Hallo Community Growth Project

Michael Fryer, Cody Kesler, Eric Todd

Brigham Young University, Provo, UT

December 13, 2019

1 Introduction

Hallo is a startup based in Provo, Utah built around a global community for English language learning through a mobile application. The application allows users to: create a profile; message, chat, call, and friend other users; and view streams and videos of English teaching streamers. The community that naturally starts to form through interactions on the application is the object of focus for this analysis and project.

We met the founders of Hallo at the BYU career fair and discovered they had a need for data analysis and modeling. To best utilize our skills and help Hallo, we have partnered with their CTO in defining the goals, scale, and data for this project. The CTO has provided access and guidance in working with the data under a loose non-disclosure agreement (NDA).

With a recent surge of growth, retaining users has been a primary concern of Hallo. After consulting with the founders of the company and taking their goals and business model into account, we have decided retention should be accomplished through community growth. Naturally, we assume communities are characterized by friendships, which are created by mutual communication on the app. We loosely define the strength of a friendship by the frequency of communication within the app. Further, we presume that a strongly connected group of friends (clique) is more effective in retention than many separate groups of two. Thus we focus on building cliques and connecting the community to increase involvement.

1.1 Problems

We are interested in building a friend-recommendation system in order to promote community growth and target user activity with the intent of user retention. As a precursor to solving this problem we first have to explore and understand the data to determine what features are useful in predicting friendships, activity-level of users, and other indicators of growth. To that end, we set out to answer the following questions:

1.2 Questions

- What similarities exist between friends?
- How can we predict friendships?
- What causes a strongly connected group of friends?
- What indicates a person starting to leave the app?

1.3 Previous Work

While similar recommendation systems exist for other social networks (such as Facebook and Instagram), Hallo has yet to explore these questions with regards to their unique community. Thus, this is the first exposure Hallo will have in using their data for analysis. Within the field of Social Network Analysis (SNA), there is research regarding *link prediction*, which describes different methods and practices we could use for predicting friends. However, the scope of this project is to *understand* the data so that we can create a friend recommendation system that will promote community growth. This will enable us to perform an expanded analysis of the Hallo community so that in the future we can use these SNA techniques and other machine learning algorithms for prediction. This combined with the fact that the nature of communities differs from platform to platform, we use techniques specific to our data to analyze and build out the recommendation system.

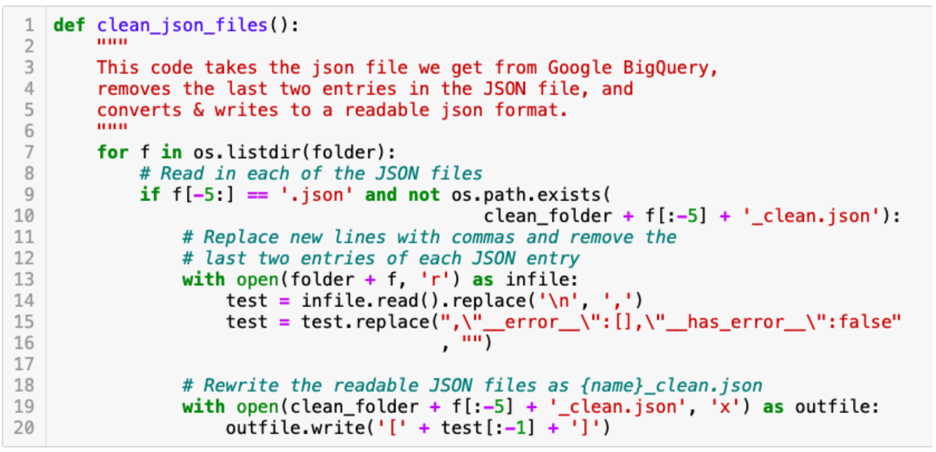
1.4 Data Source

The data set was obtained from Hallo's database which was stored in NoSQL format on Google's online suite, Firebase. This is the database format Hallo decided to use to collect data from their application. Schema includes: "users", "friends", "messages", "livestreams", "followed livestreams", "calls", etc. These tables include data about user interactions with other users and the app.

2 Data Cleaning

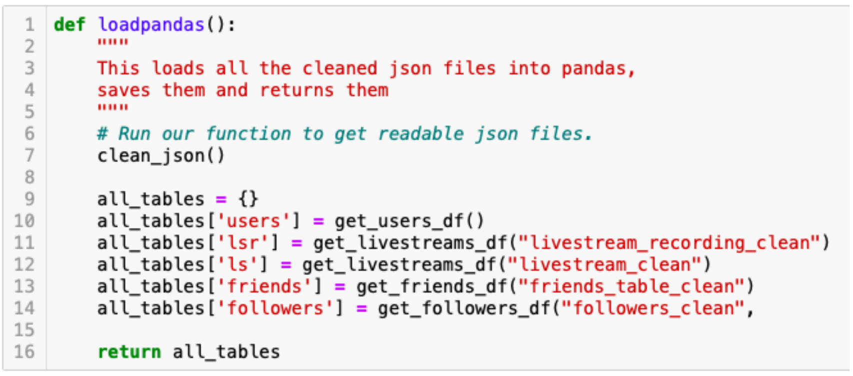
2.1 Primary Issues with the Data

As the data is natively stored in NoSQL format, we needed to extract the data from Google's Big-Query, which exports it to a .json format. To format the data in a manageable format, it was loaded into a pandas data frame. Each table of given data had its own columns and extra information to remove. Figure 1 depicts cleaning the .json files and Figure 2 depicts one function used to load the followers table into pandas. Figure 3 shows all the tables loaded into pandas.



```python
def clean_json_files():
    """
    This code takes the json file we get from Google BigQuery,
    removes the last two entries in the JSON file, and
    converts & writes to a readable json format.
    """
    for f in os.listdir(folder):
        # Read in each of the JSON files
        if f[-5:] == '.json' and not os.path.exists(
                                        clean_folder + f[:-5] + '_clean.json'):
            # Replace new lines with commas and remove the
            # last two entries of each JSON entry
            with open(folder + f, 'r') as infile:
                test = infile.read().replace('\n', ',')
                test = test.replace(",\"__error__\":[],\"__has_error__\":false"
                                    , "")

            # Rewrite the readable JSON files as {name}_clean.json
            with open(clean_folder + f[:-5] + '_clean.json', 'x') as outfile:
                outfile.write('[' + test[:-1] + ']')
```

Figure 1: Cleaning the .json Files



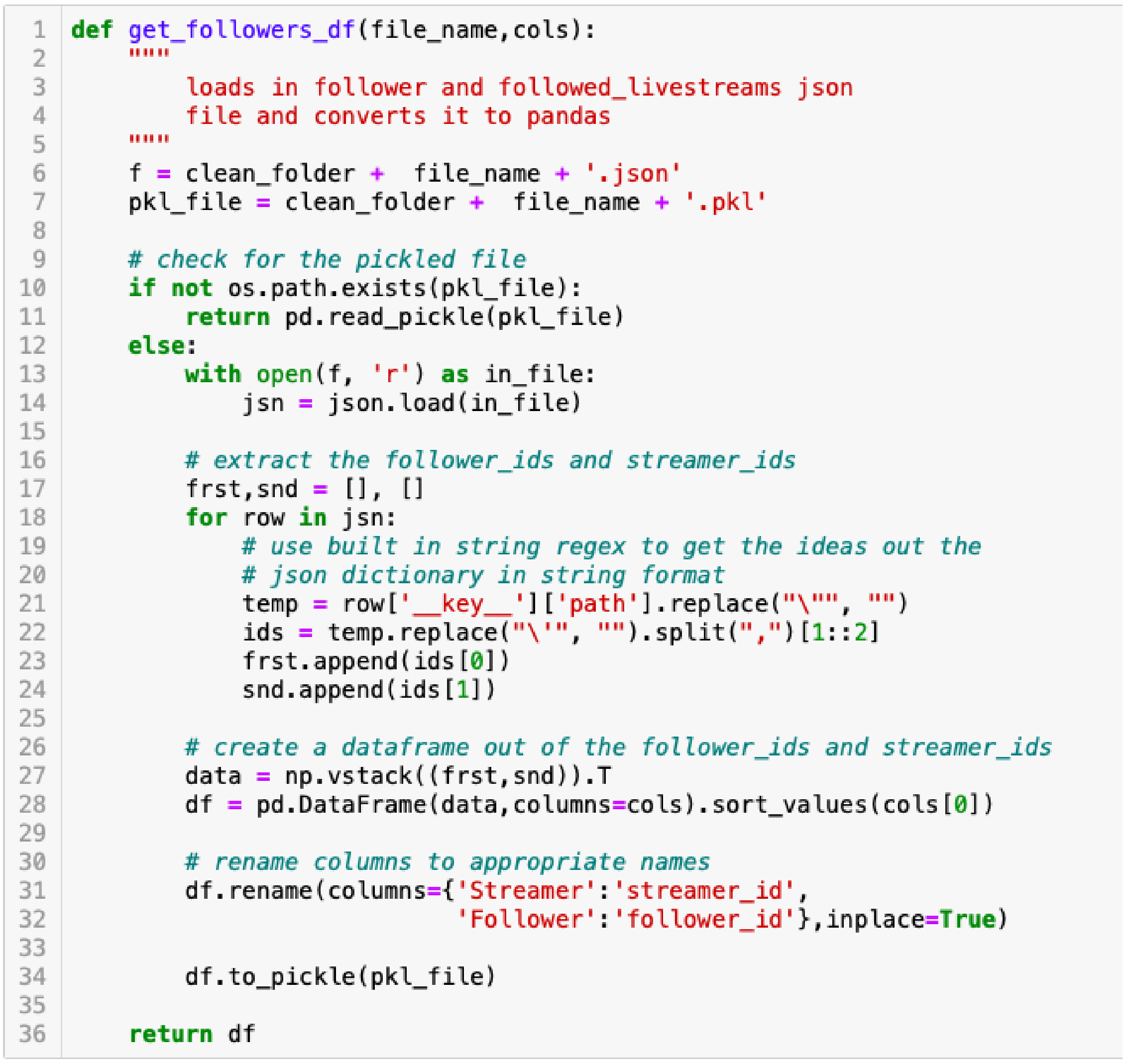
```python
def loadpandas():
    """
    This loads all the cleaned json files into pandas,
    saves them and returns them
    """
    # Run our function to get readable json files.
    clean_json()

    all_tables = {}
    all_tables['users'] = get_users_df()
    all_tables['lsr'] = get_livestreams_df("livestream_recording_clean")
    all_tables['ls'] = get_livestreams_df("livestream_clean")
    all_tables['friends'] = get_friends_df("friends_table_clean")
    all_tables['followers'] = get_followers_df("followers_clean",

    return all_tables
```

Figure 2: Cleaning All the data files

2.2 Strengths of Data for Proposed problem

- The features needed for the task of the project were not extremely complex. Due to this and since it was collected through automation, minimal errors were encountered in the cleaning process.

- Another strength is that we have data on interactions within a user base of around 300,000. This size will allow us to try out and use a significant amount of features for predicting friendship links and retention.
- The data we currently have is rich for feature extraction. We have the opportunity to obtain text, images, and video encapsulating user interaction which could be mined for information through NLP and Computer Vision. Since there could be many factors influencing why someone is not retained in the app, we may use these techniques to infer more about the person to better predict and understand why they leave. However, due to privacy concerns, we will not be working with photos or videos for the time being.



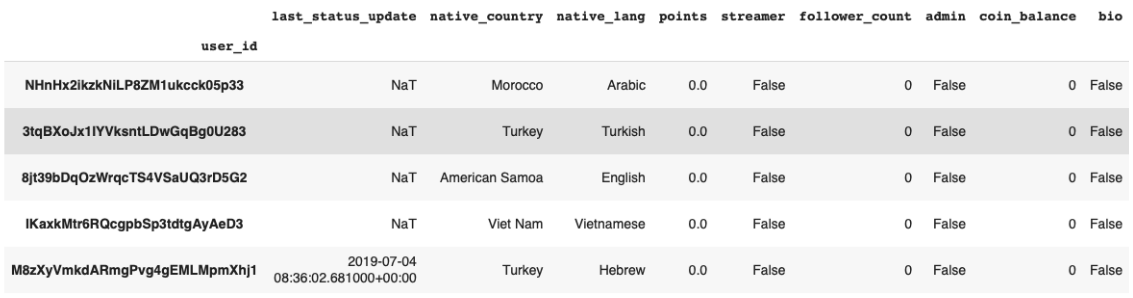
```python
def get_followers_df(file_name,cols):
    """
        loads in follower and followed_livestreams json
        file and converts it to pandas
    """
    f = clean_folder +  file_name + '.json'
    pkl_file = clean_folder +  file_name + '.pkl'

    # check for the pickled file
    if not os.path.exists(pkl_file):
        return pd.read_pickle(pkl_file)
    else:
        with open(f, 'r') as in_file:
            jsn = json.load(in_file)

        # extract the follower_ids and streamer_ids
        frst,snd = [], []
        for row in jsn:
            # use built in string regex to get the ideas out the
            # json dictionary in string format
            temp = row['__key__']['path'].replace("\"", "")
            ids = temp.replace("\'", "").split(",")[1::2]
            frst.append(ids[0])
            snd.append(ids[1])

        # create a dataframe out of the follower_ids and streamer_ids
        data = np.vstack((frst,snd)).T
        df = pd.DataFrame(data,columns=cols).sort_values(cols[0])

        # rename columns to appropriate names
        df.rename(columns={'Streamer':'streamer_id',
                           'Follower':'follower_id'},inplace=True)

        df.to_pickle(pkl_file)

    return df
```

Figure 3: Cleaning the Followers Table

2.3 Weaknesses and Bias of Data for Proposed Problem

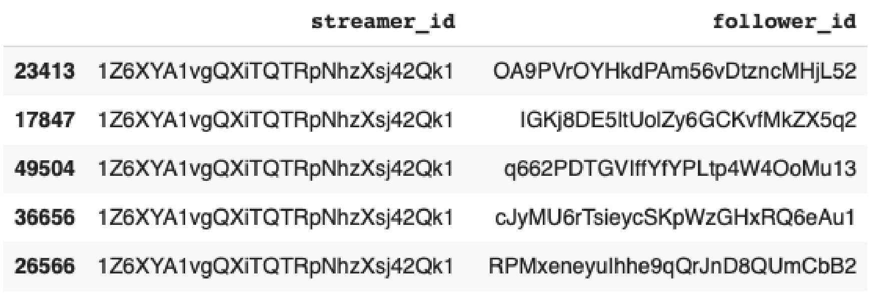
- Time series analysis assumes patterns of the past can be used to infer patterns of the future. This assumption may not be true with the type of community described by our data. Specifically, our data encompasses the initial growth of the community which has many inherent biases such as who initially downloaded the app and how much their intents lined up with the purpose of the app. Perhaps there exists a different set of initial users that would result in a more successful cascade of naturally forming friendships. Such could render all current time-series data of the past irrelevant in predicting the strongest factors of friendship in the future.

- In the case of time series data being relevant, there are some data that lack time stamps which could be a significant loss.
- Another weakness of our data is that the current setup of Hallo's data storage deletes call data after a certain period. This was a business decision made to reduce data storage costs, but it limits our ability to understand user interactions to just messages and livestreams.
- For obvious reasons, personal data such as culture, religion, and interests, often used in community analysis, is not available to us. This will make it more difficult to predict community interactions.
- We lack data on lasting relationships from students to teachers and so we will have to create an evolving model with regards to suggesting teachers to students.
- Another source of potential bias in the data is that the English Proficiency is self-reported. Though some users may rate themselves incorrectly, we assume that most users will report their true proficiency level since they are using the app to learn English.
- Potential bias could also come from limited data as the app is new, and the user base for the app could be skewed with the newness of the community. Since users of the app are all learning English, there might also be a potential bias in the variety of users.



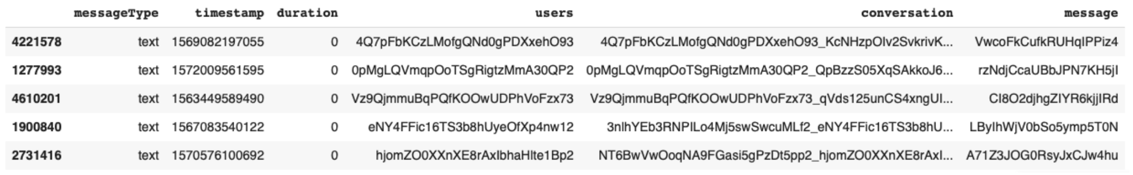
user_id	last_status_update	native_country	native_lang	points	streamer	follower_count	admin	coin_balance	bio
NHnHx2ikzkNiLP8ZM1ukcck05p33	NaT	Morocco	Arabic	0.0	False	0	False	0	False
3tqBXoJx1IYVksntLDwGqBg0U283	NaT	Turkey	Turkish	0.0	False	0	False	0	False
8jt39bDqOzWrqcTS4VSaUQ3rD5G2	NaT	American Samoa	English	0.0	False	0	False	0	False
IKaxkMtr6RQcgpbSp3tdtgAyAeD3	NaT	Viet Nam	Vietnamese	0.0	False	0	False	0	False
M8zXyVmkdARmgPvg4gEMLMpmXhj1	2019-07-04 08:36:02.681000+00:00	Turkey	Hebrew	0.0	False	0	False	0	False

Figure 4: Users Table



	streamer_id	follower_id
23413	1Z6XYA1vgQXiTQTRpNhzXsj42Qk1	OA9PVrOYHkdPAm56vDtzncMHjL52
17847	1Z6XYA1vgQXiTQTRpNhzXsj42Qk1	IGKj8DE5ltUolZy6GCKvfMkZX5q2
49504	1Z6XYA1vgQXiTQTRpNhzXsj42Qk1	q662PDTGVIffYfYPLtp4W4OoMu13
36656	1Z6XYA1vgQXiTQTRpNhzXsj42Qk1	cJyMU6rTsieycSKpWzGHxRQ6eAu1
26566	1Z6XYA1vgQXiTQTRpNhzXsj42Qk1	RPMxeneyulhhe9qQrJnD8QUmCbB2

Figure 5: Followers Table



	messageType	timestamp	duration	users	conversation	message
4221578	text	1569082197055	0	4Q7pFbKCzLMofgQNd0gPDXxehO93	4Q7pFbKCzLMofgQNd0gPDXxehO93_KcNHzpOlv2SvkrivK...	VwcoFkCufkRUHqIPPiz4
1277993	text	1572009561595	0	0pMgLQVmqpOoTSgRigtzMmA30QP2	0pMgLQVmqpOoTSgRigtzMmA30QP2_QpBzzS05XqSAkkoJ6...	rzNdjCcaUBbJPN7KH5jl
4610201	text	1563449589490	0	Vz9QjmmuBqPQfKOOwUDPhVoFzx73	Vz9QjmmuBqPQfKOOwUDPhVoFzx73_qVds125unCS4xngUl...	Cl8O2djhgZIYR6kjjIRd
1900840	text	1567083540122	0	eNY4FFic16TS3b8hUyeOfXp4nw12	3nlhYEb3RNPILo4Mj5swSwcuMLf2_eNY4FFic16TS3b8hU...	LByIhWjV0bSo5ymp5T0N
2731416	text	1570576100692	0	hjomZO0XXnXE8rAxlbhaHlte1Bp2	NT6BwVwOoqNA9FGasi5gPzDt5pp2_hjomZO0XXnXE8rAxl...	A71Z3JOG0RsyJxCJw4hu

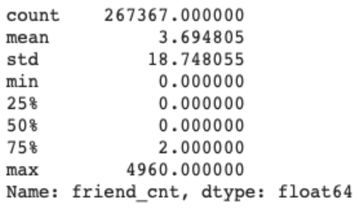
Figure 6: Messages Table

3 Feature Engineering

After cleaning, organizing and assessing the data, we determined which features would need to be extracted and interpolated from to properly answer the questions posed.

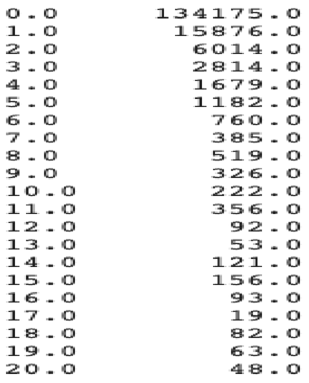
3.1 Added User Features

- Total Number of Friends
  - Description: Number of friends the user has in the app
  - Formulation: $U_f(x) = sig_f(|\{u|u \in friends(x)\}|)$
    - $sig_f(x) = max(0, tanh(\frac{x}{6}))$
  - Justification: Getting the number of friends a user has will be a substantial indicator of how active the user is on the app. As seen in Figure 7 (b), the difference between the number of users who have 11 friends and 12 friends is 356 and the difference from 12 to 13 is only 92. This gives reason to believe a function should level off to 1 at 12. The $tanh(\frac{x}{6})$ fits this behavior very well as it cuts off at 0 and reaches 1 around an x value of 12.



```
count    267367.000000
mean          3.694805
std          18.748055
min           0.000000
25%           0.000000
50%           0.000000
75%           2.000000
max        4960.000000
Name: friend_cnt, dtype: float64
```

(a) Stats of Number of Friends



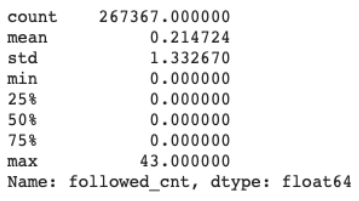
```
0.0       134175.0
1.0        15876.0
2.0         6014.0
3.0         2814.0
4.0         1679.0
5.0         1182.0
6.0          760.0
7.0          385.0
8.0          519.0
9.0          326.0
10.0         222.0
11.0         356.0
12.0          92.0
13.0          53.0
14.0         121.0
15.0         156.0
16.0          93.0
17.0          19.0
18.0          82.0
19.0          63.0
20.0          48.0
```

(b) Number of Friends - Difference in Number of users with Given # Friends

Figure 7: Number of Friends Descriptors

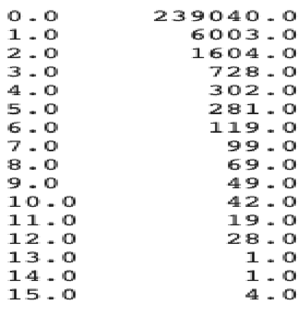
- Summarized User Bio
  - Description: A binary variable, has or does not have a user bio, (0,1).
  - Formulation: $U_b(x) = \mathbb{1}_{bio}(x)$
  - Justification: As a user has a biography filled out, they are more likely to be earnest in using the app and being involved in the community.
- Last Online
  - Description: One Hot Encoding of the last time they used the app: Last Day, Last Week, Last Month
  - Formulation: $U_l(x) = sum([\frac{2}{3} * \mathbb{1}_{day}(x), \frac{1}{6} * \mathbb{1}_{week}(x), \frac{1}{6} * \mathbb{1}_{month}(x)])$. Based on the relative "last day" for our current data.
  - Justification: Having an indicator for when the user was last online, and how frequent, will also supply a great indicator of the activity of the user. If a user has been active in the last day it is more of an indicator the user is active than if the user was only active in the week or month.
- Number of Followed Streamers
  - Description: Number of followed streamers.
  - Formulation: $U_s(x) = sig_s(|\{u|u \in followed\_streamers(x)\}|)$

    * $sig_s(x) = max(0, tanh(\frac{x}{6}))$
  - Justification: Getting the number of streamers a user follows will be a substantial indicator for how active the user is on the app. Figure 8 (b) shows a drop off in the difference of followed streamers at around 12 users, the same as the number of friends per person.



```
count    267367.000000
mean          0.214724
std           1.332670
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          43.000000
Name: followed_cnt, dtype: float64
```

(a) Statistics of Followed Streamers for Users

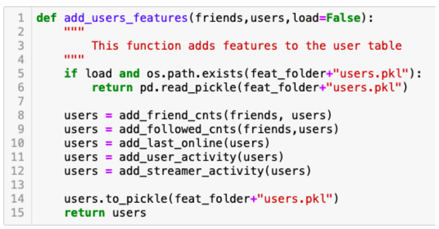


```
0.0       239040.0
1.0         6003.0
2.0         1604.0
3.0          728.0
4.0          302.0
5.0          281.0
6.0          119.0
7.0           99.0
8.0           69.0
9.0           49.0
10.0          42.0
11.0          19.0
12.0          28.0
13.0           1.0
14.0           1.0
15.0           4.0
```

(b) Number of Followed Streamers - Difference in Number of users with given # of Features

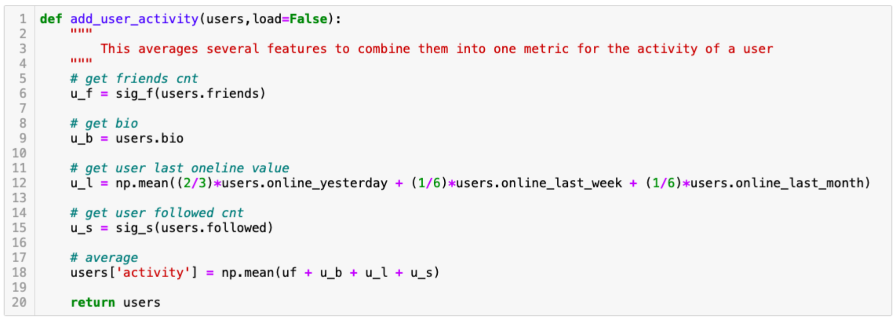
Figure 8: Follower Count Descriptors

- User Activity Measure
  - Description: Weighted Average of Last Online, Bio, Friends Count, Number of followed Streamers
  - Formulation: $U_a(x) = a * U_f(x) + b * U_s(x) + c * U_b(x) + d * U_l(x)$
    * Where a, b, c, d are weighted coefficients : $a + b + c + d = 1$. Here we have decided to treat each feature evenly when considering their contribution to the measure of user activity. Thus, $a = b = c = d = .25$
  - Justification: Combining the indicators extracted from the data can give advanced insight into how involved the user is with the app and community. This will help with identifying users that will leave the app or who do not use it much to be able to target them and bring them back into the community.



```python
def add_users_features(friends,users,load=False):
    """
        This function adds features to the user table
    """
    if load and os.path.exists(feat_folder+"users.pkl"):
        return pd.read_pickle(feat_folder+"users.pkl")

    users = add_friend_cnts(friends, users)
    users = add_followed_cnts(friends,users)
    users = add_last_online(users)
    users = add_user_activity(users)
    users = add_streamer_activity(users)

    users.to_pickle(feat_folder+"users.pkl")
    return users
```

Figure 9: All User Features Added

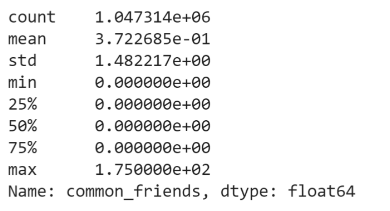


```python
def add_user_activity(users,load=False):
    """
        This averages several features to combine them into one metric for the activity of a user
    """
    # get friends cnt
    u_f = sig_f(users.friends)

    # get bio
    u_b = users.bio

    # get user last oneline value
    u_l = np.mean((2/3)*users.online_yesterday + (1/6)*users.online_last_week + (1/6)*users.online_last_month)

    # get user followed cnt
    u_s = sig_s(users.followed)

    # average
    users['activity'] = np.mean(uf + u_b + u_l + u_s)

    return users
```

Figure 10: Calculate the User Activity Feature

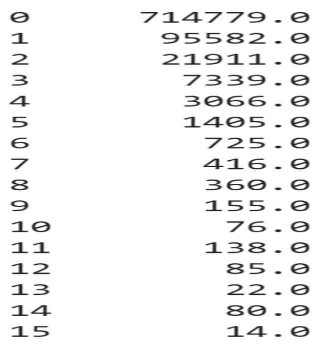
3.2 Added Friendship Features

- Messages between Friends
  - Description: Number of messages between two friends.
  - Formulation: $F_m(u1, u2) = |\{messages(u1, u2)\}|$
  - Justification: The amount of messages sent between two users indicates a much higher friendship strength as they interact willingly and more frequently.
- Friendship strength
  - Description: Weighted combination Frequency of calls and messages and Friends in Common.
  - Formulation: $F_{str}(u1, u2) = sig_{str}(F_m(u1, u2))$
    * $sig_{str}(x) = min(1, (\frac{1}{2} + e^{\frac{x}{-6}})^{-1} - \frac{2}{3})$
  - Justification: The strength of a community is measured, in part, as a sum of individual friendships with their corresponding strengths. The main source of friend interaction is seen in communication through messages and calls. Since more interaction is given in a video chat, we assume this is the best indicator of friendship in comparison to a voice call. However, as mentioned above, Hallo currently deletes its call data, so we can only use messages for this computation. We hope to be able to incorporate call data if it is ever stored, but for now, we cannot. We feed the number of messages between friends into a quasi-sigmoid function to get a friendship strength value bounded by 0 and 1.
- English Proficiency Difference
  - Description: Normalized Difference in English Proficiency
  - Formulation: $F_e(u1, u2) = 1 - abs(eng\_prof(u1) - eng\_prof(u2))/4$
  - Justification: Normalizing the difference in English Proficiency provides a good measure of how similar the background of 2 users will be. The max English proficiency is 4 and the min is 0, thus we divide by 4 to normalize. We want the same amount of English Proficiency, i.e. the difference 0, to contribute the most, so we subtract the normalized difference from 1.
- Friends in common
  - Description: Number of friends two users have in common.
  - Formulation: $F_f(u1, u2) = sig_f(|\{friends(u1) \cap friends(u2)\}|)$
    * $sig_f(x) = max(0, min(1, (e^{-\frac{1}{2}(x-7)})^{-1}))$
  - Justification: If two users have many friends in common, we assume similarities exist between the individuals of those groups, thus the users should also be similar. This feature is therefore important in predicting future friends. There is also behavioral research suggesting friends in common are good indicators of friendship as shown in the justification of its coefficient below. Our sigmoid approaches 1 at x = 12, because if users have more than 12 friends in common, they have already found a clique which in which their similarity is already saturated, which means they are highly likely to already be friends, which does not contribute to the goal of friend recommendation.
- Streamers in common
  - Description: Number of common streamers followed between two friends.
  - Formulation: $F_s(u1, u2) = sig_s(|\{streamers(u1) \cap streamers(u2)\}|)$
    * $sig_s(x) = max(0, min(1, (e^{-\frac{1}{2}(x-7)})^{-1}))$
  - Justification: If friends follow the same streamer, they are likely to interact more, have similar interests, or like the same kind of users. This will help indicate how similar two friends are. As noted by the numbers in Figure 12 below, we note that at about the difference drops 55 between 12 common streamers and 13 common streamers. This the sigmoid function reaches close to one at 5.



```
count    1.047314e+06
mean     3.722685e-01
std      1.482217e+00
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      0.000000e+00
max      1.750000e+02
Name: common_friends, dtype: float64
```

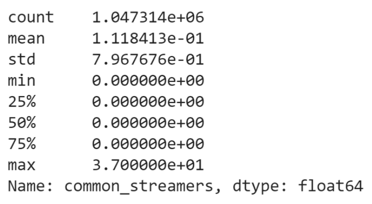
(a) Statistics of Friends in Common Users



```
0       714779.0
1        95582.0
2        21911.0
3         7339.0
4         3066.0
5         1405.0
6          725.0
7          416.0
8          360.0
9          155.0
10          76.0
11         138.0
12          85.0
13          22.0
14          80.0
15          14.0
```

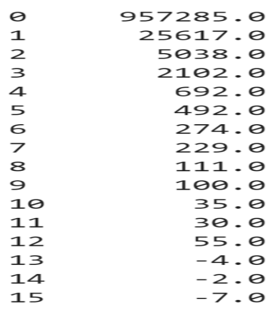
(b) Difference in Friends in Common

Figure 11: Common Friends Count Descriptors



```
count    1.047314e+06
mean     1.118413e-01
std      7.967676e-01
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      0.000000e+00
max      3.700000e+01
Name: common_streamers, dtype: float64
```

(a) Statistics of Followings in Common Users

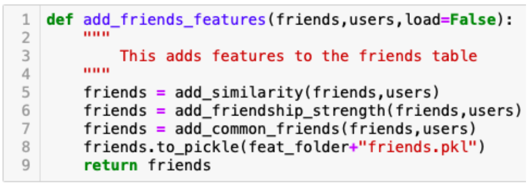


```
0       957285.0
1        25617.0
2         5038.0
3         2102.0
4          692.0
5          492.0
6          274.0
7          229.0
8          111.0
9          100.0
10          35.0
11          30.0
12          55.0
13          -4.0
14          -2.0
15          -7.0
```

(b) Difference in Common Followings

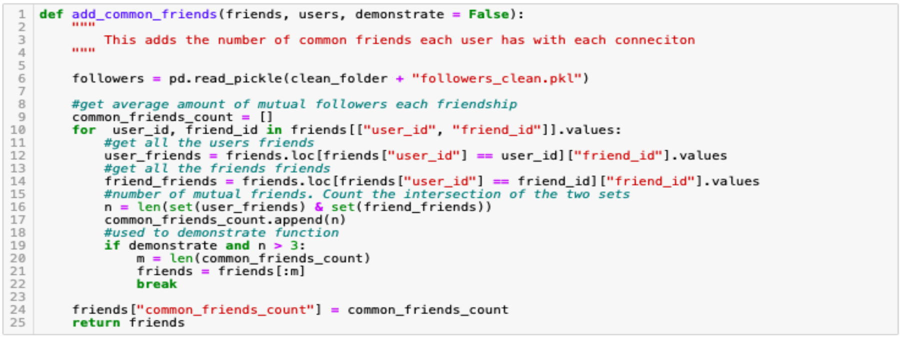
Figure 12: Common Following Count Descriptors

- Similarity
  - Description: Weighted average of Native Country, Native Language, English Proficiency, Common Followed/Streamers, and Common Friends with heavier weight on English Proficiency.
  - Formulation: $F_{sim}(u1, u2) = e * F_e(u1, u2) + f * \mathbb{1}_{country}(u1, u2) + g * \mathbb{1}_{language}(u1, u2) + h * F_s(u1, u2) + i * F_c(u1, u2)$
    - ∗ Where $e + f + g + h + i$ are weighted coefficients : $e + f + g + h + i = 1$. Coefficient i is justified here and e, f, g, h are justified in the next section.
  - Justification: We assume that friends tend to spend time on the app with users that are similar to them in characteristics such as culture, similarly followed streamers, or common friends. If two users have the same English proficiency levels then we hypothesize they are a good match for learning together. On the other hand, if two users have different proficiency levels, then we assume one will slow down the learning of the other. We reflect this belief in weighting language proficiency relatively higher.
    - ∗ The weight i, given to the friends in common feature, is justified by a commonly held view in Behavioral Psychology that users generally pick friends they consider similar to themselves [1]. Users tend to compare similarity based upon a variety of things such as age, common world views, and many other complex dimensions for which we do not have data [2] [3]. Thus, if person A is friends with Person B, then they will likely have some of these similarities for which we cannot directly measure. Further, if person C is friends with person A, then person C is likely to have similarities with person B. If two users have many friends in common, then we should expect they likely have even more things in common due to each friendship having a unique handful of the many possible similarities.



```python
def add_friends_features(friends,users,load=False):
    """
        This adds features to the friends table
    """
    friends = add_similarity(friends,users)
    friends = add_friendship_strength(friends,users)
    friends = add_common_friends(friends,users)
    friends.to_pickle(feat_folder+"friends.pkl")
    return friends
```

Figure 13: All Friendship Features Added

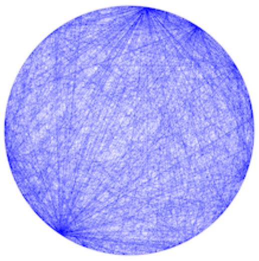


```python
def add_common_friends(friends, users, demonstrate = False):
    """
        This adds the number of common friends each user has with each conneciton
    """

    followers = pd.read_pickle(clean_folder + "followers_clean.pkl")

    #get average amount of mutual followers each friendship
    common_friends_count = []
    for  user_id, friend_id in friends[["user_id", "friend_id"]].values:
        #get all the users friends
        user_friends = friends.loc[friends["user_id"] == user_id]["friend_id"].values
        #get all the friends friends
        friend_friends = friends.loc[friends["user_id"] == friend_id]["friend_id"].values
        #number of mutual friends. Count the intersection of the two sets
        n = len(set(user_friends) & set(friend_friends))
        common_friends_count.append(n)
        #used to demonstrate function
        if demonstrate and n > 3:
            m = len(common_friends_count)
            friends = friends[:m]
            break

    friends["common_friends_count"] = common_friends_count
    return friends
```

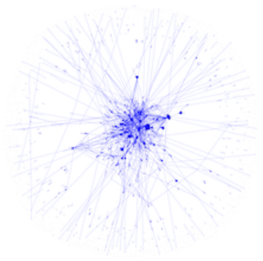
Figure 14: Calculate Common Friends of 2 Users

4 Data Visualizations

In this section, we found that we could predict friendships from Native Country, Native Language, English Proficiency, Common Followed/Streamers, and Common Friends. Coincidentally, friends tended to have similar values for these features. These findings justified our similarity function between two users that are used to predict the likelihood of friendship.



(a) Global Connections



(b) Local Clique

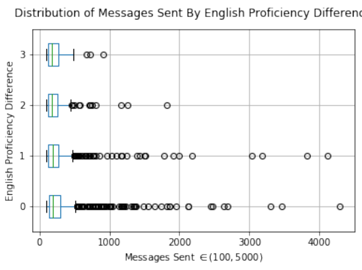
Figure 15: Network Connections

To get an overall sense of the community at hand, we graphed a local and global model of the community through friendship links. Figure 15(a) led us to discover that there exist a few users with a large portion of all the connections. Figure 15(b) demonstrates that users connected to these popular users tend to have very few friends. Since a connection usually implies similarity, friends of the popular user should also be friends amongst themselves. This gives a great starting point for future community growth.

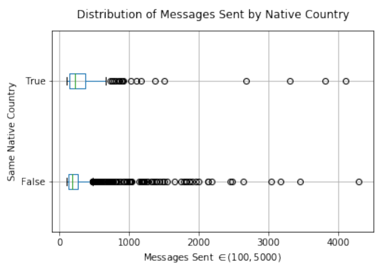
One mystery that remains is how some users obtained so many connections over time and why others did not. It would be informative to know what separates the two. It would be extremely valuable to know why friends of popular users had few friends overall. Unfortunately, these phenomena were not explored due to the lack of time-series data. Timing is likely essential for the popularity of some users and not addressing this with any analysis could lead to very biased results.

4.1 Justifying Friendship Similarity Score Weights

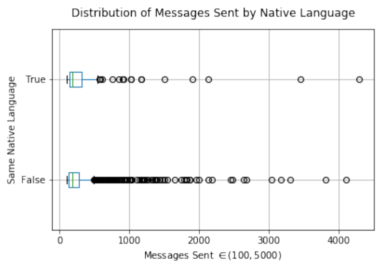
When producing an equation to define a similarity score there were 5 features we deemed necessary to include and thus the need arose to determine weights for each feature, e, f, g, h, and i. This was done by looking at each feature concerning current friendships. Each graph below in Figure 16 measures the respective categorical variable to the number of messages sent. As a baseline, one can see that the IQR for each graph corresponds to a larger number of messages sent between two users for that feature being the same for the two users. This suggests these features are reasonable to make part of a similarity score between two users and more importantly, that two users are likely to interact if they have these features in common.



(a) (English Prof. Diff.) Coefficient e



(b) (Native Country) Coefficient f



(c) (Native Language) Coefficient g

Figure 16: Friendship Strength Score Features

We present the 75% quartile for the categories of each of the above features and the differences in the 75% quartile from the True or Best to the False or Second Best. The differences will give light to how much each feature should contribute to our function.

- English Proficiency Difference - 0: 292, 1:272, 2: 260, 3: 278.
  - Difference: 20
- Native Country - True: 364, False: 268.
  - Difference: 96
- Native Language - True: 310, False 276.
  - Difference: 34

Thus the weight for Native Country would be the highest, while English Proficiency and Native Language would contribute less to a similarity measure of users.

The weight h for the last feature, the number of commonly followed streamers, is determined by regression. This regression will learn the relationship between the number of messages and the number of commonly followed streamers. Figure 17 shows this correlation between messages and common followed streamers with correlation .458. This indicates a correlation, but was not high, as it is less than .5.

The plot below in Figure 18 (a) shows that as the similarity increases between two users, the frequency and amount of messages between them tends to increase. Although this is not a perfect fit and the number of messages seems like a good indicator for friendship, the number of messages is only one dimension of user interaction and thus not a perfect metric to validate our similarity measure. Because we are currently lacking other user interaction data and are forced to only use message data, we have supplemented some of the data with behavioral research concerning friendships and interactions for calculating the similarity

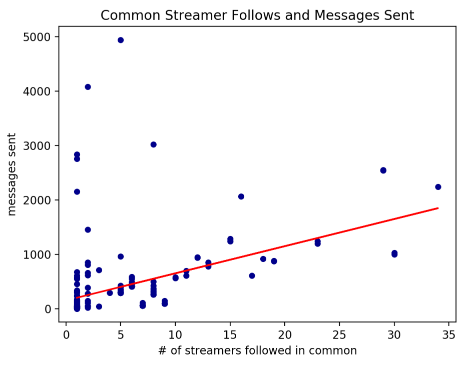
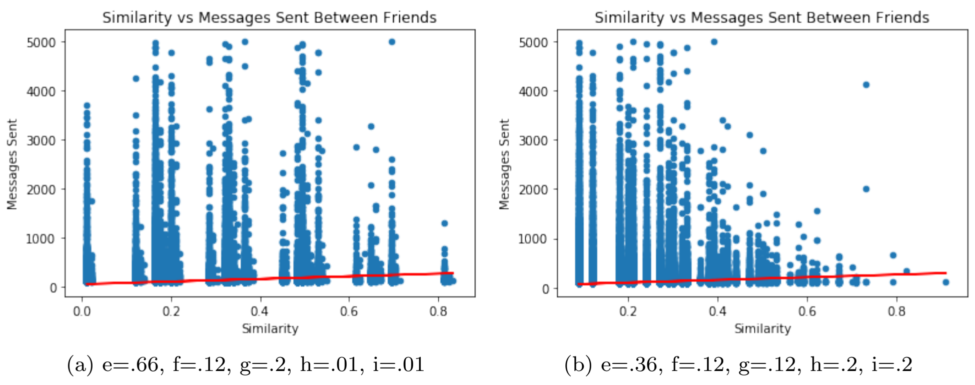


Figure 17: Plot of Common Streamers Followed vs Messages Sent

measure. We can see in Figure 18 (b) that when the weights associated with common friends and common streamers increase, our similarity measure becomes less correlated with the number of messages sent between friends. While these features may not be correlated with messages, behavioral research suggests they are still valuable in predicting friendships because users with similar friends tend to have more in common and have a higher likelihood of becoming friends.



(a) e=.66, f=.12, g=.2, h=.01, i=.01

(b) e=.36, f=.12, g=.12, h=.2, i=.2

Figure 18: Messages vs Similarity

4.2 Similar Features

The graph in Figure 19 shows the percent of friends that share the same native language, while only 9.7% of friends share the same native country. We used these as part of our similarity score between friends.

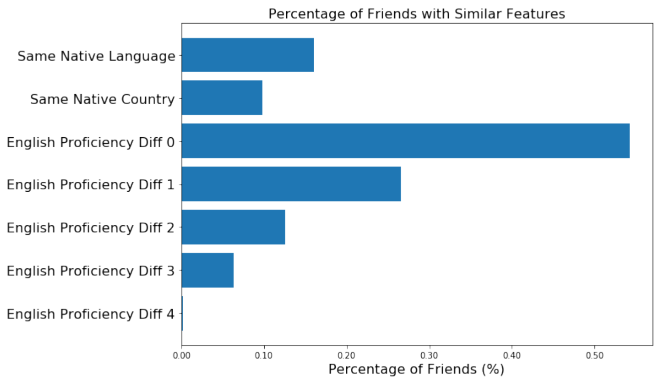


Figure 19: Percentage of Friends that Share Features

Similarly, over 50% of friends have the same English Proficiency level (values of 0-4). The percentage of friends that have a *different* English proficiency level drops significantly compared to those with the same proficiency level. About 27% of friends differ by one level of English proficiency, 13% of friends differ by 2 levels, 6% differ by 3 levels, and only 1% differ by 4 levels of proficiency. This suggests that English Proficiency is a good indicator of whether or not two users will become friends on the app.

5 Conclusion

From our data, we see friends that are from the same native country or have the same native language tend to send more messages on average than those who did not share those similarities. The Difference in English proficiency also follows a similar trend. Though the number of common friends and common streamers between two users doesn't share this same correlation with messages, they are still valuable indicators in predicting friendships. Given these factors, we are confident similarity features we have will be good predictors in determining future friendships.

Having also developed an activity measure of a user, we are now able to determine which users are fairly inactive in the app. This will allow the company to target individuals who may have used the app once or twice or who are declining in their activity, giving the company more control over the growth of the community and engagement of its users. If the community can retain users and increase their engagement, it can grow faster and provide the company with more success and investment in the long run.

6 Future Work

For future work, the derived features about similarity can be used with clustering, algorithms, and similarity metrics to determine a proper friend recommendation algorithm to be implemented for use within the Hallo app. The activity measure can be used to learn how to target users who are leaving the app and get them involved in the application. The features derived today will provide a foundation for the next step of community retention and growth for the Hallo Community.

References

[1] Similarity and propinquity in friendship formation. Nahemow, L., & Lawton, M. P. (1975). Journal of Personality and Social Psychology, 32(2), 205213.
`https://doi.org/10.1037/0022-3514.32.2.205`

[2] Childrens drawing of friendship and family relationships in different cultures, Pinto, G., Bombi, A.S.
`https://www.tandfonline.com/doi/abs/10.1080/016502597385225`

[3] The Relations Among Talking, Liking, and Similarity Between Friends, Francine M. Deutsch, Lisa Sullivan, Cristina Sage, Nicoletta Basile,
`https://journals.sagepub.com/doi/10.1177/0146167291174008`