$ and $C_j$ are the two column vectors corresponding to users $i$ and $j$. Similarity between users can be found by using Min-Hashing and Locality-Sensitive Hashing.

## 4.2 Min-Hash

Min-Hashing is a hash-based technique to obtain a compressed representation/signature of our interest-user incidence matrix's columns such that the probability that signatures of two columns are equal is directly proportional to their similarity. The underlying idea of min-hashing is to randomly permute the rows and to hash each column to the first row in which that column has a value of 1. One such shuffle creates only one row. To obtain a compressed vector representation of the column, we shuffle the matrix 100 times. The original matrix's rows are read in, and each row is supplied to 100 hash functions. For each column where there is 1, a hash value is calculated and the lowest hash value is maintained. After completing a single pass over the entire matrix, we obtain 100-length signatures for each column. We implement the 100 hash functions using universal hashing. Our hash function is defined as $(ax + b)\%p$, where $x$ is the key, $a$ is a prime number in the range $[1, p]$, $b$ is a random number in the range $[0, p]$ and $p$ a prime number greater than the number of rows in the original matrix. Thus, we obtain signatures for each user.

## 4.3 Locality Sensitive Hashing

Once we have reduced each column of our interest-user incidence matrix to min-hashed signatures, our data easily fits into memory. However, performing similarity computations on all possible pairs of users is still computationally expensive. Hence, we apply Locality-Sensitive Hashing (LSH) to generate candidate pairs from the user signatures.

LSH is useful for finding the highly-correlated signatures. LSH, introduced by Indyk and Motwani [15] proposes a basic idea to hash the signatures in a way such that the probability of having collisions is directly proportional to their similarities. For example, if two signatures are very similar to each other, they should have a much higher probability of having collisions than that of signatures which are very different from one another.

We first divide the compressed signature matrix into bands with multiple rows. To determine candidates for similarity, we apply a hash function for each column in the band. Hashing collisions indicate candidates for similarity. The number of rows in each band can be varied to alter the similarity thresholds. As we increase the number rows in a band, the number of candidate pairs decreases. This is because having a higher number of rows sets stricter rules for users to collide in the hash function, which typically produces few false negatives. On the other hand, if the number of rows in a band is small, it provides a lot less strict rules for bands to match, producing more false negatives. For our implementation, we used a band size of 4, producing 25 bands in the min-hashed signature of length 100.

## 4.4 Calculating Exact Similarity

Since Min-Hashing and Locality-Sensitive Hashing (LSH) merely output candidate pairs, we now need to test these candidates in order to find the exact clones. Keep in mind that the primary motivation for using Min-Hashing and LSH is that it is too computationally expensive to calculate the exact similarity values between every pair of users, for the naive approach would be $O(n^2)$. Along the same lines, we still wish to calculate the actual values for the remaining candidates as efficiently as we can, for there could potentially be thousands of candidates who survived from our original 176,786 users.

We first generated a File *interests* that stored each user's interest, where each interest name was stored as an encoded number to save space. This file was small enough to store in memory, so our efficient algorithm follows:

```
HashMap userInterests (user -> list of interests)

// store interests in memory from file
for each users
  userInterests.put(user, list of interests)

// calculates actual similarity for each pair
for each candidate-pair
  HashSet userA <- userInterests(user_1)
  HashSet userB <- userInterests(user_2)
  HashSet intersection = userA ^ userB
  HashSet union = userA U userB
  similarity = size(intersection)/size(union)
  print user_1,user_2,similarity,intersection
```

Note that we are also reporting the exact items on which each pair of users matched. We can then compare these shared items with those that are most popular across the entire network. Similarly, now that we know the actual similarity % (scaled between 0 and 1), we can evaluate the performance of our Min-Hashing and LSH algorithm. We also

found it worthwhile to calculate the similarity value between each of these users with their respective friends. This will put into perspective how special/similar the clone candidate actually is. For example, if user A has an average similarity value of .1 with his friends, but our algorithms found his clone candidate to have a value of .7, then we know that the result is pretty good and distinguished. Related, we can also easily find pairs of users who overlap within multiple categories. In other words, we can find users who match in both movies, music, and books, for example. This functionality returns even more accurate results of clones.

Moreover, with this extracted information, we could determine if popular shared items between clones are also the interests that are most shared globally. We could determine correlations between interests over various categories. For example, if a user likes movies *a, b, and c*, what books *x, y, and z* will he likely be interested in? This and other correlations are areas we would like to explore.

# 5. CONNECTIVITY

One of our primary motivations for finding clones is to try to find characteristics of these clones. Specifically, we find it to be interesting to think about in a real-life situation. For example, given a person in a city, if we could find his perfect match with regard to quantity of shared interests between the two, how distant are these two people? It is doubtful that they already know each other, but how close are they from knowing each other? To answer this question, we are explored the connectivity issue within our social graph, calculating the distance between each pair of clones. Note, that although we have collected all users from the Los Angeles network, some user profiles set privacy controls that would limit our overall calculations. For example, some profiles are set to private, or some set their listing of friends to be private, thus removing edges from our social graph. Plus, let it be stated that one's Facebook connectivity is clearly just a lower-bound on their actual connectivity in real life; a person will know more people in real life than is represented on any given social web site.

# 6. DATA ANALYSIS

After we extracted all of the listed 6 categories of interest from each of the 176,786 users, we were left with quite a few singleton items–items that were only listed by 1 user throughout the entire dataset. Keep in mind that users have free will to write anything they wish as their interests, and although Facebook's fields are well formatted, we naturally found some garbage interests. For example, a user would type items such as "pretty much anything other than country and techno" as one of their music interests. Similarly, another example is a user listing "ummmmmmmmm.... i dont know?!" as one of his favorite books. To combat this, we removed all items that appear only once. We believe that this will not remove legit information because it seems unrealistic that a person is so unique that he likes a particular item that none of the other 176,000 users like. The counts of the pruned, singleton-free interests for each category are shown in Table 1. Moreover, to shed light onto the amount of provided data, Table 2 shows the details of how many items were provided by the average user for each category.

## 6.1 Finding clones

**Table 1: Interests within the Los Angeles network**

| category | # of unique items |
|---|---|
| activities | 10,159 |
| interests | 20,508 |
| movies | 18,107 |
| books | 15,676 |
| music | 25,889 |
| tv | 8,724 |

**Table 2: Number of items listed by users**

| category | min | med | max |
|---|---|---|---|
| activities | 1 | 3 | 105 |
| interests | 1 | 4 | 129 |
| movies | 1 | 7 | 171 |
| books | 1 | 3 | 79 |
| music | 1 | 7 | 363 |
| tv | 1 | 5 | 98 |

Having pruned the singleton interests and users who consequently had 0 interests after pruning, we fed our data into Min-Hashing and LSH with a threshold of finding 80% similarity . Note that given $n$ unique users, there exists a maximum possibility of $\frac{n(n-1)}{2}$ unique pairs. In our case, $n = 176,786$, yielded us with a possibility of over 15 billion pairs. However, LSH's results yielded us with a relatively small, ideal number of candidates (see Table 3). We then used our aforementioned algorithm to calculate the exact similarity %s for every pair within these small candidate lists.

As seen in Figure 1 (the other three non-pictured categories possess similar characteristics to these that are pictured), LSH reported many high-similarity pairs that in actuality had a significantly lower similarity than predicted. However, this seemingly poor performance should not actually be viewed as poor results; it merely suggests that Min-Hash and LSH returned many false-positive candidates, but the true-positive results are not too bad. In fact, having users who match on 40-80% of their interests is pretty rare when one considers the average # of interests that each user list (Table 2). The totals # of false-positives relative to each category are in Table 3.

To further demonstrate the good performance of our finding clones, we compare users' average similarity values that they have (1) with their friends versus that (2) with their found clones (See Figure 2). It is apparent that the clone matches that we returned had significantly higher similarity

**Table 3: Performance of LSH**

| category | # $f_p$ | # candidates | % $f_p$ |
|---|---|---|---|
| activities | 54018 | 109448 | 49.3 |
| interests | 11828 | 90432 | 13.1 |
| movies | 349200 | 377422 | 92.5 |
| books | 190938 | 224820 | 84.9 |
| music | 368863 | 466884 | 79.0 |
| tv | 884936 | 910316 | 97.2 |

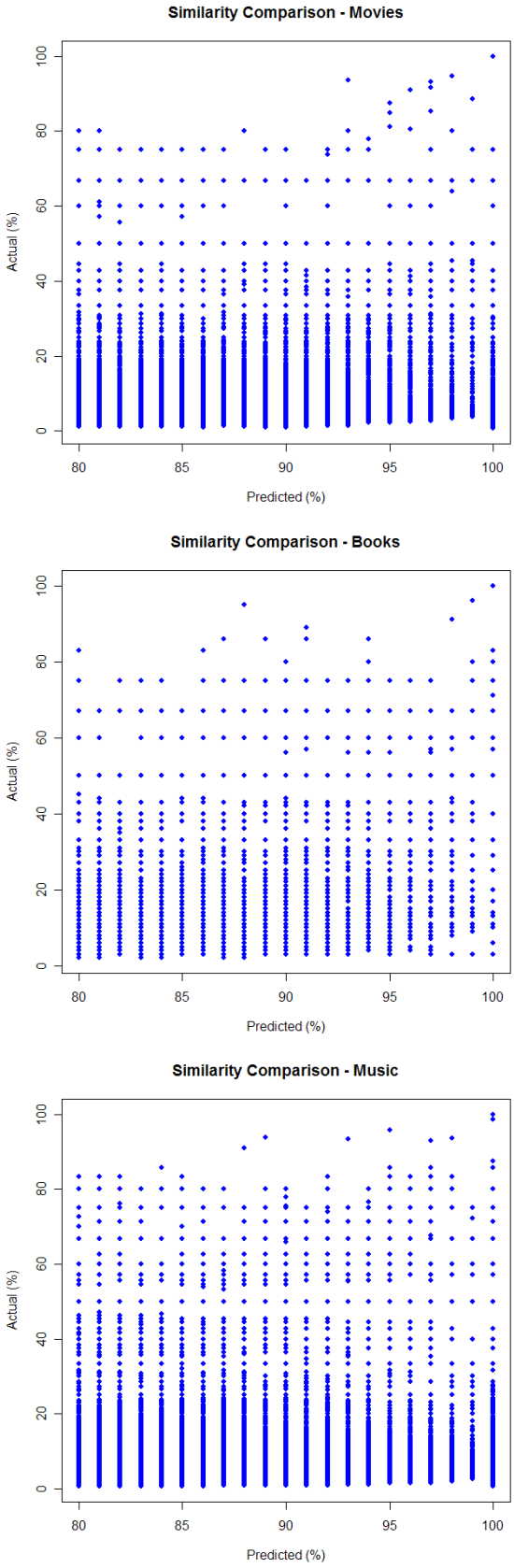



**Figure 1: Performance of predicting clones**

than they did with their friends. In addition, for the users who had a similarity value of 1.0 with both their clone and one of their friends, this was because these users only listed very few items. Naturally, it is much easier to match other users when you and your counterpart on have a few interests.

So, now we turn to understanding why this disparity exists between the expected and actual similarity values. First, we questioned if our implementation was correct. Having run many evaluations on smaller sample data, we concluded all was done correctly. After further understanding, we concluded that we get such a large number of false positives because our columns (user profiles) are very highly sparse. Using min-hashing we are essentially mapping a sparse vector of length 3-5 (the average # of interests per category) to a representation of length 100. This naturally lends the algorithms to have many hash collisions and generously report candidates. Alternatively, others have used a nearest neighbor approach with LSH, and achieved a small error with data as sparse as ours [16]. Similarly, using support vector machines have shown to be useful for social data mining [10]. Nevertheless, we biased our implementation to err on the side of yielding false-positive so that they can be easily filtered out in post-processing step.

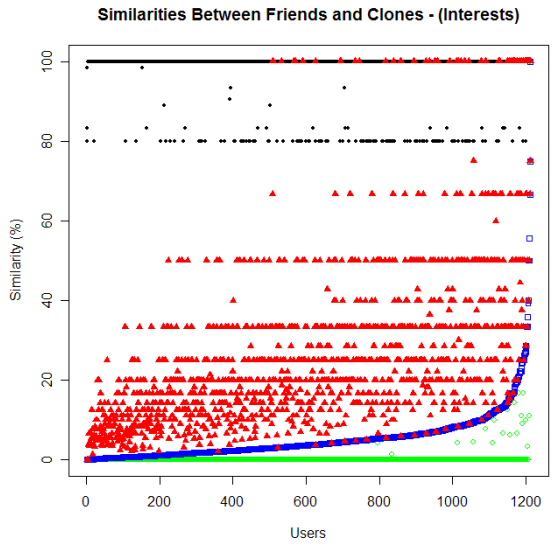
## 6.2 Dealing with Clones

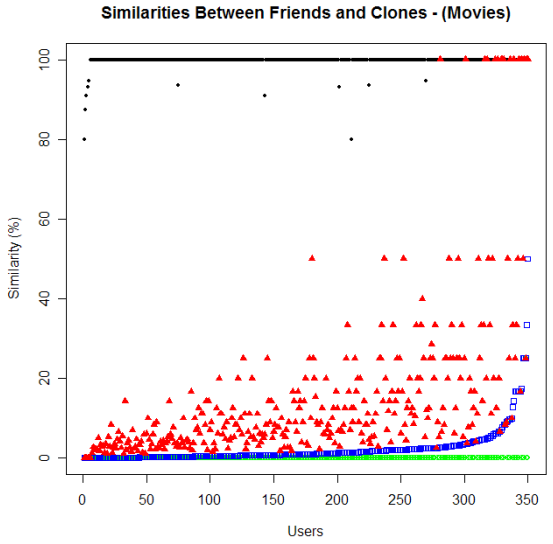
### *6.2.1 Finding Cross-Category Clones*

Now that we know which pairs of users are clones for each given category, we can further our notion of clones by considering which pairs of users are present as clones in other categories. For example, we can find all pairs of users who are clones in both categories $x$ and $y$, where $x, y \in \{$activities, interests, movies, books, music, t.v.$\}$. Since our number of resulting clones is relatively small, we could theoretically run a naive $O(n^2)$ comparison to find all duplicates within a pair of categories. However, our implementation followed the more efficient manner that was mentioned for our calculating actual similarity:

```
findSharedClones(categoryA, categoryB) {
  // store each file of candidate pairs
  HashSet clonesA <- clones from categoryA
  HashSet clonesB <- clones from categoryB
  HashSet allClones = userA U userB
  for eachPair in allClones
    if eachPair E {clonesA & clonesB}
      print eachPair
}
```

This may be useful for an individual, and it provides information about the overall trends. For example, if we look at the actual values that clones often shared with one another, we can see that many of them are actually amongst the most popular within the given category in which we are looking. This result makes sense, and it confirms our intuition that the more popular items are shared by more people and are consequently more receptive to matching other users' interests. In order to save space, we present only three of the six categories in Table 4. First, the reasoning for the clones having a higher # of occurrences than overall



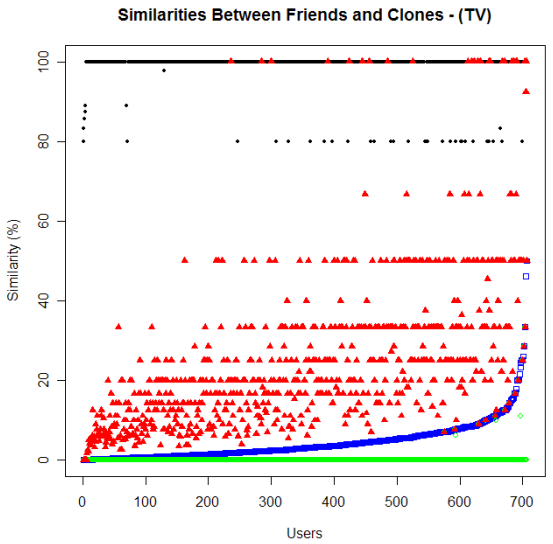




**Figure 2: Performance of predicting clones**

**Table 4: Most Popular Items Per Category**

Most Popular Music

| Overall | | Within Clones | |
|---|---|---|---|
| Item | # Occur | Item | Occur |
| the beatles | 4,267 | everything | 44,933 |
| radiohead | 4,048 | all kinds | 7,143 |
| coldplay | 3,923 | country | 4,681 |
| jack johnson | 3,533 | r&b | 4,302 |
| the killers | 3,383 | anything | 3,588 |

Most Popular TV

| Overall | | Within Clones | |
|---|---|---|---|
| Item | # Occur | Item | Occur |
| family guy | 10,049 | i dont watch tv | 2,016 |
| the office | 7,067 | the simpsons | 1,401 |
| greys anatomy | 6,888 | friends | 1,085 |
| lost | 6,773 | man vs wild | 916 |
| entourage | 6,525 | prison break | 750 |

Most Popular Interests

| Overall | | Within Clones | |
|---|---|---|---|
| Item | # Occur | Item | Occur |
| music | 10,944 | music | 33,547 |
| movies | 5,899 | movies | 4,546 |
| reading | 4,426 | you | 4,113 |
| traveling | 3,683 | everything | 3,659 |
| art | 3,132 | sports | 2,735 |

is because when we are considering clones, we are considering pairs of users. So, we could potentially have $\frac{n(n-1)}{2}$ pairs, instead of just $n$–yet, of course we are presented with not nearly this amount due to the pruned candidates that LSH produced. Second, we see that the most popular music and T.V. categories do not align too much. It's apparent that garbage input persisted towards the top, as many users were matched on meta-items such as "everything" and "all kinds." These phrases were also heavily prevalent within the global counts, but because of the large # of unique music terms, it because less likely that users would match on given items. This makes sense if you consider how variance, for a larger variance decreases the likelihood that users will share the same values. For example, if users had to choose a music interest from only five given choices, the garbage values would not persist. Yet, if there were 1 million possible open-ended items, users' interests would be more sparse and filled with esoteric values; thus garbage input would be more likely to be the shared items. Moreover, within the "Interests" chart, we can see that users have a tendency to enter sub-par data; the generic items of *music* and *movies* should not even belong within "Interests" because they each have their own separate category. Not visible, but a valid concern, is when users enter semantically similar items that have syntactical difference. For example, users may enter "running" and "runnin," which mean the same but appear differently. Thus, natural language processing issues arise also, but we did our best to combat the easy-to-catch cases.

Yet, it gives rise to the next topic.

### *6.2.2 Finding Related Categories*

**Table 5: Closely Related Categories**

| categories | % of shared clones |
|---|---|
| activities and movies | .1029 |
| books and music | .0819 |
| books and tv | .0186 |
| interests and tv | .0164 |
| activities and tv | .0149 |
| activities and interests | .0134 |

*findSharedClones*() provides us with a means to find the clone pairs that co-exist in two given categories. Yet, if we count the # of shared clone pairs for two given categories, and divide this value by the union, it will give us an indication towards how related the two categories are 6.2.2

```
calculateSimilarity() {
  forall combinations of categories
    HashSet clonesA <- clones from categoryA
    HashSet clonesB <- clones from categoryB
    HashSet intersection = clonesA ^ clonesB
    HashSet union = userA U userB
    similarity = size(intersection)/size(union)
    print similarity
}
```

Note that this is very similar to how we calculated actual similarity. And, of course, for exact calculations, it would be best to calculate the total # of users who share the same interests within both categories. However, we need not aim for this because (1) it is computational infeasible to look at all pairs of 176,000 users and (2) Min-Hash and LSH may have many false-positives, but it has not been shown that it has many false-negatives. Min-Hash and LSH typically has very good recall, and we assert that our low-threshold is sufficient for giving us all worthwhile candidates. We found that *activities* and *movies* were the closest related categories, closely followed by *books* and *music*. Table 5 lists the closely related categories.

### 6.2.3 Connectivity

Now that we know which users have clones, we look for ways to find patterns regarding connectivity. As hinted on, we are particularly curious about finding how distant two clones are from one another. In order to do this, we form a social graph in generic manner:

- Let vertices $V_1$ and $V_2$ represent two users. $V_1$ and $V_2$ are directly connected via edge $E$ if they are friends with each other
- Construct the connections $\forall\ V_i \in V$, where V = all users and $V_i$ represents a unique user

As suggested, our connectivity will be a lower-bound of the actual connectivity in real life, and also lower than what is present on Facebook. This is due to privacy controls that may sometime restrict us from seeing one's profile or listing of friends. Nevertheless, it's a pretty well-connected graph;


**Figure 3: Distribution of Degrees of Separation between Clones**

[3] showed that the graph of this dataset is comprised of one large strongly-connected component (SCC), and that there are numerous singletons and loners at the edge of the graph. Furthermore, it was shown that **everyone is connected within 4.788 connections,** and that there exists a standard deviation $\sigma$ of 1.103.

In order to determine the distance between any two given users, we implemented a form of depth-first iterative-deepening (DFID). DFID was chosen due to its space and time complexity; however, a slight modification was needed due to the possibility of having cycles–as friendship is transitive and we can easily encounter repeating users within our paths. Our results found that the **average degree of separation between clones was 4.244**–barely less than that of randomly chosen strangers. This maginal difference is almost negligible, although it does suggest that **clones have some slight tendency to be stronger connected than strangers.** The distribution of our results is shown in Figure 3.

## 7. FUTURE WORK

Our approach has encompassed a majority of the so-called dirty work in the sense that we have gathered all of the data and implemented adequate algorithms that allow us to find some basic patterns. However, there is a myriad of avenues and possibilities that we wish to explore:

Specifically, we have shown that Min-Hashing and LSH allows us to approximate clone candidates. But, a majority of these candidates evaluated to being false-positives. It would be interesting to explore other associative algorithms, such as using nearest neighbor mixed with LSH. Moreover, we could try a search-based technique were we naively start with the most popular users, asserting that they are more likely to have clones due to their having more friends than others. Rather, we could choose starting candidates based on a weighting scheme that not only includes their popularity, but logarithmically weights their value based on the

rareness of their interests. So, if the user has very cliche interests, that would cause him to be more heavily weighted–as we have seen that popular interests are often the shared interests between users, except when the category has a very wide spread range of values.

Additionally, we have shown the correlation of interests between categories. For example, users who have similar activities also share similar movies–suggesting that these categories are related and good indicators of a person's interest. Yet, we could expand this idea and consider individual interests. With this, we could find relationships between what interests are related to others. If 75% of users who like "The Beatles" also like "Led Zeppelin," then we have a good indication that the remaining 25% of Beatles fans would like Led Zeppelin if they were exposed to the music. This recommendation-type system would serve as a very nice fit into the current flood of online marketing and recommendation frenzy. It would also be highly valuable from a sales perspective. In order to implement it, we could naively calculate percentages, or we could implement some form of Bayesian network or Markov Chain with weights based on characteristics of the users and their interests.

Regarding the concept of connectivity, we could advance this idea by using it as a metric for determining (1) where to start looking for one's clones (i.e., to start looking from our friends, or from a random stranger) and (2) studying the paths that exists between two user clones for the sake of seeing if other characteristics other than interests are shared between them. For example, it might be seen that two clones have many mutual friends, and that many of them share similar items also, or that the direct path between the two clones all went to the same secondary school. Interesting data findings like this, and especially the approaches to mining this data, presents many interesting challenges.

# 8. CONCLUSIONS

Throughout our efforts to find clones and the similarities that arise, we have encountered a few findings. One of the first lessons we learned is that it is highly difficult to capture an accurate model of an entire social network. In order to capture all user pages within the second-largest city within the United States is a gigantic, tedious task that lends one to battle privacy issues, server and connection issues, and terms of service agreements. Even once one has this data, it is difficult to manage, for over 200,000 files is cumbersome to move, navigate, and work with.

As for more research-based findings and conclusions, we experienced and learned about the limitations of Min-Hashing and the LSH algorithms. We conclude that although it does a sound job of providing us with some good candidates, there might be numerous false-negatives within the pruned set. It is probably more appropriate to try numerous associations-finding or approximation algorithms. Nevertheless, we have shown that clones do exist, and that they possess an amount of shared similarities way beyond that of what each party typically possesses with his immediate friends. Similarly, we have noted that the clone pairs seem to have no strong relation with regard to being closely connected. In fact, it appears that clone pairs are merely random strangers who are highly similar in their listed interests. We believe that having a clone-finder feature on social networks would be a worthwhile, useful tool, especially because it is rather impossible for a user to blindly find his matched counterpart amidst the mass of strangers.

We have found that certain categories of interest are apparently more parallel-correlated than others; particularly, activities and movies are related, along with books and music, are highly related in the sense that users who have similar taste in one category will have similar taste in the other. Last, we conclude that the popularity/frequency of items that are often shared between clones are directly related to their popularity across users within the global network. Yet, categories that have huge variance and a wide-range of values lend itself to be susceptible to more commonly shared noisy inputs. These findings are worthwhile to note, but we have only begun to investigate the numerous opportunities that lie within the highly data-enriching field of social networks. We anticipate further development within clone-finding, along with the subsequent patterns that can be found therein.

# 9. ACKNOWLEDGMENTS

Our research efforts would not have been made possible without the help of others. First, we would like to thank Dr. Stott Parker, as he was our UCLA professor for "Special Topics in Data Structures" (a.k.a. Data Mining) during the time we conducted our research. It was in this class that we learned various data mining techniques, some of which would now be interesting to apply towards our work. Second, we would like to thank colleagues Keenahn Jung and Tim Coulter for helping with our obtaining our dataset of all Los Angeles network users of Facebook.

# 10. REFERENCES

[1] Myspace is the number one website in the united states according to hitwise. HitWise Press Release, July, 11, 2006 http://www.hitwise.com/press-center/hitwiseHS2004/social-networking-june-2006.php.

[2] Alexa - gloabl top 500 sites. October 30, 2007 http://www.alexa.com/site/ds/top500.php.

[3] C. Tanner,C. Hsieh,K. Jung. Understanding Pure Social Networks: Structure and Connectivity. *Unpublished. Submitted for "Web Information Management" course at UCLA*, 2007.

[4] Myspace hits 100 million accounts. August 9, 2006 http://mashable.com/2006/08/09/myspace-hits-100-million-accounts/.

[5] Myspace reaches 190 million users. July 2007 http://www.myspace.com/.

[6] D. J. Watts. Six degrees: The science of a connected age. *W. W. Norton*, 2003.

[7] S. Milgram. The small world problem. *Psychology Today*, 1967.

[8] M. Jamali,H. Abolhassani Different Aspects of Social Network Analysis. *Web Intelligence, IEEE/WIC/ACM International Conference*, 18-22 Dec. 2006 Page(s):66 - 72

[9] S. Staab,P. Domingos,P. Mike,J. Golbeck,et.al. Social networks applied *Intelligent Systems, IEEE*, Volume 20, Issue 1, Jan-Feb 2005 Page(s):80 - 93

[10] Makrehchi, M.; Kamel, M.S. Learning Social Networks

from Web Documents Using Support Vector Classifiers *Web Intelligence, 2006. WI 2006.*, IEEE/WIC/ACM International Conference on 18-22 Dec. 2006 Page(s):88 - 94

[11] Golbeck, Jennifer Trust and Nuanced Profile Similarity in Online Social Networks *MINDSWAP Technical Report TR-MS1284*

[12] T. Fenner, M. Levene, G. Loizou, and et al. A stochastic evolutionary growth model for social networks. *Computer Networks*, 2007.

[13] Liu, Hugo Social network profiles as taste performances *Journal of Computer-Mediated Communication* Vol. 13, 2007

[14] Spertus, E., Sahami,M. and Buyukkokten, Orkut Evaluating similarity measures: a large-scale study in the orkut social network *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* p. 678–684, 2005

[15] Cohen, E. and Datar, M. and Fujiwara, S., et. al. Finding interesting associations without support pruning *Knowledge and Data Engineering, IEEE Transactions on* pages 64-78, 2001

[16] Aristides Gionis and Piotr Indyk and Rajeev Motwani Similarity Search in very high dimensions via hashing *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*