Sharing personal knowledge over the Semantic Web

Zlatko Kostadinov
Faculty of Mathematics and Informatics,
Sofia University St. Kliment Ohridski
e-mail: zlatko@fmi.uni-sofia.bg

Keywords
Distributed decision support, Semantic Web, Knowledge representation, Networks of Practice

Abstract
Every day humans solve a lot of common tasks and find out a lot of similar knowledge. In common sense sharing this knowledge could save them efforts and improve their decisions. In the global community of human beings there exist subcommunities of persons oriented to specific problems. These communities develop knowledge in time and share this knowledge to achieve better decision making. What is common within knowledge development and sharing is that the “Research has consistently shown that social relationships are important to the ability of individuals to gather knowledge and to perform their work and that the creation of knowledge is innately a social process among individuals” [1, p. 2]. These communities were investigated in the early 90’ s by Lave and Wenger [2] and called Communities of Practice (often abbreviated as CoPs). They described the Community of Practice as “… a set of relations among persons, activity and world, over time and in relation with other tangential and overlapping CoPs”. Later John Seely-Brown and Paul Duguid [3] developed the concept of Networks of Practice (often abbreviated as NoPs). The later work in the area of knowledge sharing and representation is concentrated around the term ontology. Ontologies are “specifications of conceptualizations” [4]. In 1998 Tim Berners-Lee [5] presented his view of a new network of shared knowledge calls Semantic Web, built over the current Web, that should enable the AI applications to share knowledge in the most powerful way currently provided by computer technologies.

1. Introduction

Every day humans do a lot of similar stuff, solve a lot of common tasks and find out a lot of similar knowledge. The problems solved could be community-oriented like arranging a common space or creating a common document, or person oriented problems like planning a weekend or tuning a software application; business-oriented like an insurance agent support system or non-profit based like a mutual aid forum. We state that the current situation in the field of computer network technologies and social networks development is bringing up the need of such applications in the limelight.

In common sense sharing this knowledge could save people efforts and improve their decisions. In the global community of human beings there exist subcommunities of persons oriented to specific problems, that develop knowledge in time and that share this knowledge to achieve better decision making. What is common within knowledge development and sharing is that the “Research has consistently shown that social relationships are important to the ability of individuals to gather knowledge and to perform their work and that the creation of knowledge is innately a social process among individuals” [1, p. 2]. These communities were investigated in the early 90’ s by Lave and Wenger [2] and called Communities of Practice (often abbreviated as CoPs). They described the Community of Practice as “… a set of relations among persons, activity and world, over time and in relation with other tangential and overlapping CoPs”. Later John Seely-Brown and Paul Duguid [3] developed the concept of Networks of Practice (often abbreviated as NoPs). The later work in the area of knowledge sharing and representation is concentrated around the term ontology. Ontologies are “specifications of conceptualizations” [4]. In 1998 Tim Berners-Lee [5] presented his view of a new network of shared knowledge calls Semantic Web, built over the current Web, that should enable the AI applications to share knowledge in the most powerful way currently provided by computer technologies.

2. Characteristics of the examined networks

Specifics of the networks of practice

Lave and Wenger [2] use the term Legitimate Peripheral Participation (LPP) to describe the process of interaction between the actors in a NoP. It refers to the complicated process of joining, using and contributing to a NoP and turning the newcomer into an experienced member of the community.

The initial research of the local CoPs showed, that the face-to-face knowledge exchange strongly
depends on the stable ties, co-location, agents similarity and the prior relationships [6]. With the process of economic globalization and company internationalization the distributed and electronically organized CoPs came in front [7]. Bourhis, Dube and Jacob [8] defined Electronic CoP as “a CoP whose members use information and communication technology as their primary mode of interaction. ...Being virtual does not exclude the use of face-to-face meetings, but several factors such as geographical dispersion and busy schedules, make communicating through ICT much more efficient.”. According to Landqvist and Teigland [9] the heterogeneity of the actors and resources and the quality of the social ties in the NoP have significant role in the process of knowledge development. Usually the knowledge shared in a community of practice is not formalized, sometimes it even could not be formalized, it is mostly “an unrecognized resource held in the minds of workers” [7]. This knowledge is usually called tacit knowledge, in opposite to the explicit knowledge, which is usually formalized, oriented to a specific domain and therefore – easy to process by machines. As Edvinsson and Malone [10] say “Tacit knowledge is highly personal and hard to formalize, making it difficult to communicate or share with others”.

Informal description of the examined networks

Not every NoP could fit in a formal model and not every NoP could be spread over a computer network. So we define a subset of NoPs could take advantage from the computer network shared knowledge. These NoPs should fit the above description – they should contain of a group of people that solve a similar problem, every person works to produce a (part of) problem decision. The problem decision is based on knowledge about the concrete problem field. So we formalize those features of the NoP, that we consider to stay in the foundation of the successful knowledge sharing community based on a computer network:

1* Users have basic computer access and literacy. Even the best software application could not help people that do not want or can not use it at all.

2* Users are autonomous both in acting and evaluating the results. In the common case the desired decision could be different for the different persons or subgroups of persons, but they share the same problem domain, so they could share a domain-specific knowledge to help the problem solving.

3* Users like each other. Increasing the quantity of the knowledge shared will increase the benefit from the network. We do not consider cases where the users have contrary interests. Of course in a bad system more knowledge could make things worse, but we want to create the best system anyway.

4* Users like each other in a different way. In a NoP there could be very complicated relations between the persons. We would like to describe these relations as far as they affect the knowledge sharing and using process.

5* The knowledge domain is a subject of a formal description, but the knowledge shared could be incomplete. This does not mean that the knowledge shared should be well described in a formal language, but this means that the users should be able to share some formally described knowledge about the problem field.

6* Computation technology... We go further than Dube at Al. and define our NoP as an electronic NoP whose members use computation technology for knowledge exchange and development. We reckoned the current electronic NoPs do not take the best advantage of the current computation technologies and suggest how to change this.

3. Software application model

Below we define mathematical model of an application that could be used to share person-specific knowledge on a common tasks and to take advantage of the share by automated development and usage of new knowledge. Such applications do not exist yet, and, as far as we know, there have not been neither developed, nor defined any at all. The model we define conforms the features 1* to 6* and therefore could be applied to any problem field that fits in this features.

According to 5* we have formal description of the problem and the related knowledge and this description is presented in a machine-readable
between the users (4*). These relationships are complicated. We know there are complicated relationships between users. For example, the rule action could be present as a transformation of the system state before the rule execution to the system state after the rule execution. So our agent needs to find an appropriate series of transformations \( t_1^q, ..., t_n^q \), such that \( t_1^q:Q \rightarrow O \) and \( t_n^q(t_{n-1}^q(... t_1^q(q))) = u(q) \) for as many \( q \) as possible (5*). Basically our agent should find out a procedure \( P \), that will provide this series for a specified \( q \). The transformations are kept in the knowledge repositories of the underlying user and the other users. For evaluation of the result is used \( u^* \) function which we know. We know the knowledge shared is unreliable or incomplete (5*). Thus we nominate a factor of trust of transformation \( FT: T \times U \rightarrow [0, 1] \), where \( T = \{ t : O \rightarrow O \} \), which measures the trust every user agent gives to a specific transformation.

We know there are complicated relationships between the users (4*). These relationships are based on the social ties, and to describe them well is not the subject of this study. We are interested only in the degree of the usability of one's knowledge in the repository of another user. So, we define a metric on the set of our user agents \( U \), \( m : U \times U \rightarrow [0, 1] \). This metric is used to model a simple relation between users (4*), \( m(U_i, U_j) \) mean that the knowledge defined by user \( i \) is more useful for the user \( j \) than the knowledge from user \( l \). Every user \( i \) defines the values for \( m(U_i, U_j) \) himself (2*). Thus the value \( m(U_i, U_j) \) measures how much user \( U_i \) trusts user \( U_j \) or how much the agent of the user \( U_i \) relies on the knowledge received from the user \( U_j \). On later phases we could use a set of metrics to represent better the complicated relationships between users. Since users are supposed to give the best trust to their own judgments, we consider \( m(U_i, U_j) = 1 \).

We define the factor of trust of shared transformation \( FTST: T \times U \rightarrow [0, 1] \) as the arithmetic average \( FTST(t, U) = \sum FTST(t, U)m(U, U)/|U| \) for \( U \in U \). This measures the trust of a transformation for a specific user according the set of the available users. Closer relationships between users and bigger \( FTST(t, U) \) lead to bigger \( FTST(t, U) \). Bigger \( FTST(t, U) \) means user \( U \) thinks transformation \( t \) is more likely to draw us nearer to a good result. We also define a factor of trust of shared series of transformations \( FTSS(t, U) = \prod FTST(t, U) \) \( t = (t_1, ..., t_n) \), that will define the trust on a series of transformation according to the available knowledge. The selected function product gives more value to the individual trust of a concrete transformation than, for example, the simple addition. The procedure \( T \) will create a set of possible transformation series \( t, ..., t_n \) ordered by \( FTSS(t) \).

**Knowledge flow**

The system that implements the above model contains the following components:

1. An **external knowledge repository**. It will contain the trust distances to the other agents as defined by the user, the knowledge received from the other agents and parts of the \( u^* \) functions of the related agents.
2. A **local knowledge repository**. Contains a set of transformations \( t \) and the values of...
FTST($t_i$).

3. A calculation subsystem. It will take care to support the most appropriate $P$ procedure according to the current state of the knowledge repository.

4. An user agent. It will take care about the translating user knowledge in a formal language and vice-versa. Obviously the concrete implementation strongly depends on the problem area.

There are two main procedures – updating knowledge repository and generating transformation chains:

A knowledge repository update could be caused by update of any data in the external knowledge repository. In this case the set of transformations and the FTST values should be refreshed.

The answer generation procedure starts with question from the user. The procedure $P$ generated by the calculation subsystem is called to find the best available transformation chains and the result is sent back to the user agent. The agent chooses its own way to present the result to the user and to get the corresponding error.

**Evaluation of the results**

As we do not know about similar projects, we can not compare the results of the system with the results of already existing ones. But we still can get a judge of its usefulness based on the errors in the answers from the user. The user agent will provide an evaluation of the error for every result generated by the calculation subsystem.

4. Technologies

It is is possible to implement the software application model described above with the existing software technologies. Of course the choice of a concrete technology depends on the problem field, knowledge described, resources available, but we can still give some directions in choosing the appropriate framework. “The Semantic Web (SW) provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries” [12]. This is how the information of the SW itself is created: “different communities of practice develop independently, bottom up, and then can connect link by link, like patches sewn together at the edges” [13]. So we stand on this framework to describe tools and technologies to implement the desired application.

SW suggests the OWL (Web Ontology Language) language for knowledge description. This language is designed to support distributed, versioned and inconsistent ontologies over the Web. For a concrete system we can describe an ontology in this language and base further knowledge development on it. The ontology must be common for all users and must well respond to their view of the problem field. This itself is a task of creating a community good based on the community shared knowledge, but we will pass along it for now and will accept there already exists an ontology that is good enough for all users to share knowledge trough it. Because we want be able to deduce new knowledge from the already shared, we will limit the language used to OWL DL.

Defining transformations in the terms of the SW means to define procedures using rules based on the ontology graph. There are various attempts to define language for rules description around the OWL, but the most common choice it the SWRL [14] as long as it is currently a submission from W3C.

We will also need a common ontology to define the common characteristics of the models, like the metrics of the trust between users, agent properties like location, protocol supported, etc.

To exchange knowledge in the form of ontologies we could use simply access to the ontology description text trough the Web. So the simplest way to retrieve the ontology content is the HTTP.

In order to improve the communication and synchronization between the different agents we could use Web Services or even Semantic Web Services[15].

The inference of knowledge and its potentialities will be based on the selected representation method. Thus we are limited by OWL DL and SWRL in the sphere of the description logics and the first order predicate logics. The inferred knowledge will be presented in the way the asserted knowledge is. There are various reasoners for OWL DL and SWRL, so implementors will be able to pick the one that best fits their needs.
5. Conclusions and future development

Based on the above arguments we can try to implement systems that share and use personal knowledge. There are no systems that implement the model described above, but there exist a lot of appropriate problem fields, like GUI translation communities, local neighborhood communities, activity planning communities, etc. These systems will fill a niche in the field of software applications and will make the computers more useful for personal use. Still, there is much more to do in the theoretical field. We will underline some problems we consider as important for future development.

Currently we do not take into account the difference between person-oriented and community-oriented tasks. Solving these two types of problems should be based on specific types of knowledge. Understanding the knowledge type specifics and extending the above model could help in development of concrete systems of the corresponding type.

Any implementation of the model will need a common ontology accepted by all the members of the community. The creation of this ontology is a community-oriented personal knowledge sharing problem itself, which could be based on a prime ontology itself. It will be useful to define and implement a system or a subsystem as a decision of the problem. Currently the SW framework does not support an explicit definition of fuzzy and even inconsistent ontologies. In our system the knowledge will be not only distributed, but also uncertain, which means the global knowledge base and eventually the basic common ontology will contain fuzzy knowledge. Therefore an OWL extension will be useful for description of this type of knowledge.

6. References