1. INTRODUCTION

The growing number of opportunities and ways people can communicate and exchange information within an organization provide us with a previously unknown possibility to find, extract and store knowledge. The data extracted from email services, phone calls and other communication systems or common activities allow creating social networks (SNs) that contain information about humans interaction and collaboration. The capability to quickly find the correct answer for a specific question is crucial for every big company or organization. Unfortunately, most of the information needed to answer queries remain in the workers’ minds as a latent knowledge. Thus, the system called Social Latent Knowledge Explorator (SocLaKE) has been developed. SocLaKE makes use of the organization’s SN together with the information about people’s areas of expertise. It creates the list of recommended people who might know the answer for a given question or may know another person who knows the solution. The paper is organized as follows: first, the problem description and motivation is presented (Section 2); next, the related work is described (Section 3). In Section 4, the general concept of the SocLaKE system is introduced. Subsequently, the model of query propagation with recommendation (Section 5) along with the recommendation strategies (Section 6) are demonstrated. The experimental study and its results are depicted in Section 7. Lastly, discussion (Section 8) about some additional problems that need to be resolved before the deployment of SocLaKE system and the final conclusions (Section 9) are presented.

2. PROBLEM DESCRIPTION

In the daily life of most of the companies or organizations, employees ask hundreds of questions and face hundreds of problems. Some of them might be answered quickly owing to guides, forums or descriptions available on the intranet or internet web pages, instructions, etc. However, very often there are some questions, for which it is hard to find any answer. Hence, employees usually communicate with the company help desk, office supervisors, etc. waiting for an answer or assistance. This so-called “official way” often costs much time and energy. Hence, the
crucial problem within many medium-sized and large organizations is to simplify and speed-up the whole process of finding the right answers to inquiries that occur during the daily work. “Ask your friends for help” is one of the most common advices one can be given. Maybe your friends already know the answer and will provide you with it. If not, maybe they know someone else who faced the same problem, solved it and now they may contact you with this person. If your friends cannot help you, then they probably will ask their friends for assistance. In this way the question will probably be passed along until the satisfying response is found. Why do people enjoy doing this? The reason is that they are much more likely to help their acquaintances rather than strangers even though their expertise is out of the domain of the question and they do not know the proper solution directly. Hence, this approach is based on the rather obvious sociological phenomenon. That is, if people have a problem that they cannot solve by themselves, they look for help from their friends. Moreover, they usually trust an opinion stated by their friends more than that by strangers. However, the recent IT systems existing within companies do not contain any explicit information about user friendships and other social relationships, and hence, the company cannot support the searches for answers using these invisible relationships. On the other hand, it is possible to gather the data about communication within the company either from email systems, instant messaging systems, internal phone systems, etc. Moreover, people get into relationships during common activities, while solving tasks, developing projects, participating in meetings, etc. Information about these is usually also available within the organizational IT systems. The data about communication and common activities can be used to create a multi-layered SN of the company employees and their relationships [1]. Another interesting issue is the fact that the vital knowledge is very often passed only from mouth-to-mouth, and only exists in people’s minds and is not stored anywhere. For example, consider that a company has automatic air-conditioning that usually works properly but once it breaks down. Then, someone calls some mysterious Mr. John who has worked with this air-conditioning for 30 years and he tells to press a few buttons to restart the device. Air-conditioning starts working again and everyone is happy. Unfortunately, one sad day Mr. John retires. A month later, the air-conditioning breaks down again, and suddenly no one knows how to fix it because only Mr. John knew this secret button combination. A solution to such problems is a new system that will help to uncover this hidden knowledge and support users in searching people who probably can solve their problems. The system will provide the possibility to ask a question and subsequently, based on the gathered data about possible “experts” and the company’s SN, suggest some acquaintances who are either likely to provide the right explanation or know some other persons who may know the answer. In other words, the system supports the creation of a path of people. The first person in this path is a good friend of the user looking for the answer and the last one probably knows the solution for a given problem. Each person in the path is connected via a social relationship with a person before and after. After the answer is given, it is evaluated by the initializing person who has asked the question. Both the answer and the entire path are stored in the system. Owing to this approach, when a similar question is asked again, the system is able to either suggest an improved route to the previous experts or create another path to some other people who are supposed to be new experts for the domain of the question if, e.g., the primary experts are no longer members of the company’s SN.

3. RELATED WORK

Studies clearly show that a good market position of a company is highly indicated by the company’s knowledge resources [2]. Therefore, knowledge management systems (KMS) have become a research area of growing importance both for researchers and practitioners [3]. The number of different KMS is continuously growing. Nowadays, various KMS are used almost everywhere, from medicine [4] or student learning processes [5] to big companies from Silicon Valley like HP, Cisco or IBM [6]. Different types of knowledge for KMS have already been distinguished in the literature. Menzies proposed seven main types of knowledge: words, sentences, behavioral knowledge, problem-solving methods, quality knowledge, fix knowledge, and social knowledge [7], while other authors in [8, 9] suggested partition into explicit and tacit knowledge as the most important knowledge classification. All types of knowledge need a special management [7]; i.e., specialized data structures and models are needed for each of them. Unfortunately, the main source of knowledge for KMS is documents and this requires advanced analysis of natural language to retrieve knowledge of good quality [10].

Some improvements in classic information retrieval methods have been done recently on fields, such as ontology-based systems [11]. Jung presented a method for ontology mappings exchange and estimation and claimed “an improvement of about $42.5\%$ compared to the keyword-based query searching method” [12].

There are also some other problems than ontology and semantics, which are related to KMS development. Motivating the knowledge exchange is one of them. King and Marks presented two approaches to encourage people to exchange knowledge in the organization — the supervisory control and the social exchange [13]. Loebecke considered the paradox of knowledge sharing in the context of competition between companies. He also developed a model for this phenomenon [13]. The next important issue is the common dynamic nature
of knowledge; thus, information cannot be treated as static, always accurate and up to date and it needs additional management with respect to time factor [8, 6].

Help desks are one of the most obvious solutions to the problems described earlier [14]. However, in some cases help desk may be insufficient. For instance, when the problem is not technical or considers some uncommon aspect of company activity, help desk may not be able to provide the right answer.

An important source of knowledge that resolves most of the problems mentioned earlier is a human expert. Hence, an expert finding system (EFS) may be considered as a class of KMS. The ability to find suitable experts or relevant artifacts created by them is crucial for any modern organization [15]. Dozens of commercial EFSs are available on the market. AskMe ¹, Autonomy IDOL ², Endeca ³, Recommind ⁴, Triviumsoft’s SEE-K ⁵ - these are just a few examples of such systems. In general, finding an expert is a difficult task and it depends on knowledge artifacts (written statements, documents, reports, etc.) gathered by the organization. In most of the cases, some company-specific solutions are necessary [16]. Many challenges of EFSs have been identified so far, among which the most important are: lack of access to data about expert’s past activities, difficulty in automatic detection of expertise topics, no standards of qualification necessary to be an expert and measuring expertise level. Besides, experts are rare and access to them is usually limited and expensive [15].

The main issue of EFS is to identify experts and classify type and level of their expertise by analyzing experts communication, documents and activities. By doing these, EFS is able to rank experts and recommend top n to the user [15]. In most of the approaches, the expertise is calculated based on information retrieval methods. The system estimates the probability that the user is an author of the artifact by calculating an association function [16]. Then, by using a candidate-based or document-based correlation, the system is able to evaluate the user expertise for a given question [16]. A more sophisticated method utilizes email contents as well as information about communication profiles derived from the email exchange network [15]. A result of its comparison with a simple content-based approach facilitates to state that the usage of extra information about social communication of experts may provide a significant improvement in EFS accuracy and efficiency.

Furthermore a specialized subset of recommender systems (RS) may be treated as KMS. RS have been developed since the mid 1990’s. Traditional RS have been used in various types of e-commerce and news services [17, 18]. However, they can be applied in EFS (and KMS) as well. The main goal of the RS is to provide a list of objects matching user needs [19]. The most common examples of RS usage are: shopping [20], tasks to be done, content to watch [21, 22], interesting web sites, what to learn, interesting people to meet, personalized advertising [23], managing web sites [19], etc. Many data sets can be used as a source of RS. Depending on the RS type, almost any data type, which contains information about users and objects useful to build ranking lists, can be utilized, e.g., content and usage data, structure of the web site, user ratings, users profiles, data about communication between users and much more [19].

Three main approaches to building recommendations have been developed: collaborative RS, content-based systems, and hybrids of these two [24]. Collaborative recommendation is based on tastes of people similar to a given user, i.e. the nearest neighborhood and it makes use of these likes to build a ranking [25]. Content-base RS ranks objects based on the similarity of their profiles (content) and those of the objects already evaluated by or assigned to a given user. The different kinds of division or ranking types are: statistic methods, profile matching, and data mining. Statistic techniques are the most basic, e.g., the most frequent purchase or the best ranked. Profile matching, also called demographic filtering, uses the same information as collaborative ranking method, but it focuses on associations between the user profile and the profile of the object. Data mining techniques are the most sophisticated—association rules, clustering, classification methods and many others can be used to recommend objects without providing any ratings [19].

One of the most important types of RS is systems that recommend people—social matching systems (SMS). Based on people’s common interests, activities and similar demographic profiles a SN is created. For example, using social connections between humans in SN, the RS suggests to a user uₓ some other SN members with a similar profile, whom user uₓ might be interested [1]. This idea is also useful in an EFS, in which user uₓ seeks for an expert with expertise matching user uₓ’s question. Therefore, an EFS recommends relevant experts instead of documents as in regular KMS. Over the last decade, many commercial solutions and research models have been proposed to solve the problem of social matching [26]. Overall, the main objective of SMS is to match together people with similar interests [27]. However, the concept of SMS has been developed extensively and recently, we have SMS matching people, who visit same websites [28] same locations [29]; SMS locating experts related to a particular subject [30, 27]. The authors proposed the social matching model, which utilizes users’ skills, social relationships and roles that a user can play, to support creation and management of work teams.

¹http://www.askme.com
²http://www.autonomy.com/content/Products/products-idol-server/index.en.html
⁴http://www.recommind.com/
⁵http://www.triviumsoft.com/
As mentioned earlier, SMS allows finding similar or potentially interesting people to establish some new social relationships. These new relationships may be incorporated into a SN, which can be used to find another person who may attract a given user [30].

SN have expanded over the last few decades from sociology to computer science. Recently, a large number of publications, commercial implementations and theoretical models have come out [31, 32, 33, 34, 35]. The general definition of SN is as follows: a finite set of individuals, who are the nodes of SN, together with the relations between them, which are represented by edges of the network. A SN commonly represents the mutual communication and activity occurring between users as well as their direction, intensity, and even specific profile. Social networks analysis (SNA) is a domain of science, which allows extracting some useful measures for an individual (social entity), a community (group) or the entire network in the context of social connections of people rather than with respect to their profiles [36]. SNA appears to be quite important and applicable, especially now when information about users’ communication and activities is much easier to be gathered, and is more accurate and up-to-date than the data about personal profiles provided by themselves [37]. An example of the usage of SN within the expert finding problem is Constellation – an application, that supports expert finding by SN visualization [38].

However, to exploit all the information from the SN structure a little more than social matching or simple visualization is required. To achieve this, the social query model (SQM) has been proposed, which uses a query propagation concept, which is, in a sense, similar to data propagation [39, 40]. SQM is a model for decentralized answer search by routing queries through a SN [41, 42]. Dispersed search algorithm treats SN as a set of potential experts to whom anyone can ask a question. The algorithm routes a question from the asker to an expert automatically. In SQM, the effectiveness of the routing policy is measured by the probability of retrieving the correct answer. Recently, a few new models for the decentralized answer search have been developed. An example might be the Aardvark system – the application improves the query propagation process by recommending experts [43]. However, in contrast to other models for decentralized answer search, SQM at every step takes into consideration factors, such as: an expertise level, correctness, response rate and general policy [41, 42] (see Section 5.1 for details).

4. GENERAL CONCEPT OF SOCLAKE

Query propagation in the SN can be defined as spreading a query from an asker down the path of friends and colleagues. This process is often used by people to find an answer to the question when they do not know a competent expert directly. For example, let us imagine a big enterprise with thousands of employees. One of them, Alice, has a problem with a software system installed on her computer. She cannot solve the problem herself and cannot find any help in an outdated documentation. In this situation, she talks to her friend, Bob, working in the same department. Bob also does not know the answer but his friend, Cecile, had a similar problem earlier. Bob sends her an email asking Alice’s question. In this way Alice’s query is propagated to others in the company and finally reaches an expert (Cecile) in another department who knows the solution to the problem. She answers the question and the information goes back to Alice.

Even in this simple example some basic features of query propagation in the SN can be noticed. First, the query is propagated by different means of communication. Phone call, email, text messages or face-to-face talk can all be used to propagate a query. Second, the way the query is spread in the SN depends only on local knowledge of the nodes. There is no central place from which a query is routed. People passing the query do not consult anyone asking who they should send their request to. Third, there are two aspects by which people decide whom to ask questions: substantial and social. Both of them are equally important. People often prefer to ask a question to someone with whom they have good relations rather than the expert whom they do not know well or not at all [44]. This is a social phenomenon of many human activities.

Based on the idea of query propagation over the SN, the system called Social Latent Knowledge Explorer (SocLaKE) was developed. The basic concept of SocLaKE is presented in Figure 1. The SocLaKE system utilizes various data about communication within the organization to extract the organizational SN. The main sources of such data are email systems, internal phone systems, and information about the organization structure, i.e. who works with whom or who shares the same room. However, depending on the organization, each IT legacy system and data related to interpersonal communication can be used to extract the more accurate network of human relationships within the organization.

After creating the initial SN SocLaKE needs to gather information about areas of expertise of each member in the SN. All documents starting from official ones through notes, descriptions, forum discussions and ending with the email contents can be used for this purpose. Additionally, once again the organization’s structure can be utilized, i.e. the data about positions occupied and workers’ responsibilities. Moreover, users themselves can provide information about their areas of expertise.

Having the SN created and areas of expertise defined, the SocLaKE system has to compute a set of coefficients that are needed to prepare recommendations. Subsequently, SocLaKE is ready to
use and supports members of the organization to solve their problems. For more details about appropriate coefficients and how to calculate them see Sections 5.1 - 5.3. SN, areas of expertise, and coefficients should be updated every day, week, month, etc. depending on the SocLaKE settings. For the update, SocLaKE utilizes the same sources of data, which were used initially; of course, the system can also be provided with some new sources of information.

The SocLaKE system may be embedded into regular organization systems used for everyday work, such as email agents (Outlook, Thunderbird), web browsers (IE, Firefox, Chrome, Opera), instant messengers (ICQ, MSN), and VoIP systems (Skype) so that users do not need to turn on any special application to ask their questions. They can just type their request and send as an email or text in the instant messenger. The SocLaKE system automatically analyzes the query and recognizes its domain. Obviously, if someone is concerned about privacy of the message, then the system can be blocked from the message. Based on the domain discovered and user relationships maintained in the SN, SocLaKE calculates the best route to the expert step-by-step. Depending on the asker settings, the proposed route might be the fastest, the most reliable, or something in-between. Next, the SocLaKE system generates a recommendation using its built-in strategy, see Section 6 for some examples. The recommendation consists of a list of friends from whom, according to the SocLaKE system, the user can get support and finally the answer to a given query. If the user decides to use one or more recommendations from the list, SocLaKE sends the appropriate request to the user’s friends selected from the list. If some of them know the answer, then they send them to the asker and if he/she is satisfied with the answer, the query propagation is stopped. If none of the friends knows the answer, then they can forward the query using the same recommendation mechanism as the asker. The question is routed through the network until the answer is found or until the predefined number of network members is provided with the query.

If the initial user agrees, the query and its answer are stored in the database and used when someone asks the same or similar question in the future. Thus, the SocLaKE system allows finding, accessing and gathering the latent knowledge in the organization. This kind of knowledge is usually inaccessible by any other means and can be efficiently revealed only by making use of local social relationships existing between the members of the large community.
5. THE MODEL OF QUERY PROPAGATION WITH RECOMMANDATION

5.1. Query Propagation Model

Query propagation can be described using the SQM introduced in [41, 42]. In this model, nodes in the SN are described by a set of probabilities denoting how people behave when obtaining a question related to a certain domain. Using SQM one can estimate the probability of finding an answer to the query propagated over the SN. Originally, the authors used their model to discuss an optimal policy of the nodes (humans) in the network, i.e., how nodes should route the query to find a proper answer effectively. SQM has been successfully applied in military knowledge sharing support systems [45].

The query propagation process described by SQM starts when, e.g., user $u_i$ has a question. User $u_i$ could be an expert in the domain of the question and may be willing to answer it on his/her own. However, if it happens that $u_i$ is not competent in this domain, then he/she sends a query to his/her friends. If a given user $u_j$ receives a query from person $u_i$, then $u_j$ can either ignore it or react somehow. If $u_j$ decides to react, $u_j$ can either answer directly to the initiator or again pass the query further to $u_j$’s friends. SocLaKE supports users at this stage - it recommends the best addressee of their message. The entire process is illustrated in Figure 2.

Overall, SQM is a stochastic model. Each node $u_i$ of the SN has a set of probabilities assigned. They describe the behavior of a particular node $u_i$ either generally or with respect to some other nodes $u_j$. Hence, we have: policy $\pi_{ij}$, responsiveness $r_{ij}$, expertise $e_{ij}^{(q)}$, and correctness $w_{ij}$. All of them belong to the range $[0;1]$.

The policy ($\pi_{ij}$) is a probability that user $u_i$ will send (pass) a query to person $u_j$. If $u_i$ is a good colleague of $u_j$, the value of $\pi_{ij}$ will be probably high. However, if users $u_i$ and $u_j$ are strangers to each other, the probability will be nearly equal to 0. Factor $\pi$ possesses the following properties:

\begin{align*}
\pi_{ii} &= 0, \\
\sum_{j=1}^{n} \pi_{ij} &= 1,
\end{align*}

where $n$ denotes the total number of nodes in the SN.

A user cannot propagate any query to himself/herself, because of Equation (1). The sum of the probabilities of passing the query to all other nodes in the network has to be equal to 1, as shown in Equation (2). The factor $\pi_{ij}$ is independent of the query, indicating that the value of $\pi_{ij}$ is the same for every query $q$.

By analyzing the communication of users, we can estimate the value of $\pi_{ij}$ for all pairs $i, j$. For example, if 50% of all emails sent by $u_i$ are directed to $u_j$, then $\pi_{ij} = 0.5$.

The policy matrix containing values of $\pi_{ij}$ for all $i$ and $j$, i.e., for all possible pairs of users, is denoted by $\Pi$ :

\[
\Pi = \begin{bmatrix}
\pi_{11} & \cdots & \pi_{1n} \\
\vdots & \ddots & \vdots \\
\pi_{n1} & \cdots & \pi_{nn}
\end{bmatrix}. \tag{3}
\]

The responsiveness ($r_{ij}$) is a probability that user $u_j$ will react to a query received from user $u_i$. According to general human social phenomena, we can assume that if user $u_j$ is a good friend of $u_i$ and $u_j$ has relatively much free time, the probability $r_{ij}$ will have high value. On the other hand, if user $u_j$ is busy or is not closely related to user $u_i$, the value of $r_{ij}$ will be low.

Similarly to $\pi_{ij}$, values of $r_{ij}$ can be calculated using communication data. For instance, if 30% of emails from user $u_i$ to $u_j$ are answered by $u_j$, then $r_{ij} = 30%$.

The responsiveness matrix of all values of $r_{ij}$ is denoted by $R$, as follows:

\[
R = \begin{bmatrix}
\alpha_{11} & \cdots & \alpha_{1n} \\
\vdots & \ddots & \vdots \\
\alpha_{n1} & \cdots & \alpha_{nn}
\end{bmatrix}. \tag{4}
\]

The Computer Journal, Vol. ??, No. ??, ????
The expertise coefficient \( \varepsilon^{(q)}_i \) is a probability that user \( u_i \) decides to answer a given query \( q \). The value of \( \varepsilon^{(q)}_i \) strongly depends on the domain of query \( q \). If \( u_i \) is a competent expert in the domain of a given query, probability \( \varepsilon^{(q)}_i \) will be high. It must be noted that the other coefficients described earlier, namely, \( \pi_{ij} \) and \( r_{ij} \) are query independent whereas \( \varepsilon^{(q)}_i \) is not.

The value of \( \varepsilon^{(q)}_i \) can be estimated by means of methods being used in typical EFS\[16\]. In general, the SocLaKE system itself evaluates the expertise \( \varepsilon^{(q)}_i \) of user \( u_i \) for query \( q \) based on the available contents and achievements, e.g., documents authored by \( u_i \) and the historical responses of \( u_i \) to the previous requests.

The expertise vector \( E^{(q)} \) aggregates values of \( \varepsilon^{(q)}_i \) for a given query \( q \) and for all users:

\[
E^{(q)} = \begin{bmatrix} \varepsilon^{(q)}_1 & \cdots & \varepsilon^{(q)}_n \end{bmatrix}.
\] (5)

The correctness \( w_i \) is a probability that an answer of user \( u_i \) satisfies the asker. It is high, if user \( u_i \) usually provides high-quality, understandable and satisfying answers. One might imagine a pseudo-expert having high expertise probability \( \varepsilon^{(q)}_i \), but providing rather poor-quality answers. In such case, coefficient \( w_i \) will be low. The value of \( w_i \) can be computed using user feedback, i.e., an initiator of the query should rate the answers obtained. Based on the historical ratings of user \( u_i \)'s answers, the recent value of \( w_i \) is calculated.

The correctness vector \( W \) contains values of \( w_i \) for all users \( u_i \) in the SN:

\[
W = \begin{bmatrix} w_1 & \cdots & w_n \end{bmatrix}.
\] (6)

The probability \( P^{(q)}_i \) of finding the answer to a given query \( q \) asked by user \( u_i \) can be calculated using SQM concept, in the following way:

\[
P^{(q)}_i = \varepsilon^{(q)}_i w_i + (1 - \varepsilon^{(q)}_i) \sum_{j=1, j \neq i}^{n} \pi_{ij} r_{ij} P^{(q)}_j. \tag{7}
\]

The probability of finding the answer by user \( u_i \) for query \( q \) is equal to the probability that \( u_i \) is an expert in the query \( q \) domain and knows a correct answer, plus a sum of probabilities that all other persons \( u_j \) asked by user \( u_i \) react and are able to find a right answer. It must be noted that the computing based on Equation (7) is recursive. Hence, to calculate a probability of obtaining the answer for user \( u_i \), all other probabilities have to be estimated.

Let us denote the probability vector as \( P^{(q)} \). It aggregates probabilities \( P^{(q)}_i \) for all users \( u_i \) in the network with respect to a given query \( q \):

\[
P^{(q)} = \begin{bmatrix} P^{(q)}_1 & \cdots & P^{(q)}_n \end{bmatrix}.
\] (8)

Let \( C^{(q)} = E^{(q)} \odot W \), \( A = \Pi \odot R \) and \( D^{(q)} = I - \text{diag}(E^{(q)}) \). Operator \( \odot \) denotes Hadamard product.

5.2. Recommendation in Query Propagation

In this paper, the recommendation will be defined as a finite subset of options available to the particular user. While making decisions, humans are capable of effectively evaluating only a few available options. When there are too many items to choose from, our brain cannot compare them efficiently and individual options become indistinguishable. Usage of recommendations prevents from information overload and enables users to undertake their decisions in a more efficient way\[46\]. This is because recommendations are, in fact, a form of pre-evaluation of the larger set of available options. Therefore, recommendations deliver a tool to distinguish between options: there is only a small group of recommended items whereas the others remain hidden. If users trust the recommendation system, then they will also be able to make use of recommendation lists quite effectively.

In terms of query propagation, recommendations can be applied at the stage of choosing the next person to pass the query to. In larger organizations, there are usually hundreds and even thousands of people who might be potentially contacted and asked, and hence, choosing among them is almost impossible. Moreover, even if the choice is limited to the closest friends, there are still many options, that are hardly distinguishable. As mentioned earlier, in such case, recommendations can significantly improve the decision process. Additionally, people are not aware of all skills and competences of all their co-workers. For all these reasons, the usage of recommendations can incorporate a new knowledge into decision process.

Recommendations obviously influence user behavior. If recommendations are of a good quality, the suggested options should be chosen significantly more frequently than those not recommended. In\[47\], the authors have proposed the assessment model based on the maximum entropy method. Using this model, one can estimate the influence of recommendations on customers’ choices. The model used by SocLaKE results from the application of a similar concept to the query propagation model.

In the SocLaKE system, a recommendation is represented by \( S^{(q)}_{ij} \):

\[
S^{(q)}_{ij} = \begin{cases} 1, & \text{if user } u_j \text{ is recommended to user } u_i, \\ 0, & \text{otherwise.} \end{cases}
\]

Let us denote the recommendation matrix by \( S^{(q)} \):

\[
\text{The Computer Journal, Vol. ??, No. ??, ?? ??}
\[ S(\Psi) = \begin{bmatrix} s_{11}^{(\Psi)} & \cdots & s_{1n}^{(\Psi)} \\ \vdots & \ddots & \vdots \\ s_{n1}^{(\Psi)} & \cdots & s_{nn}^{(\Psi)} \end{bmatrix}. \tag{10} \]

The values in \( S(\Psi) \) directly depend on a recommendation strategy (\( \Psi \)), see Section 6 for details and examples. The SocLaKE system uses its recommendation strategy (\( \Psi \)) to generate the list of recommended people for every user based on the SN coefficients (\( \Pi, R, W, E^{[q]} \), and \( \Phi \) - defined subsequently).

In our model, a recommendation changes user’s policy \( \pi_{ij} \). The final policy \( \pi_{ij} \) for user \( u_i \), influenced and modified by the recommendation, expresses the \( u_i \)'s tendency to pass a query to another user \( u_j \). The value of \( \pi_{ij} \) is expressed as follows:

\[ \pi_{ij}' = \frac{\pi_{ij} \exp(\phi_{ij}^{(\Psi)})}{\sum_{k=1}^{n} \pi_{ik} \exp(\phi_{ik}^{(\Psi)})}, \tag{11} \]

where \( \phi_i \in \mathbb{R} \) is the susceptibility coefficient of user \( u_i \).

Formula (11) was created based on maximum entropy method applied on policy \( \pi_{ij} \) with respect to susceptibility coefficient \( \phi_i \) [47].

Susceptibility \( \phi_i \) denotes the extent to which, in general, recommendations influence user \( u_i \). The greater the value of \( \phi_i \), the more likely is the user \( u_i \) to follow recommendations and pass the query to the suggested people. Neither values of \( \phi_i \) nor \( \pi_{ij}' \) depend on the query.

The susceptibility vector \( \Phi \) collects susceptibilities of all users in the SN:

\[ \Phi = [ \phi_1 \cdots \phi_n ]. \tag{12} \]

### 5.3. Coefficient Estimation

Sections 5.1 and 5.2 have introduced a set of coefficients needed to calculate the probability of finding the right answer to the question stated in a particular community (SN) and supported by SocLaKE via recommendations. To use the query propagation with the recommendation model in a real-world system, the following coefficients have to be estimated.

The policy \( \Pi \) coefficients can be calculated using the frequency method. To estimate \( \pi_{ij} \) the communication of user \( u_i \) is observed. If user \( u_i \) sends \( n_{ij}^{(m)} \) messages in total, out of which \( n_{ij}^{(m)} \) is sent to user \( u_j \), then \( \pi_{ij} = \frac{n_{ij}^{(m)}}{n_{ij}^{(m)}} \). If policy coefficients were estimated using this approach, then the probability that user \( u_i \) sends a message to user \( u_k \) for the first time is \( \pi_{ik} = 0 \). However, the probability of asking a question to a stranger should be positive rather than zero. One way of dealing with this problem assumes that there is a fixed small probability of asking a stranger. In this approach, the probability of user \( u_i \) asking his/her known friend \( u_j \) is \( \pi_{ij} = \frac{n_{ij}^{(m)}}{n_{ij}^{(m)}} \left(1 - \pi_{ik}^{(a)} \right) \), where \( \pi_{ik}^{(a)} \) is the total probability of user \( u_i \) asking any of the strangers ("aliens"). Therefore, the probability of user \( u_i \) asking an individual stranger \( u_k \) equals \( \pi_{ik} = \frac{n_{ik}^{(a)}}{n_{ik}^{(a)}} \), where \( n_{ik}^{(a)} \) is the number of strangers for \( u_i \), i.e. people belonging to the SN whom user \( u_i \) does not know. The probability \( \pi_{ij}^{(a)} \) of asking any stranger, i.e. tendency to communicate with new people rather than the known ones depends on the number of messages sent to old friends, when compared with those sent to new ones. Usually the higher number of friends that \( u_i \) has, the lesser is the probability that \( u_i \) will contact someone new. As a result, \( \pi_{ij}^{(a)} \) can be calculated in the following way:

\[ \pi_{ij}^{(a)} = \begin{cases} 1, & \text{if } n_{ij}^{(m)} = 0, \text{ i.e. } u_i \text{ has no friends} \\ \frac{n_{ij}^{(m)}}{n_{ij}^{(m)} + 1}, & \text{if } n_{ij}^{(m)} > 0, \end{cases} \]

where \( n_{ij}^{(1)} \) is the number of first messages sent by \( u_i \) to other users, i.e. \( u_i \)'s messages that were the first in the mutual communication.

Fraction \( \frac{n_{ij}^{(r)}}{n_{ij}^{(r)}} \) is applied when user \( u_i \) has at least one acquaintance. If \( u_i \) sent exactly one message to someone, then \( n_{ij}^{(1)} = n_{ij}^{(m)} = 1 \) and \( \pi_{ij}^{(a)} = \frac{1}{2} \). In general, if more messages are sent to the old friends rather than to the new ones, then \( \pi_{ij}^{(a)} \) becomes smaller.

The responsiveness \( R \) coefficients can also be estimated using the frequency method applied to the communication data. The value of \( r_{ij} \) can be calculated as \( r_{ij} = \frac{n_{ij}^{(r)}}{n_{ij}^{(r)}} \), where \( n_{ij}^{(r)} \) denotes the number of messages sent from \( u_i \) to \( u_j \) which were responded by \( u_j \) (for more information see Section 5.2). Similar to policy, the responsiveness \( r_{ij} \) also has to be calculated differently in case, when user \( u_i \) has never sent a message to user \( u_j \), because in such case \( n_{ij}^{(m)} = 0 \).

The responsiveness, by taking into account reacting to such queries can be calculated as:

\[ r_{ij} = \begin{cases} \frac{n_{ij}^{(r)}}{n_{ij}^{(r)}}, & \text{if } n_{ij}^{(m)} > 0 \\ \frac{n_{ij}^{(r)}}{n_{ij}^{(r)}}, & \text{if } n_{ij}^{(m)} = 0 \land n_{ij}^{(a)} > 0 \\ \frac{1}{2}, & \text{if } n_{ij}^{(a)} = 0, \text{ i.e. no message to } u_j, \end{cases} \]

where

- \( n_{ij}^{(a)} \) is the number of different users \( u_k \) who have sent any message to user \( u_j \), i.e. the number of the first messages from \( u_k \) to \( u_j \);
- \( n_{ij}^{(ar)} \) denotes the number of all first messages sent from any user \( u_k \) to \( u_j \), which have been responded by \( u_j \).
The expertise matrix $E(q)$ (see Equation (5)) can be calculated using regular methods known from EFS [16, 48, 49]. The probability $e_{ij}(q)$ that user $u_i$ decides to answer query $q$ can be approximated by the probability that $u_i$ is an expert in the domain of $q$. The relevance between user $u_i$ and query $q$ may be estimated by matching query $q$ with textual documents associated (authored) with user $u_i$. The content of previous communication of user $u_i$ can also be considered as a kind of document; therefore, $E(q)$ might be calculated using different communication data that includes former answers of user $u_i$ provided to the SocLaKE system as well.

The correctness $W$, Equation (6), can be obtained by collecting users’ ratings of the received answers. Users can give their feedback about the quality and usefulness of the information they have obtained from the SocLaKE system and in consequence from a particular user expert. This feedback can later be processed using collaborative filtering methods [50] to predict the correctness value $w_i$ for a given expert $u_i$. In other words, correctness expresses how relevant and helpful are users’ answers.

The susceptibility matrix $Φ$ (Equation (12)) can be estimated by analyzing the recommendation influence on users’ choices using the maxent method [47] or by processing data about SocLaKE usage. The susceptibility $ϕ_i$ of user $u_i$ can be estimated as a percentage of decisions they made in accordance with a recommendation in relation to the total number of recommendations.

6. RECOMMENDATION STRATEGIES

SocLaKE is intended to operate on very large SN. It also has to work online to provide recommendations to users in real time. Therefore, RS should be possibly optimized. Its crucial part is the recommendation strategy $Ψ$. This strategy is responsible for generating a list of people suggested to each user based on the appropriate matrices ($Π$, $Ψ$, $R$, $W$, $E(q)$, and $Φ$). During the experiments, a set of simple recommendation strategies $Ψ$ were examined, in particular:

1. No recommendation (none) - system does not provide any recommendations; used to compare with other methods.
2. Random recommendation (m) - for each user, the system recommends $m$ other users randomly.
3. Expert recommendation - a user with highest expertise is recommended, similar to EFS. This strategy is independent of social relationships between people.
4. Best relation (m) - for each user $u_i$, $m$ other users that are most likely to be chosen according to social relationships are recommended, i.e. with the highest $r_{ij}$ value.
5. Best answering (m) - for each user, $m$ other users most likely to respond to the query are recommended. Users $u_j$ are ordered according to the value of their responsiveness $r_{ij}$ towards the current user $u_i$.
6. Best answering (t) - for each user $u_i$, all other users $u_j$ for whom the probability of response $r_{ij}$ is greater than a given fixed threshold $t$ are recommended.
7. Best answering (st) - for each user $u_i$, a number of users having the highest response probability $r_{ij}$ are recommended; the sum of responsibility of the recommended users is less than the sum threshold $st$.
8. SQM - this strategy reflects the SQM optimal policy of recommendation [41, 42].
9. Max - the most efficient recommendation found using the genetic algorithm. Genetic algorithm was used because even for the small SN, the number of possible recommendations to consider is very high, e.g., for the 20-node SN, this number is $2^m(n-1) = 2^{380} \approx 2.5E14$.

The above-mentioned list of possible recommendation strategies does not exhaust all possibilities. The others can utilize some additional information, especially data gathered by SocLaKE itself, e.g., feedback from the users, paths of queries asked before, content of historical messages as well as supplementary user or system parameters (expiry date, maximum number of users in the community, who may be given a query, etc.). Moreover, the final strategy may be an adaptive ensemble of several component strategies. Some relevant discussion may be found in Section 8.

7. METHOD VERIFICATION, SN SIMULATIONS

7.1. An Example of a Simple SN

An example SN has been created for the case study and preliminary experiments and has been published.
online\(^6\) (see Figures 3 and 4). All values that describe network, users and edges were randomly assigned with distribution from real networks, i.e. Wroclaw University of Technology e-mail communication network and DBLP co-authorship network.

SocLaKE is intended to operate on very large SN, however, some methods used in the comparative experiments are very expensive. Hence, we need a sample network to be small due to the computational complexity of these non SocLaKE algorithms. The example SN contains 20 nodes representing company employees (users). Each user \(u_i\) has been assigned three values: expertise \(e_i^{(q)}\) for a given query \(q\), the correctness \(w_i\) of user \(u_i\)'s answers and susceptibility \(\phi_i\) (see Section 5). Edges reflect relationships between users and are derived from mutual communication (email, phone calls) or common activities, e.g., on internet forums or co-authorship of documents. In this case edges have the same distribution as edges in Wroclaw University of Technology e-mail communication network, and hence, the edge between two users means that they have communicated each other via email. Every edge linking from user \(u_i\) to \(u_j\) possesses two other attributes: policy \((\pi_{ij})\) and responsiveness \((r_{ij})\), see Section 5.1. Overall, edges in Figure 3 represent only user contacts or collaboration extracted from the data. However, there is always a non-zero probability of the contact between nodes not connected by any edge derived from the data. These non-zero values have been assigned due to simulation purpose. This indicates that the final SN is, in fact, a complete graph but most of the edges have only small values assigned to their attributes (see Section 5.3).

For example, user Tom was an initiator of the query flow process in the case study (see Section 7.2). However, in simulations all users were treated equally (see Section 7.3).

The average node degree in the network is 2.8 while median is 3. No user has more than five relations. The intention is to model a SN of experts from different domains loosely connected, such as architects, computer administrators or university scientists rather than a close family or a co-worker group from the same company department. There would be no point for the computer system to aid an expert search process in the network where everyone knows each other and everything about each other. Hence, the network is relatively sparse.

### 7.2. Query Flow Case Study

Figure 4 illustrates a flow for the example question \(q\) within the community shown in Figure 3. Let us assume that user Tom has a problem and he asks the SocLaKE system for help. There are six steps in the entire process.

---

**FIGURE 4.** The process of answering to the example query \(q\) in the small social network shown in Figure 3

**STEP 1:** User Tom types his question into the SocLaKE system.

**STEP 2:** Tom is given a list of his acquaintances, who may help to solve the problem. He chooses only Peter and sends him query \(q\).

**STEP 3:** Peter, however, does not know the answer, and hence, he also requests the SocLaKE system for assistance. Peter receives another list of his acquaintances, who may be helpful. He decides to choose Kate and Bob and passes them Tom’s message with query \(q\).

**STEP 4:** Unfortunately, Kate is out of the town, and hence, she cannot answer. Luckily, Bob reads the message and makes a decision to forward query \(q\) to another person - Judy.

**STEP 5:** As Judy also does not know the explanation, the system once again assists the user and provides to Judy a list of potential experts, who may know the solution for Tom’s problem. As a result, Judy sends a message to John and Lisa.

**STEP 6:** John, in turn, does not really like Judy, so he decides not to answer the query at all. Fortunately, Lisa is Judy’s best friend and also an expert in the query \(q\) domain. She provides the right answer and SocLaKE forwards it to the initiator Tom. User Tom optionally ranks the answer and the process is terminated.

### 7.3. Setup of Simulation Experiments

The experiments were divided into two parts. The first part, described in Section 7.3.1, was conducted using a small example SN having 20 nodes (see Figure 3). The results obtained from this experiment were used next as a base to the qualitative analysis of the query propagation phenomenon and verification of the experimentation methods. In the second part of experiments, large SN, having 1000 nodes, were used (see Section 7.3.2). In this kind of experiments, the proposed method was examined in many different environments to check its effectiveness.

---

In general, two distinct distributions have been used in simulation experimental studies. The beta distribution has been chosen because it is commonly used to describe the distribution of an unknown probability variable \([51]\). Also, it is a binomial distribution; thus, it appears to be natural for normalized measures. Finally, the beta distribution allows affecting its deviation by changing values of its parameters, and it better reflects phenomenon such as low number of experts in the SN. In case of the nodes communication measures, power law distribution was also used. Power law distribution is most commonly used for SN \([52, 53]\); hence, it is natural to conduct experiments based on it.

The experiments were conducted using python programming language with the numerical calculations package (NumPy) installed.

7.3.1 Small Network

The experiments were performed on the example SN presented in Section 7.1 (Figure 3) and consisted of several steps. The appropriate coefficients were stored in suitable matrices (see Section 5). The information about social connections between nodes was saved directly in the policy matrix \(P\) (Equation (3)) and in the responsiveness matrix \(R\) (Equation (4)). Both \(P\) and \(R\) were filled up randomly but with respect to the structure depicted in Figure 3, i.e. for invisible edges \(\pi_{ij} < 0.01\) and \(r_{ij} = 0.4\), whereas for edges visible in Figure 3, \(\pi_{ij}\) was significantly > 0.01, but \(r_{ij}\) could be either > 0.4 (for people communicating a lot) or < 0.4 (for users rather avoiding each other). The expertise matrix \(E^{(q)}\) (Equation (5)) was set manually so that two users Alice and Kim were chosen as the best experts with \(c_{Alice}^{(q)} = c_{Kim}^{(q)} = 0.8\) while all the others had \(c_{i}^{(q)} \leq 0.6\). The correctness \(W\) (Equation (6)), was filled with a constant value \((w_e = 0.9, i = 1, \ldots, n)\). The values of susceptibility in matrix \(\Phi\) (Equation (12)) were drawn with respect to the distribution presented in \([47]\).

Next, the various recommendation strategies were used, i.e., many different recommendation methods were studied. Each recommendation strategy estimated a recommendation matrix \(S\) (Equation (10)), using other matrices, such as \(P\) (Equation (3)), \(E^{(q)}\) (Equation (5)), \(W\) (Equation (6)) and \(\Phi\) (Equation (12)). Then, a modified policy \(\Pi'\) for each generated matrix was calculated using Equation (11). Finally, for each recommendation matrix the probability of getting an answer \(P^{(q)}\) was estimated using Equation (9).

The recommendation strategies were compared using the average answer probability for all 20 users.

7.3.2 Large Networks

The experiments on large SN consisted of over 2,500,000 runs of simulations, in total. In a single run, the average probability of getting an answer was calculated for a given network and all recommendation methods. The SN used in experiments were generated randomly for each run using the following process. In the first step, the network structure was created. The connections between network nodes were either extracted from Enron mail data set \([54, 55]\) or generated randomly using Watts and Strogatz algorithm \([56]\) or Barabási and Réka algorithm \([57]\). Networks generated using Watts and Strogatz algorithm are denoted by WS symbol and those generated using Barabási and Réka algorithm are denoted by BR symbol. The parameters of the networks are presented in Table 1. In the next step, each edge of the network had \(\pi_{ij}\) and \(r_{ij}\) coefficients assigned randomly using one of the seven probability distributions: \(\text{Beta}(1, 5), \text{Beta}(0.1, 3), \text{Beta}(0.5, 8), \text{Beta}(0.05, 3), \text{Pow}(2), \text{Pow}(2.5), \text{Pow}(3)\), where \(\text{Beta}(\alpha, \beta)\) is the beta distribution and \(\text{Pow}(\alpha)\) is the power law distribution. In the last step, each node of the network had \(w_i, w_j, s_i\) coefficients assigned randomly using one of four the probability distributions: \(\text{Beta}(1, 5), \text{Beta}(0.1, 3), \text{Beta}(0.5, 8), \text{Beta}(0.05, 3)\). For each generated network every recommendation method was applied and the resulting average answer receiving probability was collected.

Each combination of coefficient distributions was repeated 10 times and averaged. Hence, for each network, there were 376,320 simulations, i.e., 12 recommendation strategies x 7 probability distributions for \(\pi_{ij} \times 7\) probability distributions for \(r_{ij} \times 4\) probability distributions for \(w_i \times 4\) probability distributions for \(w_j \times 4\) probability distributions for \(s_i \times 10\) (\(12 \times 7 \times 7 \times 4 \times 4 \times 4 \times 10 = 376,320\)).

7.4 Experiment Results

In this section, the results of experiments are presented, separately for the small and large networks.

7.4.1 Small Network

The summary of experimental results is presented in Figure 6 and Table 2. They contain the average probabilities of finding an answer for the hypothetical query \(q\) using particular recommendation strategies.

The recommendation strategy influences the probability of finding the answer to a query by supporting
TABLE 2. Average probabilities of finding the answer in the small social network by means of different recommendation strategies ordered in an ascending manner

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>random m=3</td>
<td>0.4170</td>
<td>14</td>
<td>0.4305</td>
</tr>
<tr>
<td>random m=1</td>
<td>0.4222</td>
<td>15</td>
<td>0.4313</td>
</tr>
<tr>
<td>none</td>
<td>0.4233</td>
<td>16</td>
<td>0.4319</td>
</tr>
<tr>
<td>b.relat. m=1</td>
<td>0.4233</td>
<td>17</td>
<td>0.4324</td>
</tr>
<tr>
<td>b.ans. m=1</td>
<td>0.4241</td>
<td>18</td>
<td>0.4369</td>
</tr>
<tr>
<td>b.ans. m=3</td>
<td>0.4244</td>
<td>19</td>
<td>0.4455</td>
</tr>
<tr>
<td>random m=5</td>
<td>0.4243</td>
<td>20</td>
<td>0.4472</td>
</tr>
<tr>
<td>b.ans. m=5</td>
<td>0.4244</td>
<td>21</td>
<td>0.4479</td>
</tr>
<tr>
<td>b.ans. m=4</td>
<td>0.4254</td>
<td>22</td>
<td>0.4506</td>
</tr>
<tr>
<td>b.relat. m=2</td>
<td>0.4258</td>
<td>23</td>
<td>SQM</td>
</tr>
<tr>
<td>expert</td>
<td>0.4260</td>
<td>24</td>
<td>0.4544</td>
</tr>
<tr>
<td>random m=2</td>
<td>0.4270</td>
<td>25</td>
<td>max</td>
</tr>
<tr>
<td>random m=4</td>
<td>0.4302</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Both of them are experts in a domain of a query. SQM and “best answering” methods recommended Tom most often. Tom is friendly and often responds to queries. The “max” strategy generated a very interesting recommendation for Lisa and Rick; they were recommended to 16 and 17 people, respectively. The main goal of such recommendation is not to increase the probability of passing the query to the recommended people but to decrease the probability of users who are not recommended.

7.4.2. Large Networks

The results of the experiments carried out on large SN are summarized in Table 3. They provide different results than those of the experiments on the small network. The increases in the average probability of finding the answer are much lower than that in the small network, as they reached only 1% relatively. The actual efficiency of the recommendation strategies also changed. In large SN, recommending the neighbors having the strongest relation (highest policy πij value) to user ui, seems to give the best results. On the other hand, SQM strategy performed the worst.

Table 3 presents the summarized results of the experiments on large random networks. Each cell of the table contains the percentage of experiment runs in which the given method performed the best in the given network. The results presented in Table 3 confirm, that both for the random networks and for Enron, the “best relation” strategy performs best. In the Enron network, the difference between this strategy and all the others is much more visible - “best relation” strategy was the best one in 37% of experiment runs. The results indicate, that the efficiency of “best relation” recommendation strategy is positively correlated with the density of the SN (see Figure 8).

The use of the best recommendation strategy for each experiment run would increase the probability of finding the answer by 5.34% on an average. In the best case, the increase reached as much as 192.38%. In the worst case, no influence on the probability was measured.

8. DISCUSSION

Let us assume that user uj has typed question q to the SocLaKE system. Next, question q is spread over the community and through several different paths reaches the same final user, expert uj, which means that user uj is given a set of requests from couple of acquaintances simultaneously. Obviously, the SocLaKE system should detect such cases and manage them. Getting the same message from many people may be annoying, specially for busy experts. Hence, messages containing the same query q should not be presented as distinct items on the user message list. On the other hand, duplicate messages should not be ignored completely. Very important is the time factor. If all messages with question q have arrived to user uj before he/she sends...
TABLE 3. Summary of recommendation strategy performance for large networks, i.e. percentage of all simulations (See No. of simul column of Table 1) for which a given strategy was the best

<table>
<thead>
<tr>
<th></th>
<th>b.ans.</th>
<th>b.ans.</th>
<th>b.ans.</th>
<th>b.ans.</th>
<th>b.ans.</th>
<th>expert</th>
<th>b. rel.</th>
<th>b. rel.</th>
<th>b. rel.</th>
<th>No rec.</th>
<th>SQM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m=1</td>
<td>m=2</td>
<td>m=3</td>
<td>t=0.5</td>
<td>t=0.7</td>
<td>t=0.9</td>
<td>m=1</td>
<td>m=2</td>
<td>m=3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS n1000in10rp01</td>
<td>0.2%</td>
<td>0.2%</td>
<td>0.3%</td>
<td>0.6%</td>
<td>1.9%</td>
<td>7.0%</td>
<td>4.0%</td>
<td>26.6%</td>
<td>19.8%</td>
<td>26.0%</td>
<td>14.2%</td>
</tr>
<tr>
<td>WS n1000in10rp02</td>
<td>0.2%</td>
<td>0.2%</td>
<td>0.1%</td>
<td>0.3%</td>
<td>1.8%</td>
<td>6.9%</td>
<td>5.6%</td>
<td>23.2%</td>
<td>19.8%</td>
<td>26.6%</td>
<td>15.1%</td>
</tr>
<tr>
<td>WS n1000in10rp03</td>
<td>0.8%</td>
<td>0.3%</td>
<td>0.3%</td>
<td>0.7%</td>
<td>1.7%</td>
<td>4.6%</td>
<td>5.3%</td>
<td>26.2%</td>
<td>19.9%</td>
<td>26.9%</td>
<td>13.1%</td>
</tr>
<tr>
<td>BR n1000m2</td>
<td>5.7%</td>
<td>9.2%</td>
<td>9.2%</td>
<td>8.5%</td>
<td>9.2%</td>
<td>9.3%</td>
<td>9.1%</td>
<td>12.0%</td>
<td>9.2%</td>
<td>9.2%</td>
<td>9.2%</td>
</tr>
<tr>
<td>BR n1000m3</td>
<td>3.7%</td>
<td>4.1%</td>
<td>9.8%</td>
<td>8.8%</td>
<td>8.8%</td>
<td>9.2%</td>
<td>9.8%</td>
<td>15.4%</td>
<td>10.7%</td>
<td>9.8%</td>
<td>9.8%</td>
</tr>
<tr>
<td>BR n1000m5</td>
<td>3.7%</td>
<td>2.9%</td>
<td>3.8%</td>
<td>6.4%</td>
<td>6.9%</td>
<td>7.7%</td>
<td>8.5%</td>
<td>22.7%</td>
<td>14.0%</td>
<td>14.9%</td>
<td>8.5%</td>
</tr>
<tr>
<td>Random average</td>
<td>3.5%</td>
<td>4.9%</td>
<td>6.2%</td>
<td>6.1%</td>
<td>6.8%</td>
<td>8.3%</td>
<td>8.1%</td>
<td>17.7%</td>
<td>12.9%</td>
<td>14.8%</td>
<td>10.6%</td>
</tr>
<tr>
<td>Enron</td>
<td>1.8%</td>
<td>1.2%</td>
<td>1.4%</td>
<td>1.0%</td>
<td>1.5%</td>
<td>3.2%</td>
<td>4.6%</td>
<td>20.9%</td>
<td>17.9%</td>
<td>37.3%</td>
<td>9.3%</td>
</tr>
</tbody>
</table>

the answer to user $u_i$, the system should inform $u_j$ that he/she was found as an expert in the domain of question $q$ by many persons. That may encourage user $u_j$ to react.

In many cases, we want to prevent ourselves from waiting for too long for an answer. If we assume that the average user reaction time is about 2 working days, a path built from 5 steps gives us 2 weeks of waiting for an answer, which may be too long for urgent problems. Hence, SocLaKE system should allow the user to specify an additional parameter for the query - “time to live”, after which the query should be withdrawn from the SN. In this way, we can omit bothering experts by overdue questions. It may also be useful to define the general system timeout for queries. This eliminates floods of out-of-date questions. Query floods are one of the biggest threats in a query forward system such as SocLaKE. A query flood may occur when too many users are willing to forward questions. Application of the limit for the number of active users per query may prevent floods and improve the system performance. On the other hand, small values of such a constraint may decrease the overall probability of retrieving a suitable answer.

Overloading experts is a well-known phenomenon in EFS. Experts are rare in SN. It is not surprising, therefore, that experts may receive more questions than they can handle. Therefore, yet another task for the SocLaKE system is to prevent experts from overloading. The solution may be the default limit on the number of active queries per user. Having in mind the user preferences, the system may also allow the user to increase or decrease their personal limits. Organization members are in most cases busy people. The system task is not only to help them to find an expert in their social environment, but also to save their time and make the whole process as easy as possible. Overall, people tend to forget about other people’s problems while they have their own. It is easy to imagine that user $u_i$ after receiving a message with the query from user $u_k$, passed it further to user $u_j$ and forgot about the problem. However the problem becomes complicated if user $u_k$ takes a 2 week holidays. If neither $u_i$ nor $u_j$ has passed the query to anybody else, user $u_j$ has no chances of receiving any answer for 2 weeks. We can prevent such situation by establishing the “user timeout”, where the system would inform user $u_j$ that time has expired for user $u_k$ and maybe $u_j$ should pass the query to someone else.

The main goal of SocLaKE is to provide to users the solutions for their problems. However, what happens after receiving the answer is also important. It is not obvious that the answer will satisfy the user. Hence, the decision of whether the query should be canceled from the system after receiving the first answer should belong to the initial user. We can go even further - what if the user does not know whether the answer is good or not? An opportunity of comparing few answers may be a precious feature. Nevertheless, all evaluations of the obtained solutions need to be stored in the system and have influence on the expertise of the answers’ author. A good evaluation should increase the authors’ expertise, while bad one could lower it.

Most social relations have one common factor - the factor of expiration or time factor. It is quite easy to measure the reinforcement of social relation strength - we simply boost it at each time when two individuals contact each other. However, in the real world, people, who used to work together few months ago, may not remember each other now. If they have stopped mutual communication, the system should detect this fact and decrease the strengths of their relations or even cancel them at all.

Utilizing the data generated by the SocLaKE system is even more informative than mapping the existing data sources from legacy systems. Let us consider user $u_i$ searching for a given solution. Initially, $u_i$ does not know the answer and is not an expert as well. However, after $u_i$ receives the solution from expert $u_j$, and accepts it as the proper one, user $u_i$ also becomes, to some extent, an expert in the domain of the query. After successful evaluation both the expertise of user $u_i$ and $u_j$, increases. Therefore, if someone else asks the same or similar question, both users $u_i$ and $u_j$ should be
considered as experts for that query. Also, a bad answer evaluation can be useful. If the system points user $u_j$ as an expert and he/she provides a wrong answer (badly evaluated by the question initiator), then the level of expertise of $u_j$ should be decreased. Certainly, there is a risk that the author of the question incorrectly evaluates the answer (or does not understand it), while the answer is right.

Additionally, data about the answering procedures gathered by the SocLaKE system, may be used by this system to recalculate appropriate matrices and adapt the system to recent user behavior. For example, if user $u_i$ usually responses (in the SocLaKE service) to user $u_j$ requests, then the responsiveness probability $r_{ij}$ may be greater, even though $r_{ij}$ computed based on mutual email exchange is low. The same also holds for evaluations of other probability matrices. Moreover, time factor may be applied to damp older data and emphasize the new one (see [54] for an example).

Nowadays, the personal data and privacy of computer users are highly protected by the law and companies internal regulations. The SocLaKE system operates on artifacts created freely by users, especially on data about their communication. Therefore, it is important to respect both company privacy policy and individual concerns. As a result, a user should be able to switch off recording their queries and answers obtained.

It is also essential to make the user interface (UI) of SocLaKE as easy and intuitive as possible. Therefore, SocLaKE should be integrated with email clients or other communication tools. The general idea is to make SocLaKE almost invisible for end users. A simple but effective solution is the pop-up list of email addresses, which appears every time a user types message recipients. The SocLaKE system analyzes the message content at real time and provides the best possible candidates for addresses.

The user susceptibility $\Phi$ is not a constant factor as assumed for the experimental purpose, see Section 7.3. Overall, it may vary depending on the user mood, question stage (are there any answers so far), other persons’ opinions, experience with the SocLaKE system, accuracy of the answers received so far and many more.

In the simulation, it was also assumed that the user policy $W$ are query-independent. However, in the real world, they may change according to the question domain. One can be a very good mathematician and may have a high answer correctness rate in the mathematical domain, while the same person, as an amateur sociologist, possess high expertise but a very low correctness rate in the sociology domain.

The model introduced in Section 5 assumes that the probability of getting a correct answer to the question asked in a SN depends on a few simple coefficients. In Section 5.3 the trivial methods of estimating these coefficients have been proposed. Obviously, the proposed methods have not considered many important issues related to UI or psychological aspects. A real-world system needs to develop much more sophisticated estimation methods.

The results obtained during experiments come from simulation. The comparison of the proposed approach and classic EFS needs to be verified in a real-world environment, because EFS require much more data than the model introduced in Section 5 provides.

9. CONCLUSIONS

The problem of sharing tacit knowledge inside organizations has been recognized as a key issue to modern knowledge-based companies. This study has addressed one aspect of this problem, namely sophisticated supporting communication inside the SN of the company. The novel SocLaKE system has been proposed to carry out this task by means of recommendations. SocLaKE combines the knowledge about expertise together with social relations between members of the organization to provide its users with usually hardly accessible information of whom to ask for help when dealing with a certain topic. Besides, SocLaKE can be easily incorporated into an organization’s communication infrastructure and transparently support communication between its members.

The most important part of the system is the recommendation strategy. This component generates recommendations for a given user and query. A few different strategies have been examined and compared in this paper. Additionally, to evaluate the strategies, the SQM with recommendations has been introduced. According to the simulation performed, almost all the proposed strategies improve, although to different extents, the probability of finding the answer.

ACKNOWLEDGEMENTS

The authors are indebted to the anonymous reviewers for their valuable remarks.

The work was supported by: The Polish Ministry of Science and Higher Education, the development project, 2009-2011, The Polish Ministry of Science and Higher Education the research project, 2010-2013, and the fellowship co-financed by The European Union within the European Social Fund.

REFERENCES


FIGURE 5. Experimental matrices summary for the small social network
FIGURE 6. Gain in average probability, when compared with “none strategy” in the example small SN.
FIGURE 7. Comparison of recommendations generated by the best three strategies from Table 2 for the small SN shown in Figure 3: SQM recommendation (a), Best answering with st=2.0 (b) and Max (c); a black field in (a) – (c) denotes a recommendation of the user from the column for the user from the row; the darker field in (d) the more strategies recommended a particular user.