Expert User Discovery in a Spontaneous Social Network
An approach using Knowledge Retrieval

Gustavo Freitas, Cristiano da Costa, Jorge Barbosa, Rodrigo Righi
Applied Computing Graduate Program - PIPCA
University of Vale do Rio dos Sinos - UNISINOS
São Leopoldo, Brazil
e-mail: gustavof@gmail.com, {cac, jbarbosa, rrrighi}@unisinos.br

Adenauer Yamin
Graduate Program in Computer Science - PPGC
Federal University of Pelotas - UFPEL
Pelotas, Brazil
e-mail: adenauer@inf.ufpel.edu.br

Abstract — Nowadays, we can observe that finding answers on social networks is a hard and time-consuming task. The main contribution of this article is the creation of a model and algorithm that allows users to find answers to their questions using a spontaneous social network called Mingle. This algorithm uses the Mingle ontology-based knowledge base to find expert users. To achieve this, two important steps are taken: Mingle ontology is updated to support user-oriented expertise and a detailed model is created for the given algorithm. This model was created considering other three similar applications and algorithms. Moreover, we used a semantic web framework. In the end, an evaluation using real-life scenarios is applied to evaluate if the created algorithm meets the initial goals.

Keywords - Expert Users; Social Networks; Knowledge Retrieval; Ontologies.

I. INTRODUCTION

A social network is a set of possibilities formed by recurrent relationships between individuals [1]. Mingle [2] uses this assumption and smartly draws the range of a given social network. New connections are formed by the geographic location of individuals, content is distributed and services are offered. Spaces are provided for new ideas and relationships to flourish. All these scenarios mentioned before are covered by this spontaneous platform. In this sense, this work tries to answer how we can easily access given information inside a social network with these characteristics.

From a knowledge base (KB) specified by a given ontology [3], information can be found and used for answer the question proposed before. Also, algorithms with similar goals can be used to give an important input for any solution creation. Starting from the obvious assumption that social networks are made by people, the answer for this very question could lies precisely on these individuals.

What really matters when we build applications that handle knowledge is the construction of this knowledge constantly and distributed [4]. Moreover, these applications get better and more precise according to the number of users using it. If we have more people using an application and exchanging information, the knowledge base will increase as well. Consequently, the systems will become broader. In this way, another question comes in mind: how to provide conditions that every application user has access to this knowledge?

Having said that, the main goal of this work is the development of an algorithm that finds expert users inside the spontaneous social network called Mingle [2]. In the given context, the definition of expert users is a user that has great knowledge in one specific domain. So, qualified information can be easily found. To achieve this goal, three similar algorithms were studied and analyzed. Moreover, Mingle's ontology was updated to allow that these users could be found without any issues. Starting from the studied algorithms and the updated ontology, a new algorithm was proposed. In the end, the obtained results were evaluated according scenarios that simulate reality.

The major contribution of this article is the development of a model and an algorithm that uses semantic to retrieve knowledge from a spontaneous social network. Also, this work tackles different scenarios regarding social network development within knowledge retrieval concepts.

This article is organized in the following sections. Section II presents the main foundations of this work. The Mingle project is described in section III. In section IV, we discuss related works. In section V an algorithm model is proposed. Section VI describes the developed prototype and section VII shows an analysis of results. Finally, section VIII presents some conclusions, lessons learned and possible future works.

II. BUILDING KNOWLEDGE

Collaborative tools are conceived to allow a group of users to solve/achieve a given problem or state. These tools are known as computer systems that support a set of people in a common task, providing a shared environment [5]. A kind of collaborative tool, very popular nowadays, is a social network, composed of services that establish and reflect social and human aspects between people [6]. Social networks are structures that are in a constant flux of change, being that very transitional property something that differentiates it from any other human phenomenon [1]. That said, we understand collaborative tools and social networks as platforms that enable complimentary collaboration, where building and obtaining knowledge are always the main foundations to any action inside the given network.
Mobile computing is a reality nowadays [7]. Users with mobile devices interact between each other very frequently. Also, ubiquitous computing [8] - computers “invisible” in any given context, it is something that is getting into our daily base routine. This future would be a context where computers systems tendencies would be to stay on the background, emphasizing on the information and the integration of the various devices and infrastructures. Furthermore, if we gather these concepts together within the social network ideas’ floating around, a perfect scenario is established when we need answers for any given situation [9].

Knowledge retrieval is a topic related most of the time to store and organize information in specific data structures. Usually, this task can be successfully done using ontologies [10]. There are many other approaches and techniques to obtain knowledge from any information stored inside a computer, but when this data is specified by an ontology, a much more clear and consistent data could be obtained [3][4].

Ontologies formally represent knowledge, allowing a computer to logically infer new information about this conceptual and previously described data [3]. Most of the time, this knowledge is described through a set of concepts from a given domain. Ontologies enable the existence of a unique and exact vocabulary inside a specific domain [4]. When creating an ontology it is possible to use descriptive languages, such as OWL (Web Ontology Language), which formally specify how the knowledge will be stored.

III. MINGLE SPONTANEOUS SOCIAL NETWORK

Mingle [2] is a proposal that shares all these ideas of mobile and ubiquitous computing. This project has as the main characteristic the use of the user physical location, allowing that people in a given area to exchange ideas and services. When a user gets into a specific location, which is managed by Mingle, this person is automatically part of the social network formed by the people that are also present in this very location.

Inside Mingle, users can anonymously participate or they can share their location and profile for specific people if they want. Every cell, a group of users inside a specific context, has one or more nodes that offer services and contents related to that actual physical location, called site. Therefore, every person that is connected to this specific location can offer services and contents as well.

Another important point in Mingle is the possibility of having customized services and contents according users profiles. These profiles are user specified and, also, they are persistent among different contexts inside the system.

Lastly, a Mingle ontology structure was suggested [11]. Also, all the classes were organized within three big groups. Each set representing some specific functionality regarding classes’ responsibility. These sets are: Context: only classes related to platform actual context; Profile: only classes related to user information; Device-Info: only classes related to platform devices (hardware and software).

IV. RELATED WORKS

In this section, we present three articles studied for the development of our model and algorithm. These articles were fundamental to the progress of the current work. Furthermore, important concepts and ideas are highlighted.

The main goal of the Spree Expert Finding System is the development of a web application that enables users to easily share information, where expert users (reliable people) are found in a given domain [12]. These interactions happen essentially through questions made by the users to the application. Every user is considered in this context, there is no distinction between who is asking and who is answering the questions. In this way a user community is built, allowing that the knowledge base comes from these relationships. The platform basic flow happens according the following steps: questions are always inserted through textual inputs. Additional properties, as knowledge domain, are attributed to each question. An algorithm searches the Spree ontology for key words from these textual inputs. After that, user profiles are considered to find appropiate users to answers the questions. Finally, the application puts in touch the user that created that specific question with the set of users that can potentially answer this question.

The second work considered proposes an algorithm, called SPEAR (Spamming-resistant Expertise Analysis and Ranking) [13]. This algorithm classifies users according their expertise inside a collaborative system that uses tagging, identifying expert users and relevant documents inside a given system. The main idea here is the construction of user rank based on a taxonomy build by the very users; this taxonomy is also called folksonomy. The basic flow of the SPEAR algorithm is: as the main entry we always have tuples of user, keyword (tag), content and date/time that the given user tagged the specific content. At this point, data structures responsible to store quality scores are created. These structures are used to determine the quality level of each document and the expertise level regarding each user. In the end, it is calculated the level of knowledge of each user and content.

Finally, the third article considered proposes a search engine based essentially on users relationships. This tool is called Aardvark [14]. From a user question, the application looks for the most reliable person to answer it. Only users inside the user social circle are considered. Apart from the level of the user knowledge, the level of proximity and relationship are considered as well. Aardvark is based on the following point: a subjective question is better answered when data needed to answer this question is related directly to exchanging information from one person to another. The main Aardvark components are: the crawler and indexier, to find and index information, the question analyzer, which interprets the user needs, the ranking functionality, which selects the best resources and, finally, the UI that shows reasonable information to the final user.

V. PROPOSED MODEL

A model and an algorithm in charge of finding expert users are proposed. The main goal of the algorithm is to
make sure that knowledge will easily flow and be obtained inside Mingle spontaneous network. One of the most important mechanisms that are used in the proposed algorithm is the concept of associating knowledge domains (expertise) to platform resources. It was developed through the addition of an "Expertise" class in Mingle ontology. Other important inherited mechanism from the related work was the possibility to define users that first discover and share resources inside the network. In this way, it is possible to define if these users are potential sources of information.

Before any description, we define two important concepts that will be very important to this current work: resource and quality. In Mingle, resource is everything that a user offers to future use. Here, we are focusing on contents (pictures, texts, sounds and videos) and services. Quality is defined using a specific score, where when we have higher scores, we have special and unique characteristic set to a given resource.

The proposed algorithm will only gather information from the knowledge base. The actual data, such as quality rating score, users and services are stored beforehand on the knowledge base previously to any algorithm calculation. At the end, the algorithm writes information back on the ontology. The initial proposed model is organized in four main steps (summarized in Fig. 1). These steps represent phases of the algorithm. Each step is related to a class and different data outputs. Most of the time, these outputs are used as inputs to the next step.

The first step, which selects every document and service related to a given context, always has the same input: context (could be a cell or site) and expertise (a specific knowledge domain related to resources). Resources are selected and a list ordered by quality is created. Here, we are going to use the rating property to quantify the quality of a resource. Also, a list of users related to these resources is created. This list is ordered by quality as well.

The second step defines the average quality level of context resources. Using the documents and services found in the first step, a simple average mean is calculated using the number of resources found and the sum of the rating value of each resource. The idea of average quality regarding a specific context is to be sure about the quality of the selected resources. This average value will be used and better explained in the forth step.

The third step connects resources and users. A bi-dimensional matrix is created to support these data relationship. If a given user is related to a given resource that specific matrix position will be set to a positive integer value, otherwise zero will be used. Expertise is attributed according to each resource inside this matrix.

At the fourth step, these integer values are incremented according to quality resource aspects. When highest quality resource is considered, higher will be the integer number of a specific matrix position. The average quality of resources, calculated in the second step of the model, will be used to determine the actual level of resource quality. Resources that have a quality score greater than the average are considered reliable. Resources that have minor scores than the average are considered less reliable. Lastly, we have a final enhanced matrix containing users, resources and final quality scores. From this data structure it is possible to infer which users are more prone to contribute with quality documents and services in a specific domain inside a given context.

Table 1 shows an example of the final matrix calculated by the algorithm. Note that the expertise is not presented on this final matrix because all the resources listed are related to just one particular domain already specified on the initial phase. Users that have a significant number of resources with higher scores (considerable quality) are considered experts.

<table>
<thead>
<tr>
<th>TABLE I.</th>
<th>EXAMPLE OF A FINAL MATRIX CONTAINING USERS, RESOURCES AND QUALITY SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>User 2</td>
<td>0 1 0 0 0 0 1</td>
</tr>
<tr>
<td>User 3</td>
<td>0 0 -1 0 0 0 0</td>
</tr>
<tr>
<td>User 4</td>
<td>0 0 0 1 0 0 0</td>
</tr>
<tr>
<td>User 5</td>
<td>0 0 0 0 0 1 0</td>
</tr>
</tbody>
</table>

VI. IMPLEMENTATION

To bring the proposed model to reality, some important development tools were studied and used. The ontology was stored using OWL. To read data from these files, the Jena library was used. This library is written in the Java programming language and allows reading/writing models, querying information, and inferring new information from the model. With the objective of querying the model, the Jena library have a SPARQL engine, which reads rules written in this language and searches the knowledge base. The implementation of the prototype was done using the Eclipse IDE, with the Java programming language. This language was chosen because most of the Mingle project is already developed using it. The prototype uses the PostgreSQL 9 for storing the knowledge base.
In the development of this algorithm, three main classes were created using the proposed model presented before as the main foundation. These classes are **ResourceSelector**, **QualityAnalyzer** and **ExpertSelector**.

The first class gathers every available resource in a given context. From this specific context and a given expertise, these resources are ordered by quality and they are stored inside an array. At the same time, another array is created to store every user related to these resources. This class has two main methods `getListOfResources` and `getListOfUsersFromResources`.

The second class defines the average quality level of resources inside a given context. From the data gathered before by the **ResourceSelector** and the **rating** property that comes from every resource, an average mean stating the resources quality is calculated. This average mean is calculated using the sum of the quality of every resource related to a specific expertise and the total number of resources. This class has one main method `defineAverageQuality`.

The third and last class associates every resource to a user, through the creation of another matrix. The arrays previously created are going to be used now to populate this new matrix. One dimension of this new matrix contains resources and another dimension contains users. Positive integer values are attributed to a specific position if a given resource is related to a specific user. Otherwise, zero is attributed to that specific matrix position. Moreover, according the average mean calculated before, these specific positions could have negative and positive integer values, stating that a specific resource is reliable or not to be considered. If a resource has a quality score below the average mean, the final quality score to that specific resource is set to a negative integer (-1). If a resource has the quality score above or equal to the average mean, the final quality score is set to a positive integer (+1).

Furthermore, the same class defines who is the expert user in a given domain of knowledge. This is achieved through the matrix previously created. According to the mechanisms previously explained, this class has two main methods `createKnowledgeMatrix` and `defineExperts`.

Fig.2 shows, using a sequence diagram, how these relationships between classes and data structures happen inside the proposed algorithm.

**VII. EVALUATION AND RESULTS**

Considering the related works and all the different evaluation methods, this article used the following evaluation approaches: diverse user data creation and the establishment of scenarios based on reality according to possible Mingle uses. These evaluations methods were chosen because they were applied and successfully used at various similar approaches [11][12][13].

The first step was to create relevant and consistent data for Mingle knowledge base. This task was accomplished using Protégé, an ontology editor and knowledge-base framework. The data mass created was organized in the following way: a diverse pool of users (expert users, users with little knowledge and users with no knowledge at all) and, of course, service and resources with lots of different quality scores. Also, all this data were created with the future scenarios and use cases in mind.

![Figure 2. Classes, methods and data related to the proposed algorithm.](image)

**A. Scenarios Creation**

We created three specific use cases for evaluating the proposed model. This use cases were modeled with situations close to reality in mind. Three main scenarios were raised as feasible Mingle contexts: algorithm being used inside a mall, algorithm being used inside a public space (a park) and algorithm being used at a University. These scenarios will be better explained in the next paragraphs. Similarly, the algorithm execution results will be detailed for each specific situation in the next section.

The mall context was the first envisioned. As commercial areas are always related to exchanging information, it was appropriate to try out our algorithm in such environment. Two scenarios were raised (Fig.3): one when a customer wants a specific service related to a given domain and another scenario when a shop owner wants to offers to the public some kind of product or service related to a specific domain.

![Figure 3. Mingle scenarios regarding a commercial area.](image)

The public space context was envisioned following an idea of how Mingle should behave inside a big geographical area. As parks are visited by a large number of people and, usually they have a considerable area, it was appropriate to try out our algorithm in such environment. Two scenarios were raised (Fig.4): one when a person wants to find out more people to practice a specific sport and another scenario
in which a medical emergency happens and skilled users are notified and listed as the nearer help for the given problem.

The University context was envisioned following a simple idea of how Mingle could be used inside a University Campus. As universities are places where people are always exchanging knowledge, it was natural to try out our algorithm in such environment. Two scenarios were raised (Fig.5): one when a student looks for others students trying to find specific articles and another scenario where a teacher looks for skilled students.

**B. Execution**

To execute these three scenarios, a testing application was created. The main goal of this application was to simulate a user behavior when using Mingle. This application instantiates every class mentioned before (**ResourceSelector**, **QualityAnalyzer** and **ExpertSelector**) and returns to the console a list of users more prone to contribute with quality information.

Apart from that, the testing application is also responsible to load the knowledge base and store it temporarily inside a relational database. This is achieved through a local OWL file that is loaded as a model inside PostgreSQL. Finally, all the knowledge operations and queries are executed on this persistence using SPARQL.

Additionally, use cases related to the scenarios created were always executed using two important inputs: context and expertise. The context was used to simulate where the user is located (in this case, it could be a cell or a site) and expertise was needed to set what kind of domain the user is interested. These scenarios were run iteratively, one after another, only changing the set of context and expertise.

Regarding the shopping mall scenarios, the algorithm execution output was a list of users. These users were ordered by expertise level, meaning that the first user of the list is considered the expert user in that given context (Fig.6).

We could observe that the execution of these two different mall scenarios got the better experts in the proper order. However, important issues were found. In the first use case (customer use case), we were able to notice that it is very important that not only contents get related to specific domains. Some of the services inside the knowledge base did not have any expertise information associated, so, they were not even considered as options for the customer. Moreover, as our algorithm is in charge of defining who the expert users inside a given context are, the algorithm output was a list of users and not a list of services. However, it is possible to get the best services in given context using the expert users found.

When running the second use case, we realized that there is no possibility to offer real products using Mingle ontology. It happens because the ontology still does not provide classes related to products itself. So, instead of a product, a service was used. Also, as the use case was too specific about setting a given data on the Mingle knowledge base, the algorithm did not have any data to write back on the knowledge base. In the end, only a user related to high quality resources was returned.

Regarding the park scenario, the algorithm execution output was a list of users. These users were ordered by expertise level, meaning that the first user of the list is considered the expert user in that given context (Fig.7).

The execution of these two different public area scenarios also returned the correct experts. No major issues were found here. In the first use case, the medical emergency use case, users that had contents related to health expertise were considered as potential sources of information. When running the second use case, as we only change the necessary expertise, we found out the same results from the last use case. Users most prone to play a specific sport were selected.

Regarding the academic scenarios, the algorithm execution output was presented in Fig.8.
we could notice that every article studied sees the knowledge preferred when we need to define expert users. In addition, some data structures and certain specific mechanisms are performance inside an application. There are incredible rich concepts but at the same time they could results was the use of Mingle knowledge base. Ontologies besides, one of the biggest challenges to properly deliver specific and not a complete contribution to this proposal. achieved.

This article presented the model of an algorithm to discover expert users inside a spontaneous social network. In order to create this model, a review and an evaluation were made in every related work. Besides that, the developed algorithm was created according to requirements and specific scenarios. Lastly, we verified that the initial work goal was achieved.

Regarding future work, we could have achieved better results using real user data from Mingle. Because, as the data created was controlled, we almost not bumped in interesting edge cases, such as performance hiccups and model issues. Also, we plan to integrate this algorithm within Mingle services and its mobile clients. Furthermore, a natural language processor could be added to parse any user input and transform it in something meaningful to our model. Another possibility is the development and evaluation of the same algorithm but decoupled from a knowledge base. Following current practices, a non-relational database could be used.

ACKNOWLEDGMENT

We would like to thanks Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for funding this research.

REFERENCES


