Recipient suggestion for electronic messages using local social network data

Andrey Gomzin Stepan Ipatov Anton Korshunov Hangkyu Kim

Institute for System Programming of Russian Academy of Sciences,
Data Intellegence Lab, DMC R&D Center, Samsung Electronics Co., Ltd.
{gomzin, ipatov, korshunov}@ispras.ru, hangkyu.kim@samsung.com

Abstract

Social network users often want to send a message to a group of recipients. An algorithm that recommends other possible recipients given an initial set of recipients is introduced in this paper. The algorithm uses different types of local user data: profile, friendship graph, posts, and social interactions (likes, comments, tags). Experimental evaluation using Facebook application demonstrated that the algorithm is able to make suggestions meaningful to experts.

1 Introduction

In today's world people communicate by messages using Internet. Often they send the same message to several recipients. Usually, it's done by selecting them one-by-one by typing their names. But in most cases users have a kind of communication patterns: they communicate with more or less stable groups of users. Moreover, a group of recipients is typically related to some common group's property. For example, recipients work in the same company, or graduated the same university.

The target data domain of our work is *Facebook*[1]. Our application analyses user and user's friends data: profiles, walls and friendship graph, and provides recipient suggestions for the message, given a current list of recipients.

In our application[2] the user is supposed to select the recipients as follows:

1. The user enters a friend's name as first recipient
2. The application provides other probable recipients from the target user's friends based on the relevance to the current list of recipients (seed set).
3. The user can:
   - select recommended user and add her to the seed set
   - type a friend manually like as in step 1
   - send the message
4. Go to the step 2 if the message has not been sent yet

So, in this paper we present an algorithm that recommends more recipients of the message given an initial set of recipients.

2 Related work

Recommender systems [6] are applied in a variety of applications. They help people to find interesting items, such as music, books, videos, etc. Also they may serve as tools for friend finding in social networks.

The most canonical and simple method which is used in recommender systems is Collaborative filtering [9]. These systems infer recommendations using ratings that users give to items.

Modern recommender systems use more data, than user-item ratings. For example, social networks provide plenty of data, such as friendship relations between users, user profiles, interactions between users and so on. The task of new friends recommendation is solved in [1]. Authors proposed an algorithm that builds users recommendation using an implicit rating model, which uncovers the strength of relationship, utilizing both attribute similarity and user interaction intensity. But only interactions between two users are considered in this paper.

One of the recommender tasks is recipient suggestion. This kind of recommender system is widely spread in email services: email clients suggest multiple recipients of grouping message. There are several approaches for predicting message recipients depending on input data.

The method considered in [8] uses history of grouping mail senders and recipients. The input is initial set of users. The task is to suggest more users that may be potential recipients. Suggestions satisfy the following criteria: groups a user interacts frequently with are more important to the user than groups she interacts infrequently, group importance is dynamic over time, interactions that the user initiates are more significant than those she did not initiate. Authors consider several methods to calculate user's score. Described method dosen't consider composed message text.

In other works, messages content is analyzed to predict a group of recipients. The most of approaches are based on machine learning. Method, described in [2], finds the most similar messages for a given message, using TF-IDF. Recency and frequency of the messages are considered too. TF-IDF have been compared with

Proceedings of the Tenth Spring Researcher's Colloquium on Database and Information Systems, Veliky Novgorod, Russia, 2014

[1] http://www.facebook.com
[2] http://rs.at.ispras.ru

K-nearest-neighbour method in [4] Authors of [7] consider enhanced Naive Bayes model. Described method takes into account message body words, subject words and other recipients. These methods also don't consider social relationships between users as data source.

All described recipient suggestion algorithms don't consider social data, such as relationships between users (for example, friendship). Moreover, only one type of interaction (email sending) is considered in these methods. Social data contains different types of interactions between users (messages, user tags, comments, etc.). Our method is aimed to use all available social data to predict recipients of the message.

3 Recipient Suggestion algorithm

Our recipient suggestion algorithm is based on Interaction Table. The workflow contains 3 stages:

1. Retrieving Facebook data
2. Finding co-occurrences of users (co-likes, co-comments, graph communities and so on) from Facebook data and building corresponding interactions (raws of Interaction table)
3. Suggesting for the most relevant users given built interactions and seed set.

3.1 Terminology

- **User** is an owner of Facebook account.
- **Profile** is a set of pairs (field name, field value). Profile contains an information about a *user*.
- **Post** is a public message of a *User*. *Post* contains a text, a photo, a video, or a link.
- **Wall** is a sequence of *posts*. Each *user* has one *wall*
- **Target user**, **target** – the author of a message, the "target" of the recipient recommendation.
- **Ego-network** of the *target user* is the *target* node, the nodes to whom the *target* is directly connected to (Facebook friends) plus the connections between these nodes.
- **Seed set** – recipients of a message provided by the *target user*
- **Comment** is a text reply for *post*. *Post* contains *comments* from different *users*
- **Like** is positive feedback and an indicator that the *user* cares about the (*post*).
- **User tag** – user mention in the *post* (text, tags on photo, etc.)
- **Community** – a group of users that are more densely connected to each other (by friendship relation) than to the rest of the *target* user's *ego-network*.
- **Interaction** is a record about an action related to the group of users (*posts*, *comments*, *likes*, *user tags*, *community* membership, etc.). See section 3.3 for more details.
- **Interaction Table** is a set of *interactions*

3.2 Data collection

We use the following Facebook data

- Profiles of users
- Local Friends Graph (ego-network)
  - Target user's friends list
  - For each friend: target user's and friend's mutual friends
- Target user's and her friends posts with
  - Comments
  - Likes
  - User tags
  - Text content

The data is retrieved using Facebook Graph API[3], given target user id.

3.3 Interaction table building

In this section we describe how interaction table is built from Facebook data.

All interactions contain a set of users involved into interaction. Here is a list of all possible *interaction types*:

- *profile*: An interaction contains users with the same fields' values;
- *target_target*: An interaction is built from target's posts on the target's wall (likes, comments, tags are analysed);
- *target_user*: An interaction is built from other users' posts on the target's wall (likes, comments, tags are analysed);
- *user_target*: An interaction is built from target's posts on other users' walls (likes, comments, tags are analysed);
- *user_user*: An interactions is built from other users' posts on other users' walls (likes, comments, tags are analysed);
- *other*: An interaction is built from posts on walls that are not related to the target user (likes, comments, tags are analysed);
- *content*: An interactions is built from target's posts on the another users' walls with the same content;
- *community*: An interaction includes members of the same community.

The hierarchy of interaction types is shown in figure 1. *Wall Interaction*s are built using post's comments, likes, tags or content. These interactions have timestamp and *content type* attributes. *Content type* has 4 possible values: status, link, photo, video. *Post Interaction*s are built using post's comments, likes or tags. They have *post interaction type* attribute with 3 possible values: comment, like, user tag.

[3] https://developers.facebook.com/docs/graph-api/

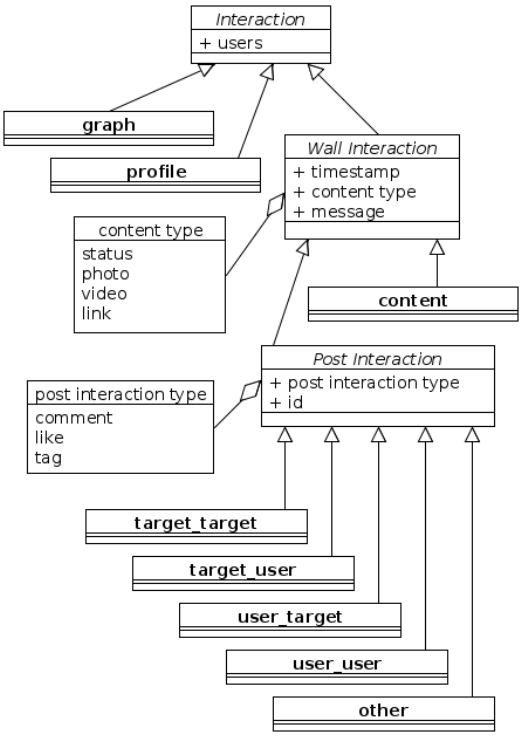

Figure 1: The hierarchy of interaction types

Target user's wall interactions

First, consider target user's wall. Author of the post is important, because she is also included into the interaction. Here we distinguish two possible options: post's author = target user and post author ≠ target user.

Post author = target user. If post contains comments/likes/tags, a *target_target* interaction is created. Interaction contains all users that commented/liked/is tagged (on) the post

post author ≠ target user. If post contains comments/likes/tags *that include target user*, a *target_user* interaction is created. Interaction contains all users that commented/liked/is tagged (on) the post and post *author*. In case of post contains comments/likes/tags that *don't include target user*, an *other* interaction is created.

Interaction type	Post interaction type	Post content type	Timestamp	Users
target_target	comment	photo	5.10, 12:06	Tom Spike
target_target	like	photo	5.10, 12:06	Tom Jerry Spike
target_target	user tag	photo	5.10, 12:06	Tom Jerry

Table 1: Interactions built from the post shown in figure 2

For example, 3 interactions are created from the post in figure 2. These interactions are shown in the table 1.

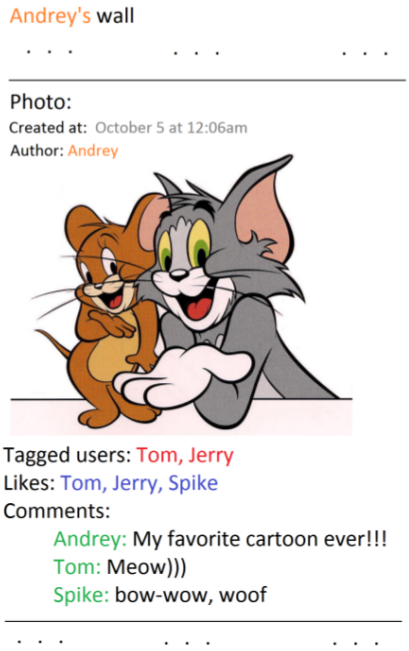

Figure 2: Target's post on the target's wall example

Non-target user's wall interactions

As in the case of processing target's wall, two possible options are considered: post author = target user and post author ≠ target user. Consider a wall of any non-target user. Let's call her a *current* user.

Post author = target user. If a post contains comments/likes/tags, a *user_target* interaction is created. Interaction contains all users that commented/liked/is tagged (on) the post and the *current* user.

Post author ≠ target user. If a post contains comments/likes/tags *that include target user*, a *user_user* interaction is created. Interaction contains all users that commented/liked/is tagged (on) the post and post *author*. In case of post containing comments/likes/tags that *don't include target user*, an *other* interaction is created.

Target posts text content interactions

Consider target user's posts on another users' walls with non-empty message. These posts are grouped by the same text content (duplicated messages). For each group of $size > 1$ a *content* interaction is created. Interaction contains target user's recipients of the messages of the group.

Graph-based interactions

Target user contains a list of her friends. Other users contain lists of mutual friends with target user. These connections between users form target user's ego-network. The ego-network is used for finding communities of target's friends using Speaker-Listener Label Propagation Algorithm (SLPA) [10]. This algorithm is very fast. It provides high-quality overlapping communities of users. Also, SLPA supports local networks (ego-networks). For each discovered community a *community* interaction is created. Community members are included into this interaction.

Profile-based interactions

User profile is a set of pairs (F, V), where F is a field name, V is the value of the field F.

For each possible pair (F, V) an interaction is created if > 1 users have the field F with value V in profiles. The interaction contains all such users.

We distinguish two types of profile fields:

- ***Single***. Only single field value is allowed. We consider the following single Facebook profile fields: *hometown, location, gender, relationship, religion, politics*.
- ***Multiple***. In this case user's profile field may have several values. We consider the following multiple Facebook profile fields: *work, work together with position, education, education together with graduating year*. For example, users that graduate the same school are included into an interaction. Additionally, the users that graduate the same school in the same year are included into another one interaction.

3.4 Users suggestion

Recipient suggester algorithm parameters include weights for *interaction types*, *content types* and *post interaction types* values.

As we have an Interaction Table and weights configuration, recipient suggestion becomes quite simple. For each user we calculate CONFIDENCE:

$$\text{CONFIDENCE}(user) = \frac{\sum_{i \in I | user \in i} \text{Weight}(i)}{\sum_{i \in I} \text{Weight}(i)} \quad (1)$$

I is a part of Interaction table, the set of interactions: $\{i | users(i) \cap \text{Seed set} \neq \emptyset\}$. Here $users(i)$ is a set of users of the interaction i.

Weight(i) is:

- $w(\text{IntT}(i))$
  - for graph interactions
- $w(\text{IntT}(i)) \times \text{timeNorm}(ts(i)) \times w(\text{ContentT}(i))$
  - for interactions obtained by finding the same content
- $w(\text{IntT}(i)) \times w(\text{PostIntT}(i)) \times w(\text{ContentT}(i)) \times \text{timeNorm}(ts(i))$
  - for interactions built from users' walls (likes, comments, tags)

$w(.)$ is a weight from configuration. IntT(i) – interaction type of interaction i. ContentT(i) – content type of interaction i. PostIntT(i) – post interaction type of interaction i. $ts(i)$ – timestamp of interaction i. timeNorm$(.)$ is a real value from 0 to 1, that is increasing function of time. We use the following function:

$$timeNorm(i) = e^{\alpha \times (ts(i) - CT)} \quad (2)$$

Here, CT is the current time timestamp. All timestamps are expressed in seconds.

Users are sorted in descending order of CONFIDENCE. Users with equal CONFIDENCE are sorted by degree (number of friends).

4 Experiments

This section describes accuracy evaluation experiments. In the first subsection we present used Recipient suggestion algorithm parameters. The following subsections are devoted to the quality evaluation.

Recommendations provided by our Recipient Suggestion algorithm are based on different kinds of source data. But the input data doesn't contain messages with recipients defined. So, we resorted to expert evaluation of the algorithm results.

In the sections 4.2 and 4.3 we present quality evaluation metrics which require experts for manual or semi-automatic test data generation.

Then the baseline algorithm is described.

The last subsection contains obtained evaluation results.

4.1 Settings

As mentioned above, user may vary impact of different interactions to user suggestions.

We did not perform experiments for automatic parameters estimation. We have adjusted application parameters manually, according to the considerations described below.

Groups of users can be formed from mix of topical and social-based principle [5]. Social groups are based on strong relationships between users. Topical groups are based on users' interests, they tend to change. We assume that most multicast messages are addressed to stable strongly connected groups of users. Thus, in our setup, SLPA communities and users united by profile attributes contribute to the CONFIDENCE most.

SLPA itself has three parameters: minimal community size (minc), number of interactions, threshold r. We use default SLPA parameters.

Content interactions are also have high weight, because author selects recipients of messages with the same content by herself.

Interactions that are related to target user's wall have higher weight than interactions on other users' walls. Interactions between non-target users (not directly related to target user) has less weight.

Application with default settings doesn't take into account content type of interactions, hence all content type weights are equal.

Interactions of "user tag" post interaction type are more important than interactions of "comment" and "like" post interaction type, because users are explicitly picked by post's author. Moreover, the impact of comment is more than likes' impact. Like is just a single click, and comment is a text message that user types, hence comment shows higher user interest to the post, that like does.

Old posts' impact in the recipient recommendation should be less than recent posts' impact. In our setup we assume that two year old posts should have weight as 0.75 of the most recent posts. The α parameter value in (2) is set to satisfy this condition.

List of application settings is shown in the table 2.

Setting	Value
Interaction types	
profile	2
target_target	1.2
target_user	1.3
user_target	1
user_user	1
other	0.2
content	1.5
community	2
Content types	
status	1
photo	1
video	1
link	1
Post interaction types	
comment	1.3
like	1
user tag	1.5
SLPA parameters	
minc	3
threshold r	0.02
number of iterations	10
Other parameters	
α in timeNorm (2)	6.3e-10

Table 2: Recipient suggestion algorithm settings

4.2 Comparing with experts suggestions

Experts are needed for quality evaluation. For this metric evaluation we've developed a special tool that shows a random seed set of expert's friends and asks to enter another $[0..5]$ users from friends list. Seed set contains two users that participate in some interaction from Interaction table. Expert should select $[0..5]$ users that are likely to be a recipients of some message, together with given seed set. The order of entered users is important: the first user is selected in assumption of current recipients are seed set, k-th user is selected in assumption of current recipients are seed set plus selected $k-1$ users. If an expert has no suggestions, the empty set of users should be specified: only sensible groups are considered in the metric. This "Association game" is repeated 20 times for each expert.

Assume that Seed set $= \{su_1, su_2\}$.

If an expert e selects one user u_1, the application calculates user suggestions given Seed set $= \{su_1, su_2\}$. User suggestions are sorted in decreasing order of CONFIDENCE (1). Let $pos_1^e(m)$ be user u_1 position in obtained sequence: an integer number from $[1..N]$, where N is number of friends. $m \in \overline{1,20}$ is a number of current iteration.

If an expert e selects users $u_1, u_2, ..., u_k$, an application calculates k $pos_i^e(m)$ values. $pos_1^e(k)$ is calculated for Seed set $= \{su_1, su_2\}$ and user u_1 as described above. $pos_i^e(m)$ is calculated for Seed set $= \{su_1, su_2, u_1, ...u_{i-1}\}$ and user u_i as described above.

Given *all* $pos_i^j(m)$ values, obtained from *all* experts and *all* expert selections (denote them as POS set), we can calculate probability of falling into top-K of user suggestions:

$$P_K = \frac{|\{pos_i^j(m) \in POS | pos_i^j(m) \leq K\}|}{|POS|} \quad (3)$$

P_K takes values from $[0..1]$

4.3 Analysing application logs

This metric is based on logging demo user's actions with our demo-application. The demo allows users to enter user name and add her into seed set. Also the demo-application provides fast calculated users suggestions as list of top-10 recommended users with higher confidence (figure 3).

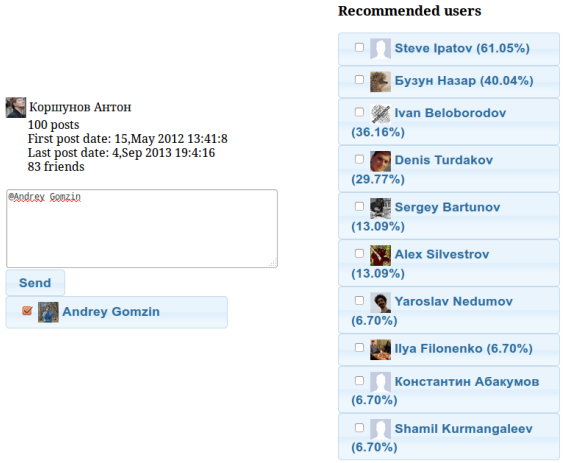

Figure 3: Demo-application

In the beginning, when seed set is empty, recommendations are not provided. Hence there are 4 different types of demo user's actions:

- **First recipient**. The demo user types the first recipient name
- **Accept the suggestion**. The demo user selects the recipient from provided list of suggested users
- **Ignore the suggestion**. The demo user types the next recipient manually
- **Send the message**. The message is sent and the seed set is cleared.

User suggestion quality metric is calculated as an average *score* value according to the demo user behaviour:

- In case of *Accept the suggestion* action, the demo user picks a user from a given top-10 with position k. $Score(a) = \frac{11-k}{10}$.
- In case of *Ignore the suggestion* action $Score(a) = 0$.

The $Score$ shows how strongly recommendations are approved by demo user. For example, if user picks first ($k = 1$) user, $Score$ is 1; if she picks the last user from the top-10 ($k = 10$), $Score$ is 0.1.

User suggestion quality is:

$$Q = \frac{\sum_{a \in A} Score(a)}{|A|} \quad (4)$$

A includes all related actions (*Accept the suggestion* and *Ignore the suggestion* action types) obtained from one application session.

4.4 Baseline

We use the method similar to the proposed in [8] as a baseline. This method recommends recipients for e-mail, given an initial set of recipients, based on group messages history. Incoming and outgoing multicast e-mails are considered in this work. Facebook private messages may be considered as an analog for e-mails. But we don't have private messages. Our algorithm provides user suggestion based on public data.

Online social network users often communicate with each other using public data: posts on walls, comments, likes, tags in case of Facebook.

We assume that *Wall interactions* with *comment* or *like* post interaction type are *incoming* messages, and *Wall Interactions* with *tag* post interaction type and interactions of *content* type are *outgoing* messages (see the diagram in figure 1). Since the authors of [8] don't consider e-mails that are not related to the target user, interactions of *other* type are not considered here.

Time decay function in [8] is the same as we use (2).

Hence, the algorithm presented in [8] applied to Facebook public data could be seen as a special case of our Recipient suggestion algorithm with the following changes in setup: interactions of *graph*, *profile* and *other* types have zero weights.

4.5 Evaluation results

We asked 10 experts to choose recipients of 20 "imaginary messages" given two initial recipients for each message, without any recommendations. And then we asked them to use demo-application which provides user suggestions.

The result of comparing algorithm suggestions with experts suggestions is shown in figure 4. The histogram shows the probability for a user selected by expert to fall into top-K users (3). The figure contains histograms for described in this paper recipient suggestion algorithm and a baseline method, described in section 4.4 and paper [8].

The log analysis shows that user suggestion quality Q, according to (4), is **0.656**.

Additionally, we have calculated user suggestion quality Q according to (4) for the data, used in P_K evaluation (see section 4.2). Here $Score(u) = \frac{11-k}{10}$ when user position is $k \leq 10$, and $Score(u) = 0$ otherwise. Obtained Q is **0.454**.

5 Results

The purpose of developed Recipient suggestion algorithm is to help people to find multiple recipients of the message. Suggestions are built using information about user and her friends (local data) in social network. The main feature is using different sources of information: profiles, communities inferred from local social graph of

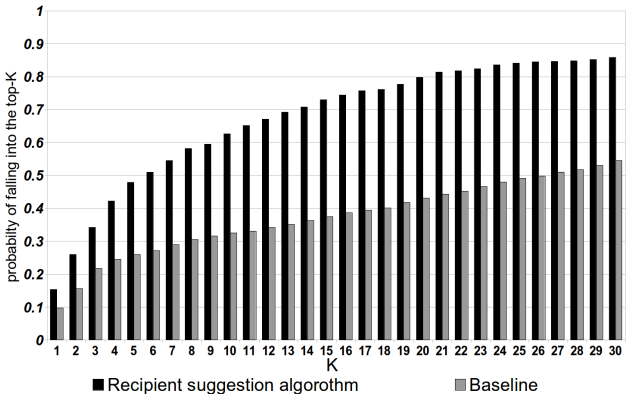

Figure 4: The probability for a user selected by expert to fall into top-K users. Recipient suggestion and Baseline algorithms

target user, and users' walls, including comments, likes, tags.

The algorithm quality evaluation is based on users' feedback. Evaluation consists of two steps. First, user chooses recipients without any suggestions. And then she uses a demo-application, that suggest users, according to developed algorithm. Results of our Recipient Suggestion algorithm were compared with baseline algorithm results. As shown in figure 4 our algorithm is significantly better than baseline. The probability of falling into top-10 is more than 60%. This enhancement is achieved by using additional data, such as ego-network communities structure and users' profiles.

Moreover, analysis of recommendation score, introduced in section 4.3, shows that average scores differ in cases of suggestions are shown to target user and are not shown (0.656 vs. 0.454). In first case the value is greater. This means that recommendations that are provided by Recipient suggestion algorithm really help user to choose message recipients.

6 Future work

Our Recipient Suggestion algorithm uses local friendship graph, users' profiles and users' walls, including comments, user tags, likes, text content. Of course, this is not full list of available Facebook user data. In our future work we plan to use more available data, retrieved using Facebook API, and use technics for detailed text analysis. Moreover, we plan to introduce group suggestions.

Here is a list of possible future directions:

- Use technics of text analysis. For example, we can use Topic Modeling [3] for analyzing topics of messages. Given topics, algorithm would recommend users that are interested in current message's topic.
- Develop an algorithm that suggests a group of users.
- Take into account period of a day: during working time colleagues are recommended, otherwise – friends.
- Develop an algorithm which automatically estimates optimal algorithm parameters based on target user's feedback.

References

[1] Vinti Agarwal and KK Bharadwaj. A collaborative filtering framework for friends recommendation in social networks based on interaction intensity and adaptive user similarity. *Social Network Analysis and Mining*, pages 1–21, 2012.

[2] Ramnath Balasubramanyan, Vitor R Carvalho, and William Cohen. Cutonce-recipient recommendation and leak detection in action. In *AAAI-2008, Workshop on Enhanced Messaging*, 2008.

[3] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[4] Vitor R Carvalho and William Cohen. Recommending recipients in the enron email corpus. *Machine Learning*, 2007.

[5] Przemyslaw A Grabowicz, Luca Maria Aiello, Víctor M Eguíluz, and Alejandro Jaimes. Distinguishing topical and social groups based on common identity and bond theory. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 627–636. ACM, 2013.

[6] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[7] Chris Pal and Andrew McCallum. Cc prediction with graphical models. In *CEAS*, 2006.

[8] Maayan Roth, Assaf Ben-David, David Deutscher, Guy Flysher, Ilan Horn, Ari Leichtberg, Naty Leiser, Yossi Matias, and Ron Merom. Suggesting friends using the implicit social graph. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 233–242. ACM, 2010.

[9] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[10] Jierui Xie, Boleslaw K Szymanski, and Xiaoming Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 344–349. IEEE, 2011.