Identifying Influential Friends in Social Network Gaming

December 11, 2012

Group 9

Ashley Jin
ashpjin@stanford.edu

Jessica Tai
jmtai@stanford.edu

Karan Verma
karanv@stanford.edu

1 Introduction

Facebook is currently one of the largest social networks with millions of daily active users logging in to access content posted by friends, family, colleagues, or even mere acquaintances. This social network and others have grown to become hosts to external applications that leverage the capabilities of the platform for anything from authentication to messaging to social interactions and customizations[1]. Some of the largest applications that leverage these features are the social online games. These gaming companies create applications that allow players with social media accounts to share their personal profile and friend information with the game to create a customized game experience. These companies are then able to collect rich behavioral data that can be used to give insight into communication, influence, and trade patterns in the games.

Often, companies want to increase their active user base by bringing in new users or reactivating dormant users. One technique they employ is convincing players to send invites to friends by offering in-game incentives. However, most companies will ideally want to bring in new monetizing users, users who spend real money in the game. Therefore, it is important to target players who are good at bringing in monetizing users specifically. To this aim, we will experiment with influence maximization algorithms on one of these social game datasets to identify good methods of finding the most influential users at bringing in monetizing users. We will also perform a side experiment where we attempt to identify common attributes of these influential players.

2 Prior Work

There is significant work being done on social network data e.g. analyzing network properties [1]. Many have researched the diffusion of information or behavior adoption in social networks (generally categorized as social contagion). In general, the connections between two individuals, friendships, are extremely important in predicting behavior adoption due to the high level of trust friends have in one another [2].

Additionally, Microsoft researchers have compared the runtime of different influence maximization algorithms including Greedy and Cost-Effective Lazy Forward (CELF). They also test the runtime and optimal solution outputted by Hill Climbing algorithms that use the traditional marginal benefit method against a Degree Discount Heuristic-based approach. They showed that this approach returns a comparable optimal solution but runs much more quickly than either

greedy algorithm[4] Since the social game networks are very large, we will attempt to employ the Degree Discount approach as well as a reduced version of Lazy Hill Climbing that was inspired by Degree Discount to model influence maximization.

We have not read previous research that attempts to identify characteristics of high influence players.

3 Data

The social game we are analyzing for this project was currently has 700,000 daily active users (DAU) and just over 3.8 million total players. The company that produces this game has collected and stored all data generated by the game (e.g. user interactions, clicks, login activity) in a data warehouse. Therefore, the dataset is rich and contains a lot of data that will be unnecessary in our experiments.

We will use a subset of the social game graph that contains just users in April 2012, early in the game release. We chose this month because it is still a large network but is relatively free from inconsistencies in data that arise from users quitting the game or modifying their profile information.

3.1 Data Collection

The game collects data two ways: (1) by sending events to a dedicated analytics service in real time as they occur in the game on a per-player basis and (2) by updating player state information and aggregate data in a database on the game server. From this server, we pulled a graph of current (October 2012) friend edge connections, list of user ids that existed in April 2012, and a directed edge list of current friend connections where the destination node is a monetizing node called the Current Monetizing Graph.

Now, we build the two networks we will use for influence maximization. From the current friend connection network, we pull all nodes that existed in the April node list. We then add all the edge connections between this subset of nodes. This graph is the April graph where directed edges indicate that the source node invited the destination node to the game. The second network is the subset of the Current Monetizing Graph that existed in April. Here, the directed edge implies that the source node invited and successfully brought in the destination node and that the destination node is a monetizing node. However, the source nodes are not necessarily monetizing nodes. Given a scenario where a high level, non-monetizing player invites a new player. The new player may feel incentive to purchase in-game items in order to compete with his or her high-level friend. Therefore, we do not require source nodes to be monetizing nodes. This is the motivation for feature analysis on influential nodes rather than monetizing nodes.

For analyzing features of influential nodes, we pulled some player attributes from the company's database. Table 1 is a summary of the player attributes we retrieved.

In order to make the data available to our graph analysis software, we store the data in a combination of tab-delimited files and serialized Python dictionaries (using pickle).

Our current player data set includes all players identified by unique IDs, purchase behavior e.g. monetizing status, some activity attributes, and number of invitations sent and responded to. Of the invitations sent, there are 2 primary types: (1) Direct: direct messaging between players using the social platform's integrated messaging system and (2) Spam: broadcast messaging to all friends of a player using the social platform's "wall".

3.2 Initial Observations

The graph of April players has 123,954 players. The number of nodes with zero in-degree, players who found the game through banners or ads rather than friend invite, is 13,181. The average degree (in and out) of nodes in the graph is 4.

We note that the in degree of each node can be greater than one. While each player ultimately accepts at most one invitation, they may receive multiple invitations before accepting. Using in degrees greater than 1 is reasonable since the combination of invitations is what ultimately influences a node to join the social game.

	Represents	Attributes
Node	Player (unique ID)	-purchase behavior (monetizing) -participation in company loyalty program - login streak -total play time -sessionCount -number directed invites sent -number spam invites sent -number direct invite responses -number spam invite responses
Edge	Player Interactions -social message sent/received -friendship (i.e. edge only exists if a social message was sent and accepted)	-source node (inviter) -destination node (invited)

Table 1: Summary of node and edge attributes

The other main data we are looking at is the player transaction data. The game sells bundles of in-game currency for Facebook Credits that players purchase with USD. Players can then use the currency they purchased to buy in-game items. Our data set shows that there are 37,245 current players who have purchased currency bundles for real money and that the range of amounts spent are 35 FBC to 24,250 FBC (over the life of the game) with an average of 319 FBC and a standard deviation of 675 FBC.

We find that the growth of the graph is somewhere between a Preferential Graph (PG), leaning towards a Real World Graph (RWG). Figure 1 shows the degree distribution plot for the April graph. This type of growth makes sense since: there will be high clustering among the nodes since a significant portion of the people joining the game are through invites from friends (PG nature, node joining an existing node set) while the other significant portion of people joining are through advertising by the game publisher (RWG nature, new node through advertisements).

Due to the nature of the graph we believe that we can effectively convert non-paying customers in the gaming network by pinpointing the true influencers, and giving them incentive and tools to influence.

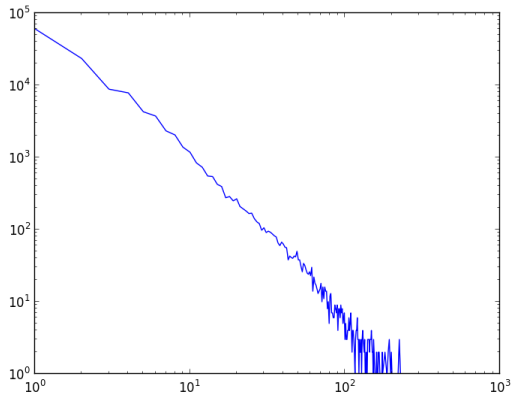

Figure 1: plot of degree distribution proves April graph follows power law. Expected degree is 4

Incentive example: Give an high influencer the option to earn X paid credits by distributing matching number of X credits e.g. 5 each to X/5 friends of his which are non-paying.

Tool example: The option to distribute credits is the tool of influence that brings in new users.

Our Influence Maximization algorithm will look to optimize incentive distribution in the network as incentives may be limited. We want to try to reach the maximum number of non-monetizing, either not in the game yet or in game and non-monetizing, players possible while giving incentives to only a limited number of players.

3.3 Extracting True Values

To find the true nodes of highest influence, we calculate the size of each node's monetizing descendent set size, the number of monetizing nodes each node can reach. We use a Breadth-First search (though a depth first search works as well) and count the number of monetizing nodes encountered during the search. Because many nodes can reach the same number of monetizing nodes due to the relatively small number of monetizing nodes in the graph, we do not rank nodes individually. Instead, we allow ties so that nodes with the same monetizing descendent set size have the same rank.

4 Experiments

Implementing the algorithms does not require much mathematical background. In general, influence maximization is a NP-hard problem (reducible tn vertex cover) so we only use approximations. Feature analysis on features that affect a node's influence probability requires some basic machine learning knowledge in order to interpret the results

4.1 Algorithms

4.1.1 Lazy Hill Climbing

To identify influential friends, we use the Lazy Hill-Climbing algorithm (LHC), a faster version of the traditional greedy Hill-Climbing algorithm. We define the influence of a node to be the number of nodes it can reach and thus influence. Briefly, the greedy Hill-Climbing algorithm is repeatedly identifies the most influential node by finding the node with maximum marginal benefit with respect to our current seed set then adding that node to our seed set. We repeat this until the seed set reaches a target size. We calculate the marginal benefit, delta, of each node u using the following equation:

$$\delta_u(S_i) = f(S_i \cup \{u\}) - f(S_i)$$
$$where\ S_i\ is\ the\ set\ of\ currently\ selected\ nodes$$

However, greedy hill-climbing will not run efficiently on our dataset because it will recompute the marginal benefits for all nodes in the network for each iteration.

Therefore, we implemented LHC. LHC utilizes the fact that marginal benefits only shrink as the set S grows. Instead of recomputing the marginal benefit for all nodes, we keep an ordered, descending list of marginal benefits from the previous iteration. We then re-evaluate the marginal benefit for the node with the max marginal benefit from the previous iteration and re-sort the list.

If the top node's new marginal benefit is still greater than the marginal benefit (from the previous iteration) of the other nodes, then we add this node to the seed set since the marginal benefit of the remaining nodes cannot increase past this node's marginal benefit. If the top node's marginal benefit is not greater than the marginal benefit from the previous iteration of some other node, the algorithm will recompute and resort until one of the top nodes remains the top node after the re-sorting.

4.1.2 Discount Degree Heuristic Selection

To optimize the generation of each node's influence set, we implement Discount Degree Selection (DDS)[4], a non-greedy influence maximization optimization. DDS is a modification on the pure degree heuristic that is commonly implemented for selecting seeds for influence maximization. The idea behind DDS is that if a node v has a neighbor u that is already in the seed set, v's degree, the nodes it can influence, should be reduced since u being in the seed set already has lowered v's helpfulness (case: edge v → u, where u is in the seed set).

Now the benefit of adding the node v into the seed set is reduced to a relatively simple function of its out-degree, neighbors, and probability of influence. From [4] we have,

For each node v, calculate:
t = num neighbors in seed set
out = out degree
*p = according to equation from 4.2.*1
then: *marginal benefit = out - 2*t + (out-t)*t*p*

We will run the lazy hill-climbing algorithm using this heuristic instead of the previous

definition of “marginal benefit”. This modification should greatly reduce the runtime but we will experiment to see if using this heuristic results in a comparable seed set on our network.

Additionally, DDS is able to ignore the multi-level influence set due to the assumption that the probability of influence, p, for the nodes in the graph is very small (around 0.01). We will see in the Experiments section that this is an accurate assumption for this network.

4.1.3 Lazy Hill Climbing-Direct Influence Set

Since DDS can ignore the benefit from indirect descendents due to its assumption that the probability of influence is very small, we decided to test modified version of LHC based on the same assumption. Since the probability of any given node influencing another is very small, we assume that in general, nodes are only able to influence their direct neighbors (since the probability of influencing a neighbor and that neighbor influencing another node should be very small).

Now, we restrict the influence set of each node to only the direct neighbors it can influence. We expect the runtime to decrease significantly since we no longer need to recursively calculate influence sets. However, the results of the algorithm will determine if ignoring the effects of indirect descendents is an appropriate assumption. We describe the implementation of this algorithm in the Experiments section.

4.1.4 Best First for Feature Analysis

We use a best first algorithm that chooses features using a greedy hill climbing algorithm with backtracking capabilities. Best first starts with the empty set of features and adds new features to the subset of selected features. We will use best first in conjunction with correlation feature selection (CFS). CFS evaluates each subset of features and returns higher scores for subsets that contain features with high correlation to the class labeling but low correlation between features.

4.2 Experiments

4.2.1 Choosing best algorithm

When running LHC, LHC-direct and Degree Discount algorithms, we want to determine if a particular algorithm is consistently better than the other. We use multiple metrics to determine the conditions upon which we would recommend each algorithm to a social gaming company.

We calculated each node’s probability of influence by counting the number of monetizing friends it has and dividing by the total number of invitations sent regardless of type:

$$p = \frac{number\ of\ monetizing\ friends}{total\ number\ of\ game\ invitations\ sent}$$

The highest influence percentage (percentage representation of p) was .33. However, over 99% of the nodes had a probability of influence that was less than 0.2. Therefore, we are able to test degree discount and LHC-direct knowing that, in general, the p value is small.

4.2.2 Feature selection for probability of influence

The probability of a node v influencing a neighbor to monetize is dependent on multiple features. The node v’s activity, level in the game, amount of total playtime, and other attributes may affect the probability of influence. Additionally, edge properties of invitation (spam versus directed) also affect the probability.

Since we want to determine the characteristics of high influence players, we will use feature selection analysis tool, weka, to determine how important each feature is with respect to determining the players’ p value. We will identify a subset of features that is relevant to determining probability of influence while also identifying some features that are irrelevant.

4.3 Results

4.3.1 Influence Max Results - Timing

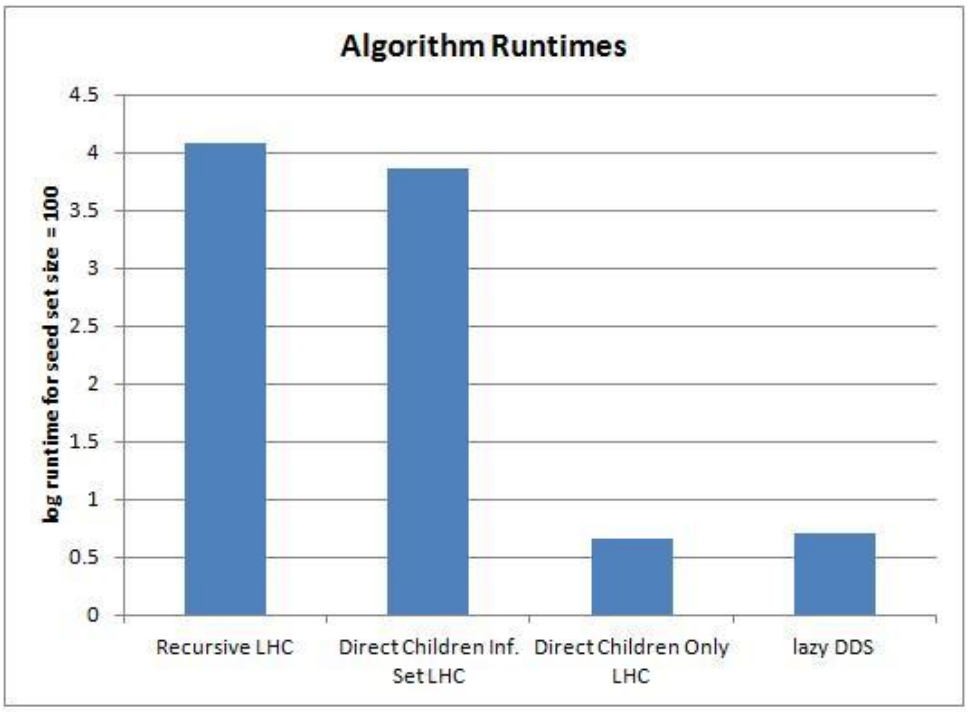

Figure2: Runtimes for IM Algorithms

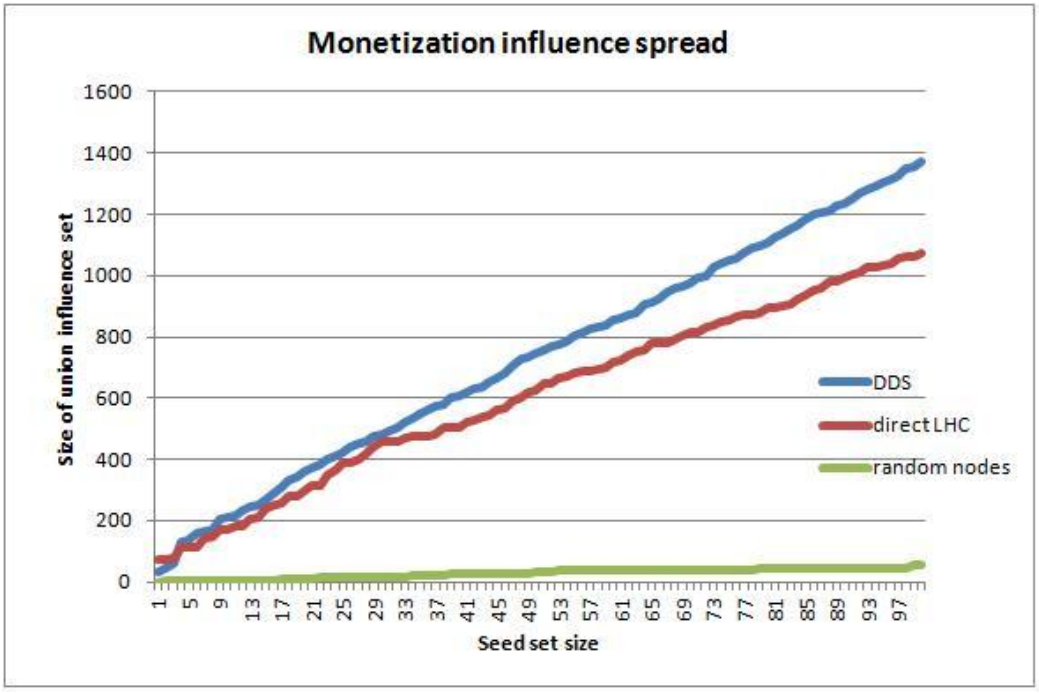

Figure3: Seed Set Size by Monetizing influence set size

The traditional version of lazy-hill climbing used recursion to generate each node's influence set. We also tried with a slight modification to the recursive algorithm just for timing purposes. The second version generated each node's direct influence set (influenced neighbors) then concatenated the influenced neighbors' influence sets and so on to generate the full influence set

The timing comparison for both traditional LHC implementations along with LHC-direct and degree discount is shown in Figure 2.

As expected, we found that the direct influence set version of LHC ran significantly faster than both traditional implementations since we did not need to generate the influence sets recursively.

We were not surprised to see that degree discount ran significantly faster than the traditional LHC and comparably to LHC-direct. But we were surprised to note that traditional LHC ran very quickly after generating the initial influence sets. We observed that the top node usually remained the top node even after the reevaluating marginal benefit. This explains why the lazy hill climbing portion for traditional LHC is fast since re-evaluation is only performed for a few nodes.

For traditional LHC, the majority of the run time was spent generating the influence set for each of the nodes. This bottleneck has order n^2 run time as each of the nodes in the graph must execute a depth-first search (DFS) starting from that node to get the individual nodes' influence sets.

4.3.2 Influence Max Results – Influence Set

To analyze the performance of the influence maximization algorithms, we plotted the seed set size versus the influence set size. The results are shown in Figure 3.[1]

As expected, both LHC-direct and degree discount performed much better than the baseline of choosing random nodes. Additionally, the performance of LHC-direct and degree discount was comparable. However, degree discount consistently chose nodes that generated a larger influence set size than LHC-direct.

Since LHC is supposed to be more accurate than degree discount, we attribute LHC-direct's performance to calculating influence set from only direct neighbors. The assumption that we can generally represent influence set for each node using direct neighbors only is too naive. Because LHC-direct loses a lot of information about the nodes such as degree and original neighbor set, the influential nodes the algorithm picked were not always optimal.

[1] It was infeasible for us to generate graph data for traditional LHC during this project given our resources and the algorithm runtime.

4.3.3 Feature Analysis Results

Using the true probability influence values as the labeling of each node, we analyzed the relevance of each of nine features:

- number directed invites sent
- number spam invites sent
- number responses from directed invites
- number responses from spam invites
- monetizing status (true or false)
- total play time
- participation in company loyalty program (true or false)[2]
- session count (number of logins)
- login streak (number consecutive days)

The subset returned by best first using CFS is:

(1) monetizing status
(2) session count

Therefore, our hypothesis monetizing status not necessarily affecting node influence probability is inaccurate Session count is chosen as the other correlated attribute. Performing attribute selection without the session count attribute returns (1) monetizing and (2) total play time. This result is expected since the number of logins should be highly correlated with the amount of time played. However, session count may be a better feature because there may be players who played a lot when they first joined but did not continue playing the game over time.

We then ran the feature selection algorithm again without the monetizing attribute and found that the best subset of features was:

(1) number of responses to spam invites
(2) session count

This is interesting because the feature subset does not include direct responses. Feature selection without the spam responses still does not pick direct responses as a feature. We believe this is due to the higher volume of responses

[2] We were surprised to note that participation in company loyalty program had no correlation to the probability of influence since those players would generally be more active. We then observed that the values for that feature were all "true". This is an error in data we received from the company.

received due to spam invites as opposed to direct invites.

Finally, we ran feature selection without monetizing status and session count. The resulting subset is:

(1) number directed invites
(2) number spam invites
(3) number directed responses
(4) total play time

We expected total play time in the subset but we were surprised to see the first three features instead of number spam responses. First, we now know that the number of directed and spam invites are not very correlated. This makes sense because players who enjoy spamming their friends would generally not also send direct invites. The opposite is also true where players who select individuals to invite would generally not also spam the rest of their friends. Second, this probably implies that the number of spam responses is correlated with total play time and/or number of spam invites since they do not appear together in feature subsets. This conclusion makes sense because players who spend more time in the game would probably send a higher volume of invites and receive a higher volume of responses as well.

5 Summary

5.1 Conclusion

Our timing results are consistent with our expectations. Both degree discount and LHC-direct run significantly faster than traditional LHC. However, we are only able to justify the use of degree discount and LHC-direct because we know that the true probabilities of influence for nodes in the graph are generally small so we are able to ignore non-neighbor influence.

Overall, we find that degree discount and LHC-direct algorithms give similar influence set sizes. However, degree discount performs slightly better at selecting influential nodes than LHC-direct. Therefore, we do not recommend LHC-direct for influence maximization because degree discount heuristics produce better results with similar runtime.

In feature selection, we concluded that monetizing status and session count composed a good feature set for determining a node's probability of influencing a node to join the game and monetize. However, we also determined that some of the features were correlated with session count and could also be used to predict a node's p value

5.2 Future Work

This project was only able to run on the social network data. It would be interesting to run similar algorithms for influence maximization and feature selection on the underlying social network data. Experiments in influence maximization on social network data would allow us compare the intersection of the high influence nodes at bringing in new users in general versus new monetizing users. Since companies are highly interested in increasing their monetizing user base specifically, such comparisons would be beneficial.

Running feature selection on a larger feature set that includes geographical location, interactions on the underlying social network, user's social network statistics, etc. would be produce more insight into the characteristics and attributes of high influence nodes in the social game.

References

[1] Ugander J, Karer B, Backstrom L, Marlow C (2011). The Anatomy of the Facebook Social Graph. 1-14.

[2] Guha R, Kumar R, Raghavan P, Tomkins A (2004). Propagation of Trust and Distrust. 403-412.

[3] Szeil M, Lambiotte R, Thurner S (2010). Multirelational Organization of large-scale social networks in an online world. PNAS

[4] Chen, W., Wang, Y., and Yang, S. Efficient Influence Maximization in Social Networks. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*: 199-209, 2009