УДК 004.9

РЕКОМЕНДАЦИЯ ПОЛУЧАТЕЛЕЙ ЭЛЕКТРОННЫХ СООБЩЕНИЙ С ИСПОЛЬЗОВАНИЕМ РАЗЛИЧНЫХ ТИПОВ ЛОКАЛЬНЫХ ДАННЫХ СОЦИАЛЬНЫХ СЕТЕЙ

А.Гомзин, С.Ипатов, А.Коршунов

RECIPIENT SUGGESTION FOR ELECTRONIC MESSAGES USING VARIOUS TYPES OF LOCAL SOCIAL NETWORK DATA

A.Gomzin, S.Ipatov, A.Korshunov

Институт системного программирования РАН, gomzin@ispras.ru

Целью исследования является разработка алгоритма рекомендации потенциальных получателей сообщений в социальных сетях. Пользователи социальных сетей часто хотят отправить сообщение группе получателей. В статье предлагается алгоритм, предсказывающий для заданного начального набора получателей сообщения других возможных получателей среди ближайших контактов отправителя. Предложенный алгоритм использует различные типы локальных пользовательских данных: профиль, граф дружбы, публичные сообщения и социальные взаимодействия (отметки «мне нравится», комментарии, тэги). Некоторые из возможных получателей сообщения могут быть тесно связаны между собой и, следовательно, могут рассматриваться как единая группа. Алгоритм пытается выявить такие группы и рекомендовать их пользователю. Разработанное демонстрационное приложение для Facebook позволяет оценить работу алгоритма на реальных данных. Экспериментальная оценка с использованием этого приложения показала, что рекомендации алгоритма достаточно осмыслены для экспертов и могут оптимизировать работу пользователей с интерфейсом отправки сообщений в социальных сетях.

Ключевые слова: рекомендация получателей сообщений, системы рекомендаций, социальные сети, рекомендация пользователей, социальные сообщества, электронные программы мгновенного обмена сообщениями

The purpose of this study is to design an algorithm for recommending potential recipients of messages in social networks. Social network users often want to send a message to a group of recipients. In the paper we introduce an algorithm which predicts other possible recipients given an initial set of recipients. The suggested algorithm draws upon different types of local user data: the profile, friendship graph, posts, and social interactions (likes, comments, and tags). Some of recipients of a message could be closely related and, therefore, can be viewed as a uniform group. The algorithm attempts to reveal such groups and recommends them to a user. For the purpose of evaluating the algorithm we created a demo application for Facebook. Experimental evaluation using this application demonstrated that the algorithm is able to make suggestions meaningful to experts and to improve the messenger interface.

Keywords: recipient suggestion, recommender systems, social networks, user recommendation, social communities, electronic messengers

1. Introduction

In today's world people communicate by messages using the 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 from the same university.

The target data domain of our work is *Facebook** which appears to be the world's most popular social network. Our application analyses the data of a user and his/her friends: profiles, walls and friendship graphs. As a result, it suggests recipients for the message given a current list of recipients. This paper is an extension of the paper [1]. In addition to the previous work, this paper contains detailed analysis of impact of different kinds of data on suggestion results. Moreover, a group recommendation algorithm is introduced in this paper.

In our application** 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). The application also provides groups of recipients.
3. The user can:

— select the recommended user and add him/her to the seed set;

— select the recommended group and add it 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. 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 (rows of Interaction Table).
3. Suggesting for the most relevant users and groups given built interactions and seed set.

2.1. Terminology

User is an owner of a Facebook account.

Profile is a set of pairs (field name, field value). Profile contains information about a *user.*

Post is a public message of a *User*. *Post* contains a text, photo, 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 who 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 a *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.*

2.2. Data collection

We use the following Facebook data:

— Profiles of users.

— Local Friends Graph (ego-network): target user's friends list, mutual friends with target for each target's friend.

— Target user's and his/her friends posts with comments, likes, user tags, and text content.

The data is retrieved using Facebook Graph API*** given target user id.

2.3. 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 which contains users with the same fields' values;

— *target_target*: An interaction built from target's posts on the target's wall (likes, comments, and tags are analysed);

— *target_user*: An interaction built from other users' posts on the target's wall (likes, comments, and tags are analysed);

— *user_target*: An interaction built from target's posts on other users' walls (likes, comments, and tags are analysed);

— *user_user*: An interaction built from other users' posts on other users' walls (likes, comments, and tags are analysed);

— *other*: An interaction built from posts on walls that are not related to the target user (likes, comments, and tags are analysed);

* http://www.facebook.com

** http://rs.at.ispras.ru

*** https://developers.facebook.com/docs/graph-api/

— *content*: An interaction built from target's posts on the another users' walls with the same content;

— *community*: An interaction which includes members of the same community.

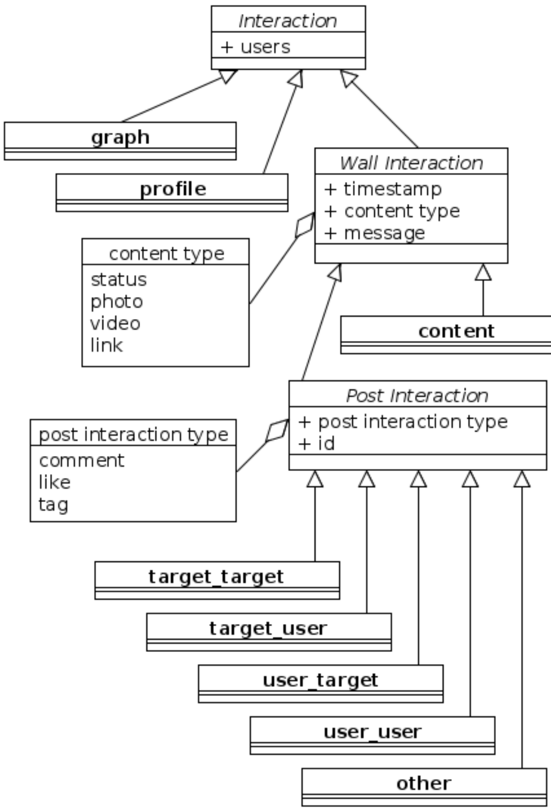


Fig.1. The hierarchy of interaction types

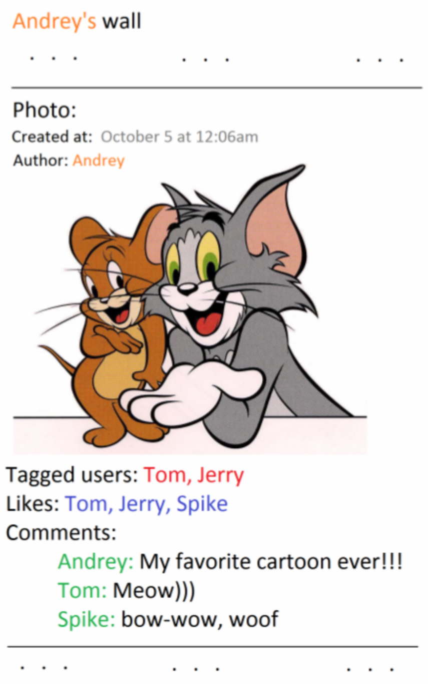
The hierarchy of interaction types is shown in Figure 1. *Wall Interaction*s are built using the post's comments, likes, tags or content. These interactions have timestamp and *content type* attributes. *Content type* has 4 possible values: status, link, photo, and video. *Post Interaction*s are built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

Target user's wall interactions

First, let us consider the target user's wall. The author of the post is important because he/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 a 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 a post contains comments/likes/tags *that include a target user*, a *target_user* interaction is created. Interaction contains all users that commented/liked/is tagged (on) the post and the post *author*. In case of a post contains comments/likes/tags that *don't include a target user*, some *other* interaction is created.



Andrey's wall

. . . .

Photo:
Created at: October 5 at 12:06am
Author: Andrey

Tagged users: Tom, Jerry
Likes: Tom, Jerry, Spike
Comments:
Andrey: My favorite cartoon ever!!!
Tom: Meow)))
Spike: bow-wow, woof

. . . .

Fig.2. Target's post on the target's wall example

Table 1

Interactions built from the post shown in Figure 2

Interation 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

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

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.

Let us consider a wall of any non-target user. Let's call him/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 a target user*, a *user_user* interaction is created. Interaction contains all users that commented/liked/is tagged (on) the post and the post *author*. In case of a post containing comments/likes/tags that *don't include a target user*, some *other* interaction is created.

Target posts text content interactions

Let us consider target user's posts on other 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

A target user contains a list of his/her friends. Other users contain lists of mutual friends with the target user. These connections between users form the target user's ego-network. The ego-network is used for finding communities of target's friends using Speaker-Listener Label Propagation Algorithm (SLPA) [2]. 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

A 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 their profiles. The interaction contains all such users.

We consider the following Facebook profile fields: *hometown, location, gender, relationship, religion, politics, work, work together with position, education, education together with graduating year*. For example, users that graduate from the same school are included into an interaction. Additionally, the users that graduate from the same school in the same year are included into another one interaction.

2.4. User suggestion

Recipient suggestion 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 the Interaction Table; the set of interactions: $\{i | users(i) \cap \text{Seed set} \neq \emptyset\}$. Here *users(i)* is a set of users of interaction *i*.

Weight*(i)* is:

— $w(\text{IntT}(i))$ - for graph and profile 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).

2.5. Group suggestion

This section describes an algorithm which recommends groups of users. It allows the target user to select desired group of recipients by “one click” rather that pick recipients one by one. The proposed algorithm is also based on Interaction Table.

The idea is to use clustering algorithm to divide all users into clusters. Then obtained clusters are recommended.

Let us consider a user as a vector. Let each interaction be a component of a user-vector. The value of the component of the user-vector is a weight of the corresponding interaction if the interaction contains given user, 0 otherwise. For example, assume that Table 1 is an Interaction Table. In this case we have the following vectors for users:

— Tom: $\{w(i_1), w(i_2), w(i_3)\}$;

— Jerry: $\{0, w(i_2), w(i_3)\}$;

— Spike: $\{w(i_1), w(i_2), 0\}$.

Here $w(i_j)$ is a weight of interaction i_j.

We use K-means algorithm [3] for clustering user-vectors. We calculate clusters for different *K*: from *2* to *N*, where *N* is $\frac{\text{number of target's friends}}{M}$. *M* stands for maximum group size. In our experiments we assume *M=5*. After that we remove groups larger than *M* and the ones which do not contain at least one user from the seed set. Then from obtained sets of clusters for different *K* parameter values we choose the set with the smallest number of recommended groups (but not 0). A small number of groups leads to increased intersection between members of groups and the seed set.

3. Experiments

This section describes accuracy evaluation experiments. 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 subsection 3.1 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.

3.1. Comparing with expert suggestions

Experts are needed for quality evaluation. For this metric evaluation we've developed a special tool that shows a random seed set of an expert's friends and asks to enter another *[0..5]* users from a 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 the recipients of some message, together with given seed set. The order of entered users is important: the first user is selected on assumption that current recipients is seed set, *k*-th user is selected on assumption that 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.

Quality of user suggestion

Let us assume that Seed set = *{su₁, su₂}*.

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 $u_1, u_2, \ldots, u_k$, an application calculates *k* $pos_i^e(m)$ values. $pos_1^e(m)$ 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 = $\{su_1, su_2, u_1, \ldots u_{i-1}\}$ and user u_i as described above.

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

$$P_K = \frac{|\{pos_i^j(m) \in POS \mid pos_i^j(m) \le K\}|}{|POS|} \qquad (3)$$

P_K takes values from [0..1].

Quality of group suggestion

The experts' data used for user suggestion evaluation is also used for groups suggestion evaluation. Let us assume that *Seed set* = $\{su_1, su_2\}$, and an expert *e* selects *Users* = $\{u_1, u_2, \ldots, u_k\}$. Now let us assume that the application recommends groups: $G_1 = \{u_{11}, u_{12}, \ldots\}$, $G_2 = \{u_{21}, u_{22}, \ldots\}$, ... $G_M = \{u_{M2}, u_{M2}, \ldots\}$. The quality metric is the mean value for all recommended groups of the Dice coefficient between recommended group *G* and $\text{Seed set} \cup \text{Users}$. The *Dice* coefficient is:

$$Dice(G, \text{Seed set} \cup \text{Users}) = \frac{2|G \cap (\text{Seed set} \cup \text{Users})|}{|G| + |\text{Seed set} \cup \text{Users}|} \qquad (4)$$

The mean *Dice* value is calculated for *all* experts and *all* expert selections in case of the algorithm suggests >0 groups.

3.2. Baseline

We use the method similar to the one proposed in [4] as a baseline. This method recommends recipients for e-mail, given an initial set of recipients, based on the 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 a user suggestion based on public data.

Online social network users often communicate with each other using public data: posts on walls, comments, likes, and 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 [4] don't consider e-mails that are not related to the target user, interactions of *other* type are not considered here.

The time decay function in [4] is the same as we use (2).

Hence, the algorithm presented in [4] 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.

3.3 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.

Quality of user recommendation

The result of comparing algorithm suggestions with experts' suggestions is shown in Figures 3-4. The histograms show the probability for a user selected by expert to fall into top-K users (3). Figure 3 contains histograms for described in this paper recipient suggestion algorithm (*wall interactions, all interactions*), and *baseline* method, described in section 3.2 and paper [4]. The histogram labeled as "*Wall interactions*" shows results for setup where only wall interactions are used (including interactions of *other* type). "*All interactions*" histogram corresponds to default setup where all interactions have non-zero weights. As we can see, the proposed algorithm works better than the one proposed in [4]. Moreover, we can see the impact of wall interaction between non-target users.

Figure 4 shows the impact of different types of interactions. "*Profile interactions only*" histogram shows P_K values for setup with all weights equal to 0 except for interactions of *profile* type. "*Graph interactions only*" histogram shows results for the setup that uses only interactions built from target user's ego-network. "*Wall interactions only*" histogram shows P_K values for setup with weights of *graph* and *profile* types equal to 0 (i.e. only *target_target, target_user, user_target, user_user, other, content* interaction types are considered here). "*All interactions*" histogram corresponds to default setup. As we can see, the usage of only profile interactions is not reasonable. The largest contribution comes from wall and graph interactions. Finally, the combination of all interactions gives the best results.

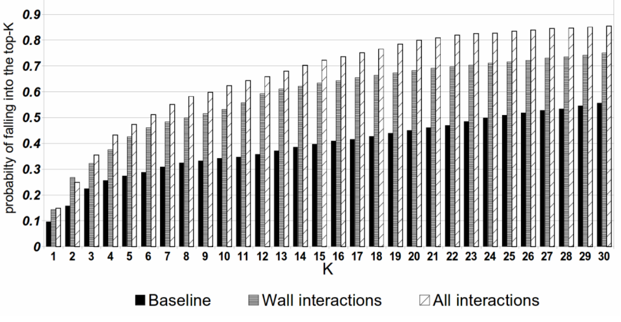


Fig.3. The probability for a user selected by expert to fall into top-K users. Recipient suggestion (using all interactions and using wall interactions only) and Baseline algorithms

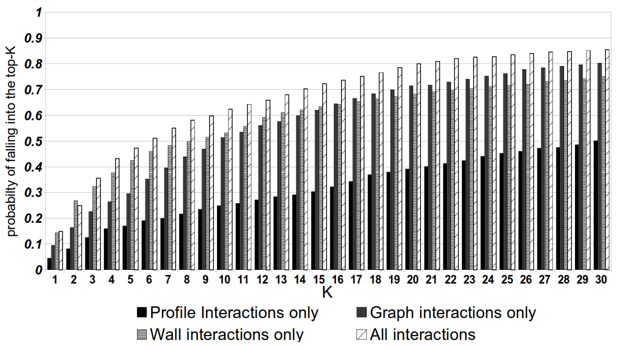


Fig.4. The probability for a user selected by expert to fall into top-K users. The impact of interactions of different types: *profile, graph, wall*

The quality of Recommendation of groups

The result of comparing algorithm's group suggestions with experts' suggestions is shown in Table 2 and Figure 5. Three setups are considered in these experiments. First, we run the algorithm using only profile interaction (all interaction weights except for *profile* interaction type are 0). The second setup has all zero interaction weights except for wall interactions (*target_target, target_user, user_target, user_user, other, content*). In the third setup only *graph* interactions are considered (other interactions have zero weight). Finally, we run the algorithm that uses all interactions.

The table shows an acceptable average dice coefficient between algorithm's and expert's data, but recall is very low: only in few cases algorithm recommends >0 groups. Moreover, Figure 5 shows that the dispersion is high for all setups. This means that in few cases algorithm recommends good groups, but the algorithm is to be enhanced.

Table 2

Groups recommendation evaluation results.
Share of non-empty recommendations and mean value of the Dice coefficient

Setup	Share of non-empty recommendations, %	Mean
Profile interactions	1.62	0.70
Wall interactions	17.89	0.66
Graph interactions	3.25	0.62
All interactions	5.50	0.68

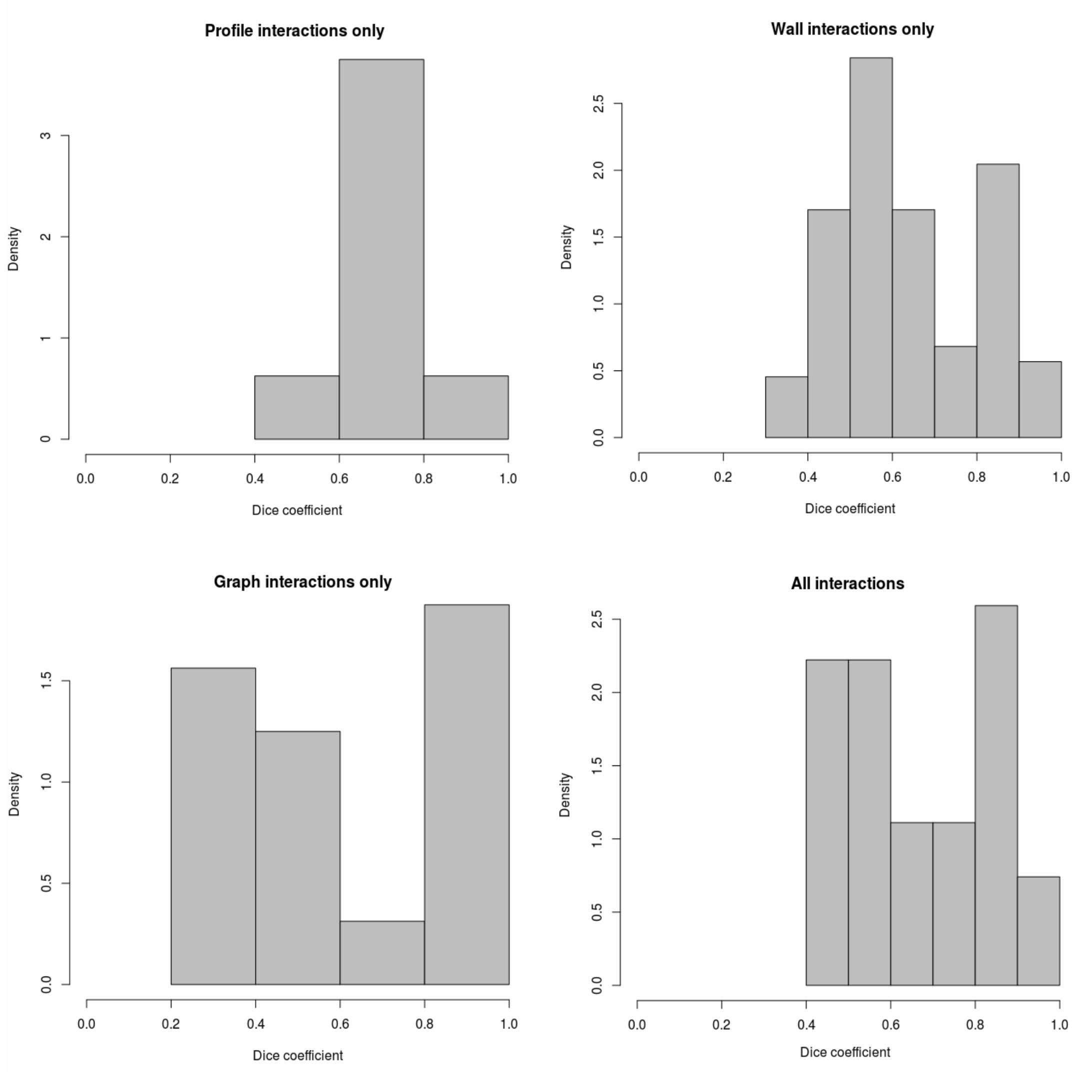


Fig.5. Groups recommendation evaluation results. Histograms of the Dice coefficient in all recommended groups

4. Results

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

The algorithm quality evaluation is based on users' feedback. Evaluation consists of two steps. First, the user chooses recipients without any suggestions. And then he/she uses a demo-application which suggests users according to the developed algorithm. The results of our Recipient Suggestion algorithm were compared with the baseline algorithm results. As shown in Figure 4, our algorithm is significantly better than the baseline one. 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. We have performed a detailed analysis of impact of different interaction sources. We obtained that graph communities and wall interactions contribute the most (Figure 4).

The algorithm that suggests groups of users is introduced. The evaluation shows that in many cases the set of recommended groups is empty, but if it is not empty, the recommended groups correlate with expert's selections. So, we've developed an algorithm with acceptable precision and low recall. The future direction is to increase the recall.

1. Gomzin A., Ipatov S., Korshunov A., Kim H. Recipient suggestion for electronic messages using local social network data. Presentation at SYRCoDIS 2014.
2. Xie J., Szymanski B. K., Liu X. Slpa. Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. Proc. of the IEEE 11th Int. Conf. on Data Mining Workshops (ICDMW 2011). Vancouver, 2011, pp. 344-349.
3. Hartigan J. A., Wong M. A. Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 1979, vol. 28, no. 1, pp. 100-108.
4. Roth M. et al. Suggesting friends using the implicit social graph. Proc. of the 16th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining. Washington, 2010, pp. 233-242.