Connecting the World:
Building a social network recommation system for both local and global connectedness

Abraham Botros
SUNet ID: abotros
abotros@stanford.edu
Stanford University, Fall 2015 - CS224W Final Project Report

1 Introduction

We all share the same home - this pale blue dot we call Earth. Yet we allow ourselves to be thoroughly divided by things such as cultures, skin color, and invisible lines on the ground. How can we nurture new relationships, new understanding, and new perspectives in this disconnected world? As one possible piece of the solution, with the rise of online social networks, we are able to instantly connect with people anywhere around the world - no longer just people only in our city, state, or even country. However, such connections are relatively few and far between, even given that the infrastructure is certainly there. Thus, to improve on this, here we build a foundation for a social network recommendation system that can encourage diverse, distant, and yet relevant relationships between users across the globe, in addition to our familiar nearby neighbors and friends. Our system leverages properties and tools of/for social graphs, user geolocation information, and machine learning to provide meaningful friend/follower/contact suggestions to users, with some desirable balance between more local/typical recommendations and more distant/global suggestions.

In doing all this, we hope that such connections can foster novel and otherwise unlikely and impractical communication and information exchange, which in turn can nudge us to move forward in our ability to comprehend, accept, and grow with people from different locations, backgrounds, and cultures across the world. We hope that related future work can provide some contribution, even if minimal, to creating truly global (as well as local) connectedness between people across the world, facilitating people to realize how different yet the same we all are on this single spaceship we call home.

2 Related work

2.1 YouTube recommendation system

[6] discusses the video recommendation system behind YouTube circa 2010. The goal of the system in general is to provide personalized recommendations that help users find high-quality videos relevant to their interests, even given the extremely large amount of video data possessed by YouTube; these recommendations are updated regularly and are based on recent user activity on the site. Some complications involve: no or very poor metadata for a video at upload time; videos are usually relatively short (under 10 minutes in length), leading to relatively short and noisy user interaction data; and videos can often go viral quite quickly, requiring constant renewal of recommendations to users.

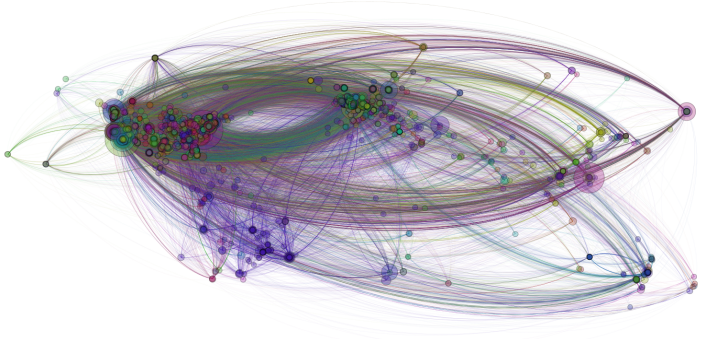

Figure 1: Visualization using Gephi of network containing users with locations obtained as in Section 3.1. This is quite representative of the visualization of the corresponding network for only ego users, which is therefore not shown here.

In order to accomplish efficient and effective video recommendation given the environment and the challenges described above, the system uses a user's personal activity (watched, favorited, liked videos) as seeds and expands on this set by traversing a co-visitation-based graph of videos. A co-visitation count for each pair of videos captures how often they were co-watched within any user's session in (usually) the last 24 hours; this gives a relatedness score that can be used as weights for edges in a directed graph over the set of all videos that can then be searched. Ranking then occurs in three stages: (1) video quality (global view count, overall rating/favoriting/commenting/etc), (2) user specificity (consider properties of the seed video in the user's history to boost videos that are closely matched with a user's unique taste and preferences), and (3) diversification (videos that are too similar in the recommended set

are removed to further increase diversity).

Evaluation of performance using live A/B testing showed this approach gave a 200% click-through rate compared to baseline approaches, showing this system is successful for YouTube video recommendation.

2.2 Twittomender recommendation system for Twitter

In [7], the authors create a recommender system to suggest followees for users using the Twitter API. There are two main modes for this tool: (1) "user search", where the user provides query terms to receive a ranked list of relevant Twitter users, and (2) "user recommendation", where the user's own Twitter profile acts as a query to generate recommended users to follow.

First, a target user is profiled by collating their recent tweets, their followees and followers, and their followees' and followers' recent tweets. Users are then represented using weighted term-vectors using TF-IDF weighting by parsing these tweets or user ID lists accordingly.

Offline tests using the various sources (and also hybrids/combinations) over data imported from 20,000 Twitter users were measured based on their "relevant recommendations" - for each target profile, we count how many of the recommendations are in the user's known followees list (ground-truth; in other words, how often the recommender suggests people that the target user is known to have followed). For various output recommendation list sizes (top 5, top 20, etc.), performance on relevant recommendations is measured via average precision (average percentage overlap between given recommendation list and target user's actual followees-list) and average relevant recommendation position (the position of relevant/verifiable recommendations in the output recommendation list). Results across different input sources are promising, and this is verified in a live trial with real users using both the "user search" method and the "user recommendation" method on their own profiles and picking which of the recommended users they would actually want to follow.

2.3 Geographic routing in social networks

[8] discusses coming up with improved models for routing across geography and distances in social networks (using "rank-based friendship"). The authors use a subset of the LiveJournal blogging social network, consisting of around 500,000 users with known city locations in the U.S. and whose friends (and their respective locations, too) we can readily see. A successful initial routing simulation is done using the geographically-greedy routing algorithm "GEOGREEDY" (if person u currently holds the message and wants to eventually reach a target t, then she considers her set of friends and chooses as the next step in the chain the friend in the set who is geographically closest to t), which yielded a success rate of 13% completion of the chain with an average length of just over four.

The authors then explore mathematically modeling the relation between geographic distance and friendship in the network. For each distance δ, let $P(\delta)$ denote the proportion of pairs u, v separated by distnace $d(u, v) = \delta$ who are friends. As we expect, as δ increases, $P(\delta)$ decreases, indicating that geographic proximity indeed increases the probability of friendship. At distances greater than 1000km, the δ-vs-$P(\delta)$ curve flattens, indicating some background probability ϵ of friendship independent of geography (such as meeting online through a shared interest, not geography-based meetings like in the workplace).

However, the authors point out that previous theoretical results contradict their finding in the LiveJournal data that the probability $f[d(u, v)]$ of geographic friendship is roughly proportional to $1/d(u, v)^2$ (which explains the finding of short paths by GEOGREEDY). To reconcile this, the authors show that previous theoretical results based solely on distance between people is insufficient to explain the geographic nature of friendships. Instead, they propose a new model that uses rank - when examining a friend v of u, the relevant quantity (rank) is the number of people who live closer to u than v does, such that: $rank_u(v) := |\{w : d(u, w) < d(u, v)\}|$. Then, under the rank-based friendship model, the probability that u and v are geographic friends is proportional to the inverse of $rank_u(v)$. Such a model is especially helpful for explaining data that involves message-passing across geographies with non-uniform populations (such as from the East Coast, through the more sparse Midwest, to the West Coast). This model is shown to provide a better fit to the LiveJournal network (with an approximately inverse linear fit from $rank_v(u)$ to the probability that u is a friend of v) than the slopes given by previous theoretical models, thus suggesting this is a viable model for more accurately explaining the relationship between geography and friendship in real-world networks.

2.4 Differences from current work

The primary goal of the current work is to apply methods related to those discussed in these previous publications in order to produce quality recommendations over a social network, but with the added feature of considering geographic location and distances between users when selecting recommendations. The YouTube and Twitter recommendation systems above do not contain any mention of explicit handling of geographic loca-

tion/distance of content. The geographic routing study certainly does include geographic location/distance in its work, but deals with probabilities of friendships as a function of geographic distance, rather than full recommendation over user-specific features, network features, and geographic distance. Thus, in the current work, we connect all of these points, developing a social recommendation system that, in addition to user-specific and network features, considers geographic distances between users as a central component when producing new user recommendations.

3 Technical Details

3.1 Data

We will use a subset of the Twitter social network for the base for our work. User IDs and directed "followee" edges (an edge from A to B means A follows B) were obtained from [4], which contains ego networks for 973 Twitter users, with a full dataset of more than 81,000 nodes and almost 1.8 million edges. However, due to the extremely low similarity between random users in the given features in this dataset, the features provided were unusable for our purposes. On average, users had 13.66 hashtags associated with their profile (hashtags they had tweeted recently on the social media network); however, already-connected users shared only about 1.7 hashtag features in common on average, and random users shared a negligible 0.03 hashtag features on average. Given these statistics for feature vectors describing user profiles, recommendation would be quite impractical; recommendation likely would fail every time if we have essentially zero features in common between almost all random users. As a result, we only are able to use the user IDs and followee edges from this dataset, and no other data.

From here, we extract location data from user profiles (Twitter users have a free-form location entry on their profile they can optionally fill out). We discard users from our network who either have no location entered, or who have locations that we are unable to automatically process using the Google Maps API [1]. For users who do have a location entered, but only an approximate one (this is the typical case, with U.S. users often listing their city or state, for example), we obtain a bounded area from the Google Maps API representing the total area corresponding to a location query; we then randomly and uniformly select an exact latitude-longitude coordinate within the bounded area to represent the user's exact location. This randomness will allow us to compare locations for nearby users without making the impractical assumption that they are at exactly zero distance from one another (if two users both list "Stanford, CA", without any random injection, they will be given the exact same single latitude-longitude coordinate tuple representing the center of this area).

A visualization of all users with locations after these steps is shown in Figure 1. In total, we ended up with 56,690 total users that still have valid public accounts on the network and that also include a valid location on their profile, with 989,463 edges among them. This included a total of 746 ego users, which are the only users in our set where we have full in-bound/out-bound edge information and are therefore the only users we will be focusing on for recommendation (all other users can be recommended *to* these ego users, but we will not recommend *for* these other non-ego users).

It is also worth noting that unfortunately simply processing all the user profiles and collecting the location data ended up consuming a non-trivial portion of the time spent on this project, from setting up interfaces with the API's to dealing with imposing rate limiters (especially Twitter's, which prevented us from freely obtaining more information/users/edges in reasonable time). In addition, the author was unable to find any better datasets (such as ones already containing locations, tweets, a full graph instead of ego networks, etc.) due to Twitter's recent limitations and restrictions on publicly-available Twitter datasets containing tweets (see [3] as an example). As a result, we were forced to download all relevant information (aside from the IDs and followee edges for the given users given in [4]) manually, step by step.

3.2 Problem details

Given a subset of the Twitter graph (certain users with locations represented as nodes, and following represented as edges), and given the data we mention (each user has an associated numerical ID and location), our approach can be outlined as follows:

- Computing geographic distances from every ego user and all other users with locations; see Section 3.3.
- Constructing vectors for each user in the graph describing that individual's network-based features; see Section 3.4.
- Generation of vectors representing topic-interest probabilities for each user; see Section 3.5.
- For each ego user, construct a ranking/scoring across users they are not already following based on network-based features, topic-interest features, and location features; see Section 3.6.
- Use this ranked list of scores to to propose a small set of recommendations specific for each ego user; see Section 3.7.

3.3 Geographic location and distance

To begin, we want to compute the geographic distances between users. Since we will be focusing on recommending only for our ego user subset, we really only need to compute the distances from each ego user to all other users with locations. Thus, for each ego user u_e, we have a precomputed map from every other valid-location user to the current ego user u_e. This map is stored and used for quick look-up as described in Section 3.6. To compute the geographic distance between two points in latitude-longitude space, we use the Haversine formula [2]. We then normalize these distances to the range [0,1], where 0 corresponds to the minimum Haversine distance we observe from u_e to any other user, and 1 corresponds to the maximum we observe. Thus, we can think of a different user having a normalized distance of 0 to u_e as being essentially at the same location as u_e; a normalized distance of 1 means the different user is as far away as possible - on the opposite side of the world. We denote the normalized distance between an ego user u_e and another user v as $D_{u_e,v}$.

3.4 Network features

We use 6 network features to describe any given user u:

- In-degree; here, the number of users following user u.
- Out-degree; here, the number of users u follows, or the size of their set of followees.
- Degree; the total number of users u follows and is followed by.
- Degree centrality; computed as degree of a user divided by $(N-1)$, where N is the total number of users with locations in the dataset.
- Closeness centrality; computed as the reciprocal of the sum of distances from a user to all other users via shortest paths.
- PageRank; computed according to the now-common PageRank algorithm, which iteratively estimates the importance of a user based on the number and importance of the users connected to that user.

All these features are computed for each user with valid location; these features are computed separately from any other component in our system's pipeline. Each feature is independently normalized across all users (each of the 6 features is normalized only with regards to itself, but across all users), such that the normalized in-degree for a given user is in the range [0,1], with 0 and 1 corresponding to the minimum and maximum in-degrees observed in the network across all users. We denote the vector of these 6 network features for user u as $f_{\text{net},u}$, where F represents all features, and we are specifically interested in the *net*work features for user u.

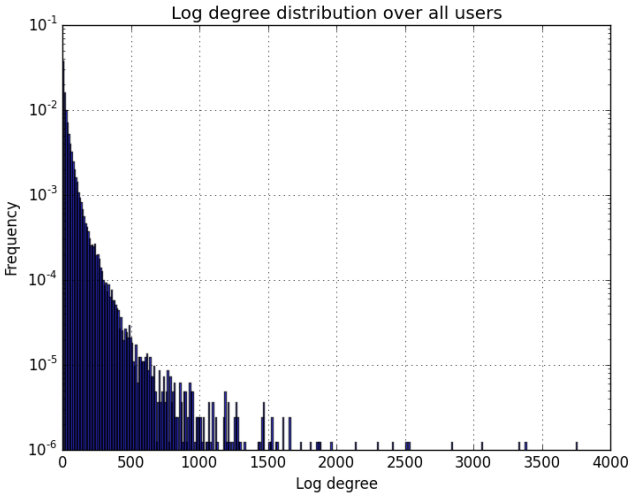

Figure 2: Log degree distribution for full network. Note the log axis and heavy skew.

As a visualization, see Figure 2 for a histogram of the (log) degree distributions (across all users). As we can see, the degree distribution is extremely skewed; almost all users have a relatively small degree (1-500), and there are only a handful of users with very large degrees in comparison.

3.5 Topic-interest generation

Ideally, we would download numerous (recent) tweets from each user through the Twitter API, then use some natural language processing techniques (such as keyword detection/similarity, sentiment detection, or TF-IDF as in [7]) and any other related techniques at our dispoal to discover what topics interest each particular user; we could then compute similarity between any pair of users given these descriptive interests. However, this proved impractical for the purposes of this project, primarily due to time constraints. Downloading a large number of tweets would have been time-prohibitive given the harsh rate limiting in the Twitter API for timeline/tweet scraping and the large number of users we would need to do this for; in addition, topic detection for each user and semantic similarity between users on this dataset would be large natural language undertakings even on their own, and thus are certainly out of the scope of this project. Despite this, we still want to have some notion of topic-interest description for each user. This will give us a notion of similarity of interests between users, and therefore similarity between users and likelihood that a recommendation/friendship would be fitting.

To accomplish this, we generate synthetic interest vectors to describe each user using 10 latent interest categories (one can imagine that a corresponding vector in

practice for the Twitter graph might correspond to topics such as science, technology, sports, fashion, news, business, etc.). We first start with the ego users, which we can think of as the "centers" or "roots" of their respective ego networks. For each ego user, we generate a vector of 10 random probabilities (each component in the vector is in the range [0,1]), representing the probabilities that the ego user is interested in each topic (a natural representation would be that a value of 0 means the user is certainly not interested in that topic, 0.5 means we are unsure, and 1 means we are certain they are interested in the corresponding topic). For a given user u, denote their interest vector I_u.

Then, given the features generated for our ego users, we iteratively assign interest vectors to each of the users the ego is following (given we our dataset is based on ego networks, if we do this for every ego user, we will reach all possible users in the network). To do this, we have to make some relatively arbitrary decisions; in particular, we assert for the purposes of these assignments that, on average, users follow other users due to similarity in topic interests about 75% of the time. Since we want to assign an interest vector I_v to a new user v followed by the ego u, we loosely propagate u's interests I_u to v; v is assigned a vector of probabilities sampled from a multivariate Gaussian distribution centered on I_u, with some moderate amount of noise (we used a diagonal matrix of 0.1 values in practice). The remaining 25% of the time, we assign a followee of u an entirely random vector of 10 probabilities, not based on I_u.[1] An example of possible interest vectors is shown in Table 1.

User	Topic-interest vector, I
u	[0.81, 0.04, 0.36, 0.99, 0.43, 0.08, 0.93, 0.49, 0.53, 0.05]
v	[0.67, 0.13, 0.35, 0.97, 0.29, 0.00, 1.00, 0.47, 0.76, 0.00]
w	[0.79, 0.00, 0.51, 1.00, 0.47, 0.38, 0.95, 0.65, 0.80, 0.00]
x	[0.65, 0.77, 0.11, 0.37, 0.88, 0.53, 0.56, 0.20, 0.30, 0.44]

Table 1: Example topic-interest vectors for four users; u is an ego node, and therefore has its vector I_u generated first randomly. v and w are two other users that u follows, and that happen to fall into the 75% case where we generate interest vectors based on the ego's interests; we can see that I_v and I_w have vectors relatively similar to I_u. On the other hand, u also follows x but x falls into the 25% case where we randomly generate an entirely new interest vector; we can see that I_x is relatively different from I_u.

3.6 Scoring

Given all the relevant features (geographic distance, network features, and topic-interest vectors, as described in the previous few sections), we can now compute a score between an ego user and all other users that encompasses weightings of all of these features into a single compositie scalar score. This will allow us to rank users for recommendation to the ego based on these composite scores.

This can be outlined as follows:

1. For each ego user u, we:
   (a) For each other user $v \in V$, where V is the set of all users that have a location and that u does not already follow:
      i. Look-up the precomputed Haversine distance between u and v, $D_{u,v}$.
      ii. Look-up the precomputed network features for v.
      iii. Compute the L2-norm between I_v and I_u. Represent this as $\|I_v, I_u\|$.[2]
      iv. Compute the average L2-norm between I_v and all other users that u already follows. This tells us how similar/different I_v is to the interests of all the users we know u already follows. Represent this as $\overline{\|I_v, I_{\text{followees of } u}\|}$.

As a side note, we mention that, in practice, we actually have to limit the number of other users in V; V is too large otherwise, especially since we theoretically want to iterate through all $v \in V$ for each ego user u. This quickly becomes intractable. To alleviate this, in practice, we randomly sample 2,000 users v from V, instead of iterating over the entire space of V.

Combining all of these features gives us a vector of length 9 (1 for distance, 6 for network features, 1 for direct L2-norm, 1 for averaged L2-norm). We will refer to this full feature vector as $F_{u,v}$, which is specific to each pair u (the ego user) and v (the user being scored). We can explicitly write this out as:

$$F_{u,v} = \Big[D_{u,v}, F_{\text{net},1}, F_{\text{net},2}, F_{\text{net},3}, F_{\text{net},4}, F_{\text{net},5}, F_{\text{net},6}, \|I_v, I_u\|, \overline{\|I_v, I_{\text{followees of } u}\|}\Big] \quad (1)$$

We can then take a dot product/weighted linear combination of these features with a vector of weights, where we can represent our weights as:

$$W = \Big[w_D, w_{N1}, w_{N2}, w_{N3}, w_{N4}, w_{N5}, w_{N6}, w_{I,\text{direct}}, w_{I,\text{avg}}\Big] \quad (2)$$

[1] We note that these assumptions are indeed assumptions, and future work could be done to develop proper models of interest similarity between users, or the semantic analysis of actual profiles could be done as mentioned above.

[2] When computing L2 norms, we note that the larger the result, the greater the *difference* between the two vectors; when computing this for two topic-interest vectors, a large result means the corresponding users have very different topic interests and little similarity.

Where the $w_{N\cdot}$ terms are the corresponding weights for the 6 (ordered) network features described in Section 3.4; $w_{I,\text{direct}}$ is the weight term for the direct L2-norm between I_v and I_u; and $w_{I,\text{avg}}$ is the weight term for the averaged L2-norm between I_v and all the interests of the users u already follows.

w_D is a special weighting term for the distance between two users. In particular, as mentioned in our problem outline, we want to have essentially two "modes": one mode where we recommend users that are similar in interests and locations and that have strong network features (call this the "distance-penalizing" mode), and another mode where we do everything the same but want to encourage larger distances (the "distance-rewarding" mode). Based on which mode we are in, we set w_D differently to either demote or promote proportional to distance between possible recommended users and the ego user we want to recommend for. Specifically, if we set w_D to a larger negative constant, we can penalize large distances heavily; if we set w_D to a larger positive constant, we instead reward large distances.[3]

Given the (unchanging) weighting vector W and our actual feature vector $F_{u,v}$, our weighted linear combination score for ego user u and v is given simply as the scalar score, $S_{u,v}$:

$$S_{u,v} = W \cdot F_{u,v} \tag{3}$$

3.7 Recommending

For a given ego user u, if we have scores $S_{u,v}$ for all possible $v \in V$, we can easily rank these scores by sorting them in descending manner. The first entry in our list will represent some user $v^* \in V$ for which $S_{u,v} \leq S_{u,v^*}\ \forall v \in V$ (i.e., v^* received the best score); the second entry will have the second-highest score, etc.

Remembering that we have two "modes" as in Section 3.6:

- For the "distance-penalizing" mode, we compute scores over all $v \in V$ after setting our distance weight term w_D to a negative value to discourage larger distances.
- For the "distance-rewarding" mode, we do the same for all $v \in V$, but with a positive w_D value to encourage larger distances.

We keep a separate scoring/recommendation list for each mode, and sort each of these individually after getting all the appropriate scores. We then take the top users from each list separately and use these as our final recommended user set to the ego user u. Call the set of recommended users for ego user u in the distance-penalizing mode $R_{u,dp}$; call the set of recommended users in the distance-rewarding mode $R_{u,dr}$. Call the full set of recommended users R_u.

We have to explicitly hard-code the number of "distance-rewarding" users we put into our final recommendation list; we desire to make 20 recommendations for each ego user, and want 20% of those (4 of them) to be for longer-distance users from our "distance-rewarding" recommendations. If we want a total of $|R_u| = 20$ recommendations, and want $|R_{u,dp}| = 16$ distance-penalizing recommendations and $|R_{u,dr}| = 4$ distance-rewarding recommendations, we simply take these corresponding numbers of users from the top of their appropriate scoring lists mentioned above.

3.8 Recommendation evaluation

Finally, we want to evaluate our recommendation results $R_{u,dp}$ and $R_{u,dr}$ for a given ego user u (we can then repeat this over all ego users). To do so, we look at a few specific values:

- Average of L2-norms between all $r \in R_u$ and u. Normalized by the minimum and maximum distances between u and all possible $v \in V$ we encountered in the previous steps.
- Average of L2-norms between each $r \in R_u$ and all of the users that the ego already follows. Normalized by the minimum and maximum distances between u and all possible $v \in V$ we encountered in the previous steps.
- Average of each of the 6 network features across all $r \in R_u$. Note that since the network features are already normalized, we do not need to normalize again here.
- Average distance between u and all the users it already followed, versus the average distance between u and $r \in R_{u,dp}$, versus the average distance between u and $r \in R_{u,dr}$. Note that since the distances are already normalized, we do not need to normalize again here.
- Number of "long-distance recommendations", which is the number of recommended users $r \in R_u$ where the distance between u and r is greater or equal to the average distance between u and all the users it already followed.

Overall, these metrics give us a fairly holistic view of how our recommendations did: how similar are their interests to the ego user and to the users the ego already follows; how strong are their network features;

[3] In practice, we used the following weights: $w_D = -2.5$ when penalizing distances, $w_D = +2.5$ when rewarding distances; $w_{N1} = 2.0$ for in-degree weighting, $w_{N2} = 0.5$ for out-degree weighting, $w_{N3} = 1.0$ for degree weighting; $w_{N4} = w_{N5} = w_{N6} = 1.0$ for centrality/PageRank weighting.

how well did we recommend closer users when encouraging smaller distances; and how well did we recommend farther users when encouraging farther distances.

3.9 Implementation

All work was implemented in Python, with the help of the Snap.py Python library [5] for network node/edge representation, and a Haversine-computing library [2] for computing geographic distances between latitude-longitude locations.

4 Results

Tables 2, 3, and 4 show summary statistics for our entire run over all ego users (and using only 2,000 random $v \in V$ possible recommendation candidates when scoring for a particular ego user u). As a result, to clarify, the numbers shown are averages (over all ego users) of averages (over all recommendations for a specific ego user).

See Figure 3 for a visualization of the distribution of distances (unnormalized) from ego users to their previously-existing followees in the network.

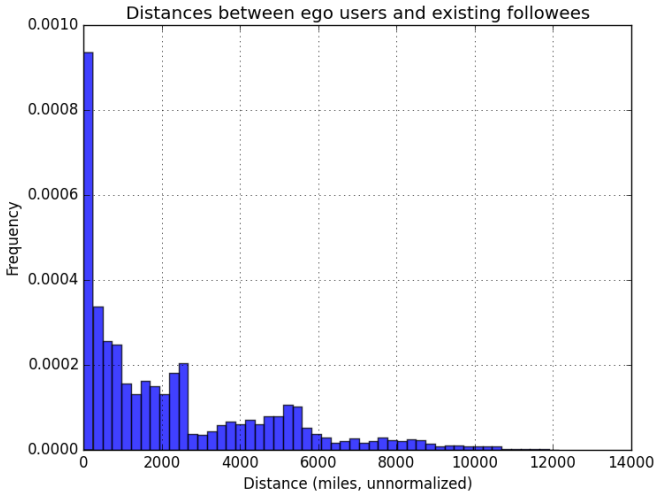

Figure 3: Visualization of histogram of distances of all ego users to their existing followees already present in the network (without any recommendation). Distances are given in miles.

5 Conclusion

5.1 Analysis

Recalling that smaller L2-norms between topic-interest vectors correlate to higher similarity, Table 2 shows that we do relatively well on average for recommending similar new users to an ego user based on topic-interest vectors; since the numbers are normalized, 0.23345 on the range [0,1] is relatively good. Unfortunately, we do not do quite as well for recommending similar-interest users compared to the ego's followees, but we expect larger variance over the set of interests of ego followees anyway, especially since approximatley 25% of them had interest vectors that were generated randomly and not based on the ego.

Table 3 shows our performance on average for our recommended users in terms of their network features. In particular, while the numbers seem low at first glance, we must remember that (1) we are using normalized-averaged values on the range [0,1], and (2) we have a very skewed distribution in degree (and as a result, we can imagine this largely skews all centrality and PageRank metrics), as we saw in Figure 2. If we look at the normalized average degree (0.01180), for example, we see that this corresponds to an unnormalized degree of 44.938, which is on the upper end of the bulk of nodes we see in the degree distribution. We then must consider that we had to sample only 2,000 of approximately 57,000 candidate recommendation users for each ego user for our computations to be tractable given the size of our network; given this, it is very likely that an unnormalized degree of around 40-50 is very much on the upper end of the possible users we sampled that we could have had the chance of even selecting as a recommendation. Future work would need to be done on breaking down the three centrality scores (degree centrality, closeness centrality, and PageRank), and comparing them explicitly to the minimum and maximum values on the network for these metrics.

Finally, Table 4 shows the overall location-centric results that are critical to our problem. We see that this actually worked very well! In particular, beforehand, our assumptions were confirmed - ego users on average followed other users that were relatively close to them (0.27493 on the range [0,1], in particular). Figure 3 further confirms this, as we see that an overwhelming majority of the existing followees for our ego users are relatively close in location to them (shown in unnormalized distance in the figure). When we penalize larger distances from the ego user to recommended users, we see that we got an average normalized distance of 0.07199, which is quite low (and what we wanted and expected); however, future work could be done on tweaking the hyperparameters to penalize larger distances less extremely when in the distance-penalizing mode, as this might allow us to have more flexibility in choosing users with better topic-interest similarity and stronger network features. That said, when we instead *reward* larger distances, which is one of our primary goals in this project, we see we indeed get larger distances (0.79342 on average on the range [0,1], so relatively long-distance users on average); this is exactly what we want. Lastly, we note that our average count of "long-distance" recommendations as we described in Section 3.8 is exactly

Table 2: Summary statistics - topic-interest similarity. Data has already been normalized for each ego user as in Sections 3.6 and 3.7, and averaged across ego users.

Topic-interest L2-norm, ego	Topic-interest L2-norm, ego followees
0.23345	0.54832

Table 3: Summary statistics - network features. Data has already been normalized for each ego user and network feature as in Section 3.4. **Note that a normalized degree of 0.01180 corresponds to approximately an unnormalized degree of 44.938**.

In-degree	Out-degree	Degree	Degree centrality	Closeness centrality	PageRank
0.00619	0.01943	0.01180	0.00987	0.37263	0.00107

what we want, too - on average, we want to recommend 4 out of 20 users that are larger distances away from the ego user compared to their existing followees, and our average count of 4.24531 is almost exactly spot-on. Some noise is likely introduced by users on the border in one of the two modes (distance-penalizing or distance-rewarding modes); for example, a user who is very similar in interests and has very strong network features but a medium-to-lower (but not trivial) distance might still get recommended in the distance-penalizing mode, bringing the count slightly up (recommending some medium-distance users that might be just above the previous-followee average distance).

5.2 Future work

Some future paths to explore regarding this project include: (1) much more exploration of the hyper-parameter settings, particular for the weight vector W's components; (2) more analysis of centrality measures, such as closeness centrality and PageRank score; (3) cleaner handling of normalized versus unnormalized values, such that normalized values can still be easily related at a glance to an intuitive measure of how well a particular metric did (as mentioned before, this includes simple additional steps, such as getting the distributions of centrality scores so we can see where along the spectrum our centrality scores lie); and (4) perhaps explore the redundancy in degree-related network features we used - is there a use of having several degree-related metrics, such as in-degree, out-degree, degree, and degree centrality, or would a subset (such as simply degree, for example) work just as well?

5.3 Closing

Overall, we consider the project a relative success. Unfortunately, the author's time was extremely crunched when executing this project (had only approximately 3 days after completing the basic data-gathering phase to work on this project), and so much more is left to be done and explored in this topic. In general, though, we were able to accomplish the general goal we were seeking: build a recommendation system that can recommend users with a bias towards long-distance users throughout the world for some subset of the time. We hope that this base will prove useful in future studies of network analysis and social network recommendation, and that ultimately this will somehow prove to be of some (even miniscule) contribution to some future work (by the author or others) that will increase the connectivity between all types of people across the face of this small single planet we call home.

Table 4: Summary statistics - location/distance. Data has already been normalized as in Section 3.3. Also see Section 3.8 for details on terminology, such as "long-distance" recommendations.

Previous followee distance	Distance, penalized	Distance, rewarded	"Long-distance" recommendations
0.27493	0.07199	0.79342	4.24531

References

[1] Google Maps API. `https://github.com/googlemaps/google-maps-services-python`, `https://developers.google.com/maps/`.

[2] Haversine formula and related Python library. `https://en.wikipedia.org/wiki/Haversine_formula`, `https://github.com/mapado/haversine`.

[3] SNAP: Network datasets: 476 million Twitter tweets. `https://snap.stanford.edu/data/twitter7.html`.

[4] Snap: Social circles: Twitter data. `https://snap.stanford.edu/data/egonets-Twitter.html`.

[5] Snap.py - SNAP for Python. `http://snap.stanford.edu/snappy/index.html`.

[6] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[7] John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 199–206, New York, NY, USA, 2010. ACM.

[8] David Liben-Nowell, Jasmine Novak, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11623–11628, 2005.