Business Expansion Engine

-By Souham Biswas, CWID:A20365242

1 INTRODUCTION

In this project, a software has been developed which aims to enumerate those persons inside a user's social network who are most susceptible towards adopting a given product or service and listing those people who, if adopt the product/service initially, would result in the maximum spread of the product/service. The software also analyzes the networks of the persons in the user's network and recommends a set of 'initial adopters' (who are in essence, given a 'free sample' of the product/service) such that the popularity of the product spreads to a maximum extent in the user's network.

For example, if you want to sell an iPhone, you would enter the keywords 'Apple', 'Gadgets' etc.., your twitter id and the number of people you are willing to give 'free promotional iPhones' to. This software would output a list of user ids in your network whom if given 'free iPhones', would result in most number of people in your network actually buying iPhones.

A prime challenge encountered in the development of this software was to devise a way to compute the payoff of a given product/service for a given user based on the keywords/tags pertaining to the product/service and the pages followed by the user. Also, only targeting the persons having the highest payoffs was not sufficient. To qualify as an adopter, a user also needs to have a network of people which is sufficiently large and also that the persons in his/her network should have a sufficient payoff to ensure maximum spread.

Hypothesis-

The approach embodied by this project involves probing the network of the user in question and computing the payoffs of each person on the network based on the pages they follow. The approach to compute the payoff involves searching Bing, the string formed by concatenating the page name followed by a user and one of the keywords/tags initially taken as input with the keyword 'and' in between. Subsequently, all the search results are analyzed to get a measure of relevance between the two words.

For example, if a user follows a page 'Steve Jobs' and the keyword is 'Apple', then string 'Steve Jobs and Apple' is searched. By intuition, we can imagine that a high number of occurrences of 'Apple' and 'Steve Jobs' being close to each other will be found in the bag of tokens suggesting a high relevance.

This is repeated for every keyword and every page a user follows to obtain a 'payoff'. This payoff is assigned to each person in the user's friend list and a graph embodying the friend network is created. Finally, a brute force search is, a cascade simulation is performed; and the set of initial adopters which result in a maximum spread, is returned.

2 DATA

The data required for proper functioning of the software includes the following –

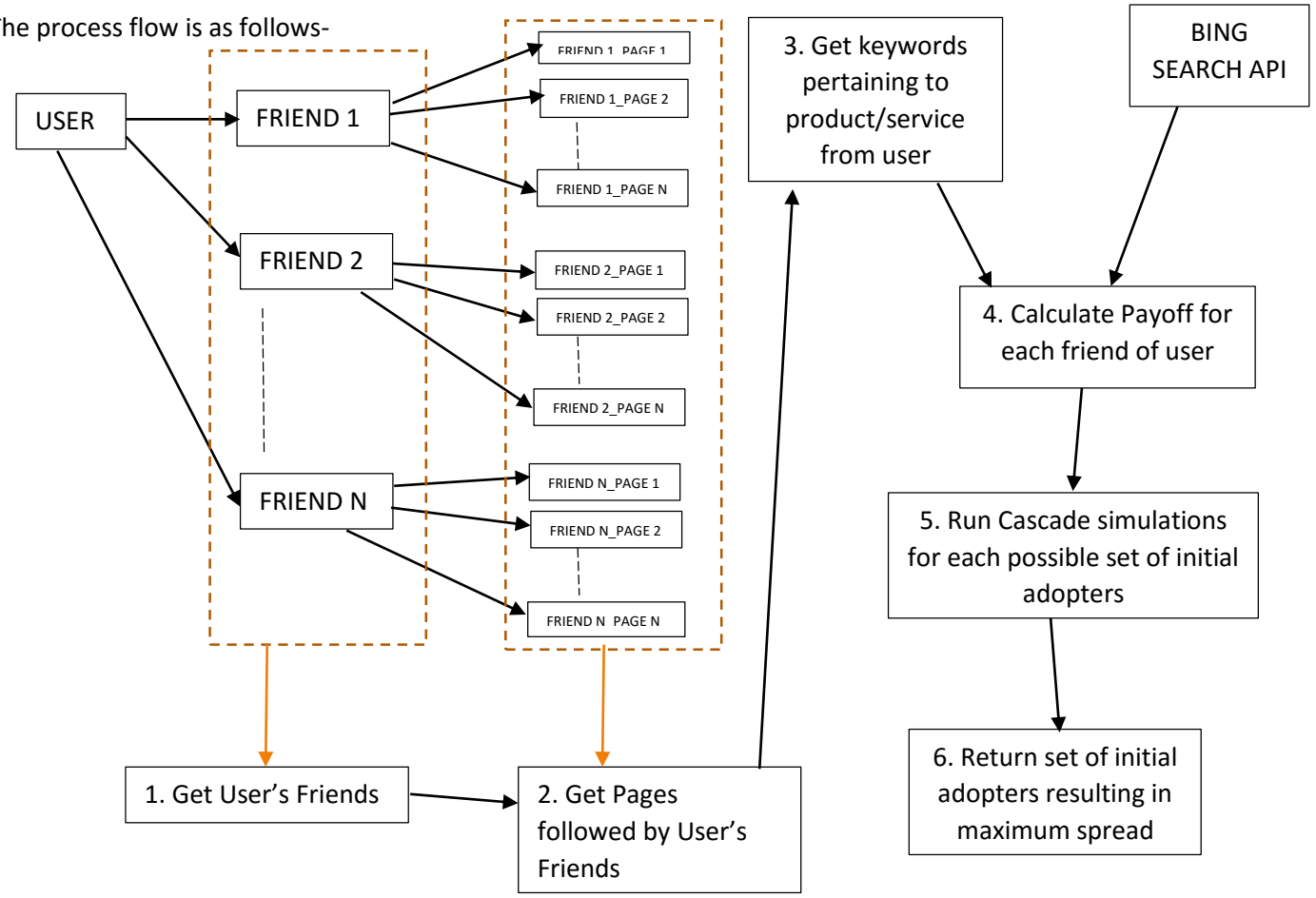
- Friend List of the user.
- Pages/People followed by the friends of user.

The data was collected using the Twitter Python API. The collection of the user's friend list is no problem. However, to collect the followings list of the friends of the user, one request is required for

each friend. Therefore, this process can be time consuming due to the 15 requests/minute rate limit of the Twitter API which limits us to making only 1 request per minute.

3 METHODS

The process flow is as follows-



The method outline is presented above. The procedure for calculation of the payoffs is explained below-

Suppose the keywords entered by the user are $K = \{k_1, k_2, k_3 \dots \dots k_n\}$ and the user has friends $F = \{f_1, f_2, \dots \dots, f_m\}$. Let friend f_i follow pages $P^i = \{p_1^i, p_2^i, \dots \dots, p_k^i\}$. To calculate the payoff for f_i, we follow the following procedure-

1. For each keyword $k_a \in K$, and page $p_b^i \in P^i$ we perform a search with search string "k_a and p_b^i".
2. Then, the token list T_{ab}^i returned by the search is pruned to derive the number of times $c_{a,b}^i$, the terms k_a & p_b^i occurred within a constant linear distance from each other (distance set by variable $ngram$ in method `calculateIndividualPayoff` in file PayoffCalculator.py).
3. This is repeated for each element in set K get a list $L_b^i = \{c_{1,b}^i, c_{2,b}^i \dots \dots, c_{n,b}^i\}$ for every page $p_b^i \in P^i$.
4. These lists are then combined to make a 2D matrix M^i by placing each element of the set $\{L_1^i, L_2^i, \dots \dots \dots, L_k^i\}$ (for each page in $P^i = \{p_1^i, p_2^i, \dots \dots, p_k^i\}$) in different rows.
5. We basically get a matrix like this –

$$M^i = \begin{bmatrix} c_{1,1}^i, c_{2,1}^i & \cdots & c_{n,1}^i \\ \vdots & \ddots & \vdots \\ c_{1,k}^i, c_{2,k}^i & \cdots & c_{n,k}^i \end{bmatrix}$$

6. Then, we normalize each column by dividing each columns entry by the RMS value of all the column elements. Therefore, if $c_{x,y}^{i}{}'$ is the normalized value of $c_{x,y}^{i}$, then the following relation holds-

$$c_{x,y}^{i}{}' = \frac{c_{x,y}^{i}}{\sqrt{\sum_{j=1}^{k}\left(c_{x,j}^{i}\right)^{2}}}$$

7. Finally, the mean of the normalized matrix is computed to yield the payoff for f_i. This is repeated for every friend of the user.

After this, a graph is created with its vertices being each friend of the user. The vertices have two attributes namely, "Payoff" which store the payoffs and "Choice" which is set to 'B' initially (assuming no one has adopted the behavior which is intended). The behavior favorable to the user's intent is 'A' which is assigned to the set of initial adopters.

A brute force search is conducted to find the optimal set of initial adopters. That is, based on the number of desired initial adopters entered by the user (numAdopters or in essence, the number of 'free iPhones, you are will to give away) a cascade simulation is run until the spread stops increasing for each combination of initial adopters of size equal to that specified by user and the set resulting in the maximum penetration is returned.

4 EXPERIMENTS

This software mostly depends upon the accuracy of the method which computes the payoffs. Therefore, a number of experiments were carried out to test the validity of the hypothesis. Running one experiment iteration can take up to 11-12 hours which also includes the data collection time.

The experiment included probing my own twitter network of friends to identify the maximal set of adopters for a an app which gives news pertaining to celebrities, gossips, parties etc.

Therefore, the keywords chosen were -

```
'celebrity', 'famous', 'party'
```

The number of initial adopters was set at 6, with my twitter handle 'ironhide23586'. The following results were obtained –

```
Found a total of 4 cluster(s) of size(s) [22, 7, 3, 2] -
Result 1
Cluster size = 22
User ids of initial adopters resulting in max spread in this cluster - [u'Deepanshu201
4Dt', u'kulbeer_sidhu', u'Funnyoneliners', u'sapan_u', u'coolnitesh09', u'manoharvvgol
la']

Spread Amount (Number of people spread to, apart from initial adopters) = 9
User ids of people spread to- [u'kulllshubh', u'Dracodos', u'TheTeKKI', u'VineetMshr',
 u'Shiwani14393', u'manishjha_', u'shkkonda', u'VibhoreAgarwal1', u'Raven_krishna']
```

Fig. 1 Network with maximal initial adopters

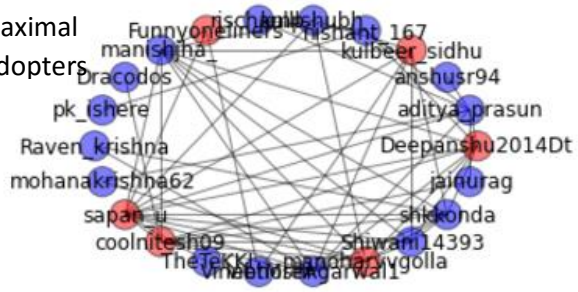
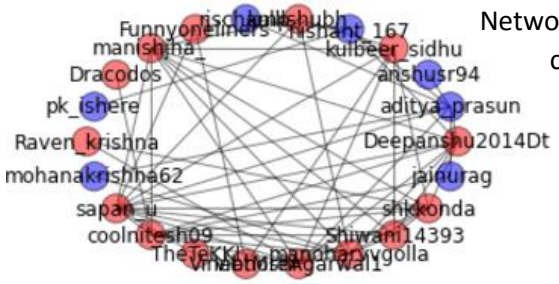


Fig. 2 Maximal Network spread after cascading




```
Suggested set of intial adopters for maximum penetration - [u'Deepanshu2014Dt', u'kulb
eer_sidhu', u'Funnyoneliners', u'sapan_u', u'coolnitesh09', u'manoharvvgolla']
```

As we can see, the red nodes in Fig. 1 and Fig. 2 are those affected by behavior 'A'. The initial adopters in Fig. 1, result in the spread illustrated in Fig. 2.

To verify these results, the users apart from the initial adopters to whom this behavior spread to, were asked if they would be interested in buying such an app and out of the 9 people namely - [u'kulllshubh', u'Dracodos', u'TheTeKKI', u'VineetMshr', u'Shiwani14393', u'manishjha_', u'shkkonda', u'VibhoreAgarwal1', u'Raven_krishna'] around 6-7 people replied 'yes' which translates to around a 75% accuracy.

5 Related Work

There is significant work in this field. However, most of the work being done depends upon collecting transactional data from users pertaining to their search history and various other factors relying on compute intensive fancy data mining algorithms. They work mostly to identify cohesive interest groups of a user and target specific ads towards him/her.

A few references are-

- Yang, Wan-Shiou, Jia-Ben Dia, Hung-Chi Cheng, and Hsing-Tzu Lin. "Mining social networks for targeted advertising." In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, vol. 6, pp. 137a-137a. IEEE, 2006.
- Ribeiro-Neto, Berthier, Marco Cristo, Paulo B. Golgher, and Edleno Silva de Moura. "Impedance coupling in content-targeted advertising." In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 496-503. ACM, 2005.

In this project, a different approach has been embodied which simply uses a search API and does not depend upon fancy algorithms to perform the work. Moreover, it gives suggestions to spread the business within one's personal network as opposed to general resource intensive advertising. Advertising within one's own social circle has a higher probably of spreading as compared to advertising to the general mass. This software further maximizes the chances of spread by recommending the best set of initial adopters.

6 Conclusions And Future Work

A novel method to compute the payoff of a given product for a person has been presented and implemented. Also, it has been explicated that the cascading concepts apply in the real world and that an inverse form of its classical implementation (of computing the spread given initial adopters) can be leveraged to maximize spread of a behavior in a network. The methodology can be made better by incorporating nltk library and good NLP algorithms which although, was tried in this project but was discarded due to insufficient compute power. Performance may be further improved by GPU-based Deep Learning technologies and Recurrent Neural Networks for advanced Natural Language Processing.